TSref

Extension Key: **doc_core_tsref**

Copyright 2000-2004, Kasper Skårhøj, <kasperYYYY@typo3.com>

This document is published under the Open Content License

available from http://www.opencontent.org/opl.shtml

The content of this document is related to TYPO3

- a GNU/GPL CMS/Framework available from www.typo3.com

Revised for TYPO3 3.7

Table of Contents

Introduction

Warning

This document is a *reference* - it does not intend to guide you step by step into TYPO3 or TypoScript. So you should know what you are looking for when coming to this document and then let tutorials do the explanatory work for you.

General information

Case sensitivity:

All names and references in TypoScript are **case sensitive!** This is very important to notice. That means that:

```
myObject = HTML
myObject.Value = <BLINK> HTML - code </BLINK>
```

is not the same as

```
myObject = html
myObject.value = <BLINK> HTML - code </BLINK>
```

The latter case will not be recognized as the content-object "HTML". In this manual the case of objects is therefore important

Data types

Introduction

The values you define to properties in TypoScript is often of a specific format. This table is your guide to these formats.

Eg. if a value is defined as the type "<tag>", you're supposed to supply HTML-code. If it is of the type "resource", it's a reference to a file from the resource-field in the template. If the type is "GraphicColor" a color-definition is expected and you should supply a HTML-valid color-code or RGB-values comma-separated.

All this is seen in this table:

Datatype reference

Datatype:	Examples:	Comment:	Default:
<tag>	**<BODY bgcolor="red">**		
align	**right**	**right / left / center** Decides alignment, typically in HTML-tags	left
VHalign	*Hori.align = right and Vert.align = center:* **r , c**	**r/c/l , t/c/b** Horizontal (right, center, left) , Vertical align (top / center / bottom)	l , t

resource	*From the resourcefield:*
	`toplogo*.gif`
	
	Reference to filesystem:
	`fileadmin/picture.gif`
		1) A reference to a file from the resource-field in the template.
		You can write the exact filename or you can include an asterisk (*) as wildcard.
		It's recommended to include a "*" before the fileextension (see example to the left). This will ensure that the file is still referenced correct even if the template is copied and the file will have it's name prepended with numbers!!
		2) If the value contains a "/" it's expected to be a reference (absolute or relative) to a file on the file-system instead of the resource-field. No support for wildcards.	

imgResource	Here "file" is a imgResource:
	`file = toplogo*.gif`
	`file.width = 200`
	
	GIFBUILDER:
	`file = GIFBUILDER`
	`file {`
	`   ... (GIFBUILDER-properties here)`
	`}`
		1) A "resource" (see above) + imgResource-properties (see example to the left and object-reference below)
		Filetypes can be anything among the allowed types defined in the configuration variable $TYPO3_CONF_VARS["GFX"]["imagefile_ext"] (localconf.php). Standard is pdf,gif,jpg,jpeg,tif,bmp,ai,pcx,tga,png.
		
		2) GIFBUILDER-object	

HTML-code	**<B>Some text in bold</B>**	pure HTML-code	
target	**_top** **_blank** **content**	target in <A>-tag. This is normally the same value as the name of the root-level object that defines the frame.	
imageExtension	**jpg** **web** *(gif or jpg ..)*	Image extensions can be anything among the allowed types defined in the global variable $TYPO3_CONF_VARS["GFX"]["imagefile_ext"] (localconf.php). Standard is pdf,gif,jpg,jpeg,tif,bmp,ai,pcx,tga,png. **The value "web"** is special. This will just ensure that an image is converted to a web imageformat (gif or jpg) if it happens not to be already!	
degree		-90 to 90, integers	
posint / int+		Positive integer	
int		integer (sometimes used generally though another type would have been more appropriate, like "pixels")	
str / string / value		string. (sometimes used generally though another type would have been more appropriate, like "align")	
boolean	**1**	boolean non-empty strings (but not zero) are "true"	
rotation		integer, degrees from 0 - 360	

Datatype:	Examples:	Comment:	Default:
x,y,w,h	**10,10,5,5**	x,y is the offset from the upper left corner. w,h is the width and height	
HTML-color	**red** **#ffeecc**	**HTML-color codes:** Black = "#000000" Silver = "#C0C0C0" Gray = "#808080" White = "#FFFFFF" Maroon = "#800000" Red = "#FF0000" Purple = "#800080" Fuchsia = "#FF00FF" Green = "#008000" Lime = "#00FF00" Olive = "#808000" Yellow = "#FFFF00" Navy = "#000080" Blue = "#0000FF" Teal = "#008080" Aqua = "#00FFFF"	
GraphicColor	**red**　　(HTML-color) **#ffeecc**　(HTML-color) **255,0,255**　(RGB-integers) *Extra:* **red** *: *0.8*　("red" is darkend by factor 0.8) **#ffeecc** *: +16*　("ffeecc" is going to #fffedc because 16 is added)*	The color can be given as HTML-colors or as a comma-seperated list of RGB-values (integers) You can add an extra parameter that will modify the color mathematically: Syntax: [colordef] : [modifier] where modifier can be and integer which is added/subtracted to the three RGB-channels or a floatingpoint with an "*" before, which will then multiply the values with that factor.	
page_id	**this** **34**	A page id (int) or "this" (=current page id)	
pixels	**345**	pixel-distance	
list	**item,item2,item3**	list of values	
margins	*This sets leftmargin to 10 and bottom-margin to 5. Top and right is not set (zero)* **10,0,0,5**	l,t,r,b left, top, right, bottom	
wrap	*This will cause the value to be wrapped in a font-tag coloring the value red:* **	**	<...>	</...> Used to wrap something. The part on the left and right of the vertical line is placed on the left and right side of the value.	
linkWrap	*This will make a link to the root-level of a website:* **<A HREF="?id={0}">	</A>**	<.. {x}.>	</...> {x}; x is an integer (0-9) and points to a key in the PHP-array rootLine. The key is equal to the level the current page is on measured relatively to the root of the website. If the key exists the uid of the level that key pointed to is inserted instead of {x}. Thus we can insert page_ids from previous levels.	
case	**upper**	"upper" / "lower" Case-conversion	
space	**5	5**	"before	after" Used for content and sets space "before	after".	

Datatype:	Examples:		Comment:	Default:
date-conf	**d-m-y**	*(dd-mm-yy format)*	See PHP function Date()! a - "am" or "pm" A - "AM" or "PM" d - day of the month, numeric, 2 digits (with leading zeros) D - day of the week, textual, 3 letters; i.e. "Fri" F - month, textual, long; i.e. "January" h - hour, numeric, 12 hour format H - hour, numeric, 24 hour format i - minutes, numeric j - day of the month, numeric, without leading zeros l (lowercase 'L') - day of the week, textual, long; i.e. "Friday" m - month, numeric M - month, textual, 3 letters; i.e. "Jan" s - seconds, numeric S - English ordinal suffix, textual, 2 characters; i.e. "th", "nd" U - seconds since the epoch Y - year, numeric, 4 digits w - day of the week, numeric, 0 represents Sunday y - year, numeric, 2 digits z - day of the year, numeric; i.e. "299"	

Datatype:	Examples:	Comment:	Default:
strftime-conf	Date "DD-MM-YY" = **%e:%m:%y** Time "HH:MM:SS" = **%H:%M:%S** or just **%T**	%a - abbreviated weekday name according to the current locale %A - full weekday name according to the current locale %b - abbreviated month name according to the current locale %B - full month name according to the current locale %c - preferred date and time representation for the current locale %C - century number (the year divided by 100 and truncated to an integer, range 00 to 99) %d - day of the month as a decimal number (range 00 to 31) %D - same as %m/%d/%y **%e - day of the month as a decimal number, a single digit is preceded by a space (range ' 1' to '31')** %h - same as %b **%H - hour as a decimal number using a 24-hour clock (range 00 to 23)** %I - hour as a decimal number using a 12-hour clock (range 01 to 12) %j - day of the year as a decimal number (range 001 to 366) **%m - month as a decimal number (range 01 to 12)** **%M - minute as a decimal number** %n - newline character %p - either `am' or `pm' according to the given time value, or the corresponding strings for the current locale %r - time in a.m. and p.m. notation %R - time in 24 hour notation **%S - second as a decimal number** %t - tab character %T - current time, equal to %H:%M:%S %u - weekday as a decimal number [1,7], with 1 representing Monday %U - week number of the current year as a decimal number, starting with the first Sunday as the first day of the first week %V - The ISO 8601:1988 week number of the current year as a decimal number, range 01 to 53, where week 1 is the first week that has at least 4 days in the current year, and with Monday as the first day of the week. %W - week number of the current year as a decimal number, starting with the first Monday as the first day of the first week %w - day of the week as a decimal, Sunday being 0 %x - preferred date representation for the current locale without the time %X - preferred time representation for the current locale without the date **%y - year as a decimal number without a century (range 00 to 99)** %Y - year as a decimal number including the century %Z - time zone or name or abbreviation %% - a literal `%' character	
UNIX-time	*Seconds to 07/04 2000 23:58:* **955144722**	Seconds since 1/1 1970...	
path	*fileadmin/stuff/*	path relative to the directory from which we operate.	
<tag>-data		**Example:** *<frameset>-data: row* could be '*150,**'	
<tag>-params		**Example:** *<frameset>-params* could be '*border="0" framespacing="0"*'	

Datatype:	Examples:	Comment:	Default:

getText	*get content from the $cObj->data-array [header]:*
	= field : header
	get content from the $cObj->parameters-array[color]:
	= parameters : color
	get content from the $GLOBALS ["TSFE"]->register[color]:
	= register : color
	get the title of the page on the first level of the rootline:
	= leveltitle : 1
	get the title of the page on the level right below the current page AND if that is not present, walt to the bottom of the rootline until there's a title:
	= leveltitle : -2 , slide
	get the id of the root-page of the website (level zero)
	= leveluid : 0
	Gets the value of the user defined field "user_myExtField" in the root line (requires additional config in TYPO3_CONF_VARS to include field!)
	= levelfield : -1 , user_myExtField , slide
	get the env var HTTP_REFERER:
	= getenv : HTTP_REFERER
	get the env variable $HTTP_COOKIE_VARS [some_cookie]:
	**= global : HTTP_COOKIE_VARS
		some_cookie**
		get the current time formatted dd-mm-yy:
		= date : d-m-y
		get the current page-title:
		= page : title
		get the current value:
		= current : 1
		get input value from query string, (&stuff=)
		= GPvar : stuff
		get input value from query string, (&stuff[key]=)
		**= GPvar : stuff
			key**
			get the current id
			= TSFE : id
			get the value of the header of record with uid 234 from table tt_content:
			= DB : tt_content:234:header
			Gets the title of the page right before the start of the current website:
			= fullRootLine : -1, title
			Returns localized label for logout button
			=
			LLL:EXT:css_styled_content/pi1/loc allang.php:login.logout
			Outputs the current root-line visually in HTML:
			= debug : rootLine
				This returns a value from somewhere in PHP-array, defined by the type. The syntax is "type : pointer"
				
				field : [fieldname from the current $cObj->data-array in the cObj.]
				As default the *$cObj->*data-array is $GLOBALS ["TSFE"]->page (record of the current page!)
				In TMENU: *$cObj->*data is set to the page-record for each menuitem.
				In CONTENT/RECORDS *$cObj->*data is set to the actual record
				In GIFBUILDER *$cObj->*data is set to the data GIFBUILDER is supplied with.
				parameters : [fieldname from the current *$cObj->*parameters-array in the cObj.]
				See ->parseFunc!
				register : [fieldname from the $GLOBALS["TSFE"]->register]
				See cObject "LOAD_REGISTER"
				leveltitle, leveluid, levelmedia: [levelTitle, uid or media in rootLine, 0- , negative = from behind, " , slide" parameter forces a walk to the bottom of the rootline until there's a "true" value to return. Useful with levelmedia.]
				levelfield: Like "leveltitle" et al. but where the second parameter is the rootLine field you want to fetch. Syntax: [pointer, integer], [fieldname], ["slide"]
				global : [GLOBAL-var, split with
					if you want to get from an array! DEPRECIATED, use GPvar, TSFE or getenv]
					date : [date-conf]
					page : [current page record]
					current : 1 (gets 'current' value)
					level : 1 (gets the rootline level of the current page)
					GPvar: Value from GET or POST method. Use this instead of global
					TSFE: Value from TSFE global main object
					getenv: Value from environment vars
					getIndpEnv: Value from t3lib_div::getIndpEnv()
					DB: Value from database, syntax is [tablename] : [uid] : [field]. Any record from a table in TCA can be selected here. Only marked-deleted records does not return a value here.
					fullRootLine : This gets the title "1. page before" in a page tree like the one below provided we are are the page "Here you are!" (or "Site root") and this TypoScript is in the template with root at "Site root". Red numbers indicate what values of *keynumber* would point to:
					- Page tree root -2
					  |- 1. page before -1
					    |- Site root (root template here!) 0
					      |- Here you are! 1
					
					LLL: Reference to a locallang label. Reference consists of [fileref]:[labelkey]
					cObj: [internal variable from list: "parentRecordNumber"]
					debug: Returns HTML formated content of PHP variable defined by keyword. Available keys are "rootLine", "fullRootLine", "data"
					-------------------- 	
					Getting more values.
					By separating the value of getText with "//" (double slash) you let getText fetch the first value. If it appears empty ("" or zero) the next value is fetched and so on. Example:
					= field:header // field:title // field:uid
					This gets "title" if "header" is empty. If "title" is also empty it gets field "uid"
					fullRootLine :
					Getting a value from the rootline (TSFE->rootLine) going all the way to the root of the page tree.
					[keynumber, fieldname, "slide"]
					Keynumber is relative to the current site root, so negative numbers refer to subsequent levels below the site root. Fieldname is the name of the datafield to get. For "slide", see *levelmedia* type above.	

Datatype:	Examples:	Comment:	Default:
dir	*returns a list of all pdf, gif and jpf-filer from fileadmin/files/ sorted by their name* reversely and with the full path (with "*fileadmin/files/*" prepended) **fileadmin/files/ | pdf,gif,jpg | name** | r | true	[path relative to the webroot of the site] | [list of valid extensions] | [sorting: name, size, ext, date] | [reverse: "r"] | [return full path: boolean Files matching is returned in a comma-separated string. **Note:** The value of config-option "lockFilePath" must equal the first part of the path. Thereby the path is locked to that folder.	
function-name	Function: **user_reverseString** Method in class: **user_stringReversing->reverseString**	Indicates a function or method in a class to call. See more information at the USER cObject. Depending on implementation the class or function name (but not the method name) should probably be prefixed with "user_". This can be changed in the TYPO3_CONF_VARS config though. Also the function / method is normally called with 2 parameters, typ. $conf (TS config) and $content (some content to be processed and returned) Also if you call a method in a class, it is checked (when using the USER/USER_INT objects) whether a class with the same name, but prefixed with "ux_" is present and if so, this class is instantiated instead. See "Inside Typo3" document for more information on extending the classes in Typo3!	

[tsref:(datatypes)]

Data types: Object types
This is some "data-types" that might be mentioned and valid values are shown here below:

Datatype:	Examples:	Comment:	Default:
cObject		HTML / TEXT / IMAGE .... (see "Content Objects" section also called "cObjects")	
frameObj		FRAMESET / FRAME	
menuObj		GMENU / TMENU / IMGMENU / JSMENU **.sectionIndex** **.alternativeSortingFields** (see the menu-object pages later)	
GifBuilderObj		TEXT / SHADOW / OUTLINE / EMBOSS / BOX / IMAGE / EFFECT	

Objects and properties

Introduction

Reference to objects:
Whenever you see ->[objectname] in the tables it means that the property is an object "objectname" with properties from object objectname. You don't need to define the objecttype.

Calculating values (+calc):
Sometimes a datatype is set to "something +calc". "+calc" indicates that the value is calculated with "+-/*". Be aware that the operators has no "weight". The calculation is done in the order of the operators.

Example:
45 + 34 * 2 = 158   (which is the same as this is ordinary arithmetic: (45+34)*2=158)

"... / stdWrap":
When a datatype is set to "type / stdWrap" it means that the value is parsed through the stdWrap function with the properties of the value as parameters.

Example:
pixels / stdWrap: Here the value should be set to pixels and parsed through stdWrap.

In a real application we could do like this:

```
.pixels.field = imagewidth
.pixels.intval = 1
```

This example imports the value from the field "imagewidht" of the current $cObj->data-array. But we don't trust the result to be an integer so we parse it through the the intval()-function.

optionSplit:
optionSplit is a very tricky function. It's primarily used in the menu-objects where you define properties of a whole bunch of items at once. Here the value of properties would be parsed through this function an depending on your setup you could eg. let the last menu-item appear with another color than the other.

The syntax is like this:

|*| - splits the value in parts first, middle, last.

|| - splits each of the first, middle, last in subparts

1. The priority is last, first, middle.

2. If the middle-value is empty (""), the last part of the first-value is repeated.

3. If the first- or middle value is empty, the first part of the last-value is repeated before the last value

4. The middle value is rotated.

ex: first1 || first2 |*| middle1 || middle2 || middle3 |*| last1 || last 2

Examples:
This is very complex and you might think that this has gone too far. But it's actually usefull.

Now consider a menu with five items:

Introduction
Who are we?
Business
Contact
Links

... and a configuration like this (taken from the example-code on the first pages):

```
temp.topmenu.1.NO {
  backColor = red
  ....
}
```

If you look in this reference (see later) at the linkWrap-property of the GMENU-object, you'll discover that all properties of .NO is parse through *optionSplit*. This means that before the individual menuitems are generated, the properties are split by this function. Now lets look at some examples:

Subparts ||

Example:
All items take on the same value. Only the *first*-part is defined and thus it's repeated to all elements

TS: backColor = red

Introduction	(red)
Who are we?	(red)
Business	(red)
Contact	(red)
Links	(red)

Example:
Here the *first*-part is split into subparts. The third subpart is repeated because the menu has five items.

TS: backColor = red || yellow || green

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
Business	(green)	first, subpart 3
Contact	(green)	first, subpart 3 (repeated)
Links	(green)	first, subpart 3 (repeated)

Parts |*|

Example:
Now a *middle*-value is also defined ("*white*"). This means that after the first two menu-items the *middle*-value is used.

TS: backColor = red || yellow **|*|** white

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
Business	(white)	middle
Contact	(white)	middle
Links	(white)	middle

Example:
Now a *last*-value is also defined ("*blue || olive*"). This means that after the first two menu-items the *middle*-value is used.

TS: backColor = red || yellow **|*|** white **|*|** blue || olive

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
Business	(white)	middle
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

... and if we expand the menu a bit (*middle*-value is repeated!)

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
Business	(white)	middle
....	(white)	middle
....	(white)	middle
....	(white)	middle
....	(white)	middle
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

... and if we contract the menu to only four items (the *middle*-value is discarted as it's priority is the least)

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

... and if we contract the menu to only 3 items (the last subpart of the *first*-value is discarted as it's priority is less than the *last*-value)

Introduction	(red)	first, subpart 1
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

"1: The priority is *last, first, middle*"
Now the last two examples showed that the *last*-value has the highest priority, then the *first*-value and then the *middle*-value.

"2: If the *middle*-value is empty, the last part of the first-value is repeated"

Example:
The *middle*-value is left out now. Then subpart 2 of the first value is repeated. *Please observe that no space must exist between the two |*||*|!*

```
TS:   backColor = red || yellow |*||*| blue || olive
```

Introduction	(red)	first, subpart 1
Who are we?	(yellow)	first, subpart 2
Business	(yellow)	first, subpart 2 (repeated)
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

"3: If the *first*- or *middle* value is empty, the first part of the last-value is repeated before the last value"

Example:
The *middle*-value and *first*-value is left out now. Then the subpart 1 of the last value is repeated. *Please observe that no space must exist between the two |*||*|!*

```
TS:   backColor =  |*||*| blue || olive
```

Introduction	(blue)	last, subpart 1 (repeated)
Who are we?	(blue)	last, subpart 1 (repeated)
Business	(blue)	last, subpart 1 (repeated)
Contact	(blue)	last, subpart 1
Links	(olive)	last, subpart 2

"4: The *middle* value is rotated"

Example:
```
TS:   backColor = red |*| yellow || green |*|
```

Introduction	(red)	first
Who are we?	(yellow)	middle, subpart 1
Business	(green)	middle, subpart 2
....	(yellow)	middle, subpart 1
....	(green)	middle, subpart 2
....	(yellow)	middle, subpart 1
....	(green)	middle, subpart 2
Contact	(yellow)	middle, subpart 1
Links	(green)	middle, subpart 2

Conditions

Condition reference:

General note:
Values are normally trimmed for whitespace before comparison.

browser

Syntax:
`[browser = browser1,browser2,...]`

Values and comparison:

Browser:	Identification:
Microsoft Internet Explorer	msie
Netscape Communicator	netscape
Lynx	lynx
Opera	opera
PHP fopen	php
AvantGo (www.avantgo.com)	avantgo
Adobe Acrobat WebCapture	acrobat
IBrowse (amiga-browser)	ibrowse
Teleport Pro	teleport
?? (if "mozilla" is not in useragent)	unknown

Each value is compared with the ($browsername.$browserversion, eg. "netscape4.72") in a strstr().

So if the value is "netscape" or just "scape" or "net" all netscape browsers will match.

If the value is "netscape4" all netscape 4.xx browsers will match.

If any value in the list matches the current browser, the condition returns true.

Examples:
This will match with netscape and opera-browsers

`[browser = netscape, opera]`

version

Syntax:
`[version = value1, >value2, =value3, <value4, ...]`

Comparison:
values are floating-point numbers with "." as the decimal separator.

The values may be preceeded by three operators:

Operator:	Function:
[nothing]	The value must be part of the beginning of the version as a string. This means that if the version is "4.72" and the value is "4" or "4.7" it matches. But "4.73" does not match. Example from syntax: "value1"
=	The value must match exactly. Version "4.72" matches only with a value of "4.72"
>	The version must be greather than the value
<	The version must be less than the value

Examples:

This matches with exactly "4.03" browsers

```
[version=  =4.03]
```

This matches with all 4+ browsers and netscape 3 browsers

```
[version=  >4][browser= netscape3]
```

system

Syntax:

```
[system= system1,system2]
```

Values and comparison:

System:	Identification:
Linux	linux
SGI / IRIX	unix_sgi
SunOS	unix_sun
HP-UX	unix_hp
Macintosh	mac
Windows 3.11	win311
Windows NT	winNT
Windows 95	win95
Windows 98	win98
Amiga	amiga

Values are strings an a match happens if one of these strings is the first part of the system-identification.

Fx. if the value is "win9" this will match with "win95" and "win98" systems.

Examples:

This will match with windows and mac -systems only

```
[system= win,mac]
```

device

Syntax:

```
[device= device1, device2]
```

Values and comparison:

Device:	Identification:
HandHeld	pda
WAP phones	wap
Grabbers:	grabber
Indexing robots:	robot

Values are strings an a match happens if one of these strings equals the type of device

Examples:

This will match WAP-phones and PDA's

```
[device= wap, pda]
```

useragent

Syntax:

```
[useragent= agent]
```

Values and comparison:

This is a direct match on the useragent string from getenv("HTTP_USER_AGENT")

You have the options of putting a "*" at the beginning and/or end of the value *agent* thereby matching with this wildcard!

Examples:

If the HTTP_USER_AGENT is "Mozilla/4.0 (compatible; Lotus-Notes/5.0; Windows-NT)" this will match with it:

```
[useragent = Mozilla/4.0 (compatible; Lotus-Notes/5.0; Windows-NT)]
```

This will also match with it:

```
[useragent = *Lotus-Notes*]
```

... but this will also match with a useragent like this: "Lotus-Notes/4.5 (Windows-NT)"

A short list of user-agent strings and a proper match:

HTTP_USER_AGENT:	Agent description:	Matching condition:
Nokia7110/1.0+(04.77)	Nokia 7110 WAP phone	[useragent= Nokia7110*]
Lotus-Notes/4.5 (Windows-NT)	Lotus-Notes browser	[useragent= Lotus-Notes*]
Mozilla/3.0 (compatible; AvantGo 3.2)	AvantGo browser	[useragent= *AvantGo*]
Mozilla/3.0 (compatible; WebCapture 1.0; Auto; Windows)	Adobe Acrobat 4.0	[useragent= *WebCapture*]

WAP-agents:

This is some of the known WAP agents:

HTTP_USER_AGENT	HTTP_USER_AGENT (continued)
ALAV UP/4.0.7	PLM's WapBrowser
Alcatel-BE3/1.0 UP/4.0.6c	QWAPPER/1.0
AUR PALM WAPPER	R380 2.0 WAP1.1
Device V1.12	SIE-IC35/1.0
EricssonR320/R1A	SIE-P35/1.0 UP/4.1.2a
fetchpage.cgi/0.53	SIE-P35/1.0 UP/4.1.2a
Java1.1.8	UP.Browser/3.01-IG01
Java1.2.2	UP.Browser/3.01-QC31
m-crawler/1.0 WAP	UP.Browser/3.02-MC01
Materna-WAPPreview/1.1.3	UP.Browser/3.02-SY01
MC218 2.0 WAP1.1	UP.Browser/3.1-UPG1
Mitsu/1.1.A	UP.Browser/4.1.2a-XXXX
MOT-CB/0.0.19 UP/4.0.5j	UPG1 UP/4.0.7
MOT-CB/0.0.21 UP/4.0.5m	Wapalizer/1.0
Nokia-WAP-Toolkit/1.2	Wapalizer/1.1
Nokia-WAP-Toolkit/1.3beta	WapIDE-SDK/2.0; (R320s (Arial))
Nokia7110/1.0 ()	WAPJAG Virtual WAP
Nokia7110/1.0 (04.67)	WAPJAG Virtual WAP
Nokia7110/1.0 (04.67)	WAPman Version 1.1 beta:Build W2000020401
Nokia7110/1.0 (04.69)	WAPman Version 1.1
Nokia7110/1.0 (04.70)	Waptor 1.0
Nokia7110/1.0 (04.71)	WapView 0.00
Nokia7110/1.0 (04.73)	WapView 0.20371
Nokia7110/1.0 (04.74)	WapView 0.28
Nokia7110/1.0 (04.76)	WapView 0.37
Nokia7110/1.0 (04.77)	WapView 0.46
Nokia7110/1.0 (04.80)	WapView 0.47
Nokia7110/1.0 (30.05)	WinWAP 2.2 WML 1.1
Nokia7110/1.0	wmlb
	YourWap/0.91
	YourWap/1.16
	Zetor

language

Syntax:
```
[language = lang1, lang2, ...]
```

Comparison:
The values must be a straight match with the value of getenv("HTTP_ACCEPT_LANGUAGE") from PHP. Alternatively, if the value is wrapped in "*" (eg. "*en-us*") then it will split all languages found in the HTTP_ACCEPT_LANGUAGE string and try to match the value with any of those parts of the string. Such a string normally looks like "de,en-us;q=0.7,en;q=0.3" and "*en-us*" would match with this string.

IP

Syntax:
```
[IP = ipaddress1, ipaddress2, ...]
```

Comparison:
The values are compared with the getenv("REMOTE_ADDR") from PHP.

You may include "*" instead of one of the parts in values. You may also list the first one, two or three parts and only they will be tested.

Examples:
These examples will match any IP-address starting with "123":
```
[IP = 123.*.*.*]
```
or
```
[IP = 123]
```

This examples will match any IP-address ending with "123" or being "192.168.1.34":
```
[IP = *.*.*.123][IP = 192.168.1.34]
```

hostname

Syntax:
```
[hostname = hostname1, hostname2, ...]
```

Comparison:
The values are compared with the fully qualiteid hostname of getenv("REMOTE_ADDR") retrieved by PHP.

Value is comma-list of domain names to match with. *-wildcard allowed but cannot be part of a string, so it must match the full host name (eg. myhost.*.com => correct, myhost.*domain.com => wrong)

hour

Syntax:
```
[hour = hour1, >hour2, <hour3, ...]
```

Comparison:
The values in floating point are compared with the current hour (24-hours-format) of the server.

Operator:	Function:
[nothing]	Requires exact match
>	The hour must be greather than the value
<	The hour must be less than the value

minute
See "Hour" above. Same syntax!

Syntax:
`[minute = ...]`

Comparison:
Minute of hour, 0-59

dayofweek
See "Hour" above. Same syntax!

Syntax:
`[dayofweek = ...]`

Comparison:
Day of week, starting with sunday being 0 and saturday being 6

dayofmonth
See "Hour" above. Same syntax!

Syntax:
`[dayofmonth = ...]`

Comparison:
Day of month, 1-31

month
See "Hour" above. Same syntax!

Syntax:
`[month = ...]`

Comparison:
Month, january being 1 and december being 12

usergroup

Syntax:
`[usergroup = group1-uid, group2-uid, ...]`

Comparison:
The comparison can only return true if the grouplist is not empty (global var "gr_list").

The values must either exists in the grouplist OR the value must be a "*".

Example:
This matches all logins

        [usergroup = *]

This matches logins from users members of groups with uid's 1 and/or 2:

```
        [usergroup = 1,2]
```

loginUser

Syntax:
```
[loginUser = fe_users-uid, fe_users-uid, ...]
```

Comparison:
Matches on the uid of a logged in fe_user. Works like 'usergroup' above including the * wildcard to select ANY user.

Example:
This matches any login (use this instead of "[usergroup = *]" to match when a user is logged in!):
```
        [loginUser = *]
```

treeLevel

Syntax:
```
[treeLevel = levelnumber, levelnumber, ...]
```

Comparison:
This checks if the last element of the rootLine is at a level corresponding to one of the figures in "treeLevel". Level = 0 is the "root" of a website. Level=1 is the first menuen

Example:
This changes something with the template, if the page viewed is on level either level 0 (basic) or on level 2
```
        [treeLevel = 0,2]
```

PIDinRootline

Syntax:
```
[PIDinRootline = pages-uid, pages-uid, ...]
```

Comparison:
This checks if one of the figures in "treeLevel" is a PID (pages-uid) in the rootline

Example:
This changes something with the template, if the page viewed is or is a subpage to page 34 or page 36
```
        [PIDinRootline = 34,36]
```

PIDupinRootline

Syntax:
```
[PIDupinRootline = pages-uid, pages-uid, ...]
```

Comparison:
Do the same as PIDinRootline, except the current page-uid is excluded from check.

globalVars:

Syntax:
```
[globalVar=   var1=value,  var2<value2, var3>value3, ...]
```

Comparison:

The values in floating point are compared with the global var "var1" from above.

Operator:	Function:
[nothing]	Requires exact match
>	The var must be greather than the value
<	The var must be less than the value

globalString:

Syntax:

```
[globalString=  var1=value,  var2= *value2, var3= *value3*, ...]
```

Comparison:

This is a direct match on global strings.

You have the options of putting a "*" at the beginning and/or end of the value thereby matching with this wildcard!

Examples:

If the HTTP_HOST is "www.typo3.com" this will match with:

```
[globalString = HTTP_HOST=www.typo3.com]
```

This will also match with it:

```
[globalString = HTTP_HOST= *typo3.com]
```

... but this will also match with a HTTP_HOST like this: "demo.typo3.com"

IMPORTANT NOTE ON GlobalVar and GlobalString:

You can use values from global arrays by deviding the var-name with a "|" (vertical line).

Example: the global var $HTTP_POST_VARS["key"]["levels"] would be retrieved by "HTTP_POST_VARS|key|levels "

Also note that it's recommended to program your scripts in compliance with the php.ini-optimized settings. Please see that file (from your distribution) for details.

Caring about this means that you would get values like HTTP_HOST by getenv(), you would retrieve GET/POST values with t3lib_div::GPvar(). Finally a lot of values from the TSFE object are useful. In order to get those values for comparison with "globalVar" and "globalString" conditions, you prefix that varname with either "IENV"/"ENV:" , "GP:", "TSFE:" or "LIT:" respectively. Still the "|" divider may be used to separate keys in arrays. "LIT" means "literal" and the string after ":" is trimmed and returned as the value (without being divided by "|" or anything)

Notice: Using the "IENV:" prefix is highly recommended to get server/environment variables which are system independant. Basically this will call and return the value from t3lib_div::getIndpEnv() function. With "ENV:" you get the raw output from getenv() which is NOT always the same on all systems! "IENV:" is supported from TYPO3 3.6

Examples:

This will match with a url like "...&print=1"

```
[globalVar = GP:print > 0]
```

This will match with a remote-addr begining with "192.168."

```
[globalString = IENV:REMOTE_ADDR = 192.168.*]
```

This will match with the page-id being higher than 10:

```
[globalVar = TSFE:id > 10]
```

This will match with the pages having the layout field set to "Layout 1":

```
[globalVar = TSFE:page|layout = 1]
```

If the constant {$constant_to_turnSomethingOn} is "1" then this matches:

```
[globalVar = LIT:1 = {$constant_to_turnSomethingOn}]
```

userFunc:

Syntax:
```
[userFunc = user_match(checkLocalIP)]
```

Comparison:
This call the function "user_match" with the first parameter "checkLocalIP". You write that function. You decide what it checks. Function result is evaluated as true/false.

Example:
Put this function in your localconf.php file:

```
function user_match($cmd) {
    switch($cmd)        {
        case "checkLocalIP":
            if (strstr(getenv("REMOTE_ADDR"),"192.168"))    {
                return true;
            }
        break;
        case "checkSomethingElse":
            // ....
        break;
    }
}
```

This condition will return true if the remote address contains "192.168" - which is what your function finds out.

```
[userFunc = user_match(checkLocalIP)]
```

Functions:

stdWrap:

This function is often added as properties to values in TypoScript.

Example with the content-object, "HTML":

```
10 = HTML
10.value = some text
10.value.case = upper
```

Here the content of the object "10" is uppercased before it's returned.

stdWrap properties are executed in the order they appear in the table below. If you want to study this further please refer to tslib/content.php/, function stdWrap()

```
if ($conf["trim"]){$content=trim($content);}
if (   ($conf["required"] && (string)$content=="") || ($conf["if."] && !$this->checkIf($conf
    $content = "";
} else {
    if ($conf["parseFunc."]) {$content=$this->parseFunc($content,$conf["parseFunc."]);}
    if ($conf["split."]){$content=$this->splitObj($content,$conf["split."]);}
    if ((string)$conf["char"]!=""){$content=chr(intval($conf["char"]));}
    if ($conf["intval"]){$content=intval($content);}
    if ($conf["date"]){$content=date($conf["date"], $content);}
    if ($conf["strftime"]){$content=strftime($conf["strftime"], $content);}
    if ($conf["age"]){$content=$this->calcAge(time()-$content,$conf["age"]);}

    if ($conf["case"]){$content=$this->HTMLcaseshift($content, $conf["case"]);}
    if ($conf["bytes"]){$content=$this->bytes($content);}
    if ($conf["crop"]){$content=$this->crop($content, $conf["crop"]);}
    if ($conf["stripHtml"]){$content=strip_tags($content);}
    if ($conf["htmlSpecialChars"]){$content=htmlSpecialChars($content);}

    if ($conf["br"]) {$content=nl2br($content);}
    if ($conf["brTag"]) {$content= ereg_replace(chr(10),$conf["brTag"],$content);}
    if ($conf["keywords"]) {$content= $this->keywords($content);}
    if ($conf["innerWrap"] || $conf["innerWrap."]){$content=$this->wrap($content, $this->st
    if ($conf["innerWrap2"] || $conf["innerWrap2."]){$content=$this->wrap($content, $this->
    if ($conf["fontTag"]){$content=$this->wrap($content, $conf["fontTag"]);}
    if ($conf["textStyle."]) {$content=$this->textStyle($content,$conf["textStyle."]);}
    if ($conf["tableStyle."]) {$content=$this->tableStyle($content,$conf["tableStyle."]);}
    if ($conf["filelink."]) {$content=$this->filelink($content,$conf["filelink."]);}

    if ($conf["wrapAlign"] || $conf["wrapAlign."]){
        $wrapAlign = trim($this->stdWrap($conf["wrapAlign"], $conf["wrapAlign."]));
```

(from tslib/class.tslib_content.php)

Content-supplying properties of stdWrap:

The properties in this table is parsed in the listed order. The properties "data", "field", "current", "cObject" (in that order!) are special as they are used to import content from variables or arrays. The above example could be rewritten to this:

```
10 = HTML
10.value = some text
10.value.case = upper
10.value.field = header
```

Now the line "10.value = some text" is obsolete, because the whole value is "imported" from the field called "header" from the $cObj->data-array.

Property:	Data type:	Description:	Default:
Get data:			
setContentToCurrent	boolean	Sets the current value to the incoming content of the function.	
setCurrent	string /stdWrap	Sets the "current"-value. This is normally set from some outside routine, so be careful with this. But it might be handy to do this	

TYPO3
get.content.right

Property:	Data type:	Description:	Default:

lang	*array of language-keys*	This is used to define optional language specific values.
		If the global language key set by the ->config property .language is found in this array, then this value is used instead of the default input value to stdWrap.
		
		Example:
		`config.language = de`
		`page.10 = TEXT`
		`page.10.value = I am a Berliner!`
		`page.10.lang.de = Ich bin ein Berliner!`
		
		Output will be "Ich bin..." instead of "I am..."	

data	getText		
field	*fieldname*	Sets the content to the value $cObj->data[*field*] **Example:** Set content to the value of field "title": ".field = title" $cObj->data changes. See the description for the data type "getText"/field! **Note:** You can also divide fieldnames by "//". Say, you set "nav_title // title" as the value, then the content from the field nav_title will be returned unless it is a blank string, in which case the title-field's value is returned.	
current	boolean	Sets the content to the "current"-value (see ->split)	
cObject	cObject	Loads content from a content-object	
numRows	->numRows	Returns the number of rows resulting from the select	
filelist	dir / stdWrap	Reads a directory and returns a list of files. The value is exploded by "|" into parameters: 1: The path 2: comma-list of allowed extensions (no spaces between); if empty all extensions goes. 3: sorting: name, size, ext, date 4: reverse: Set to "r" if you want a reversed sorting 5: fullpath_flag: If set, the filelist if returned with complete paths, and not just the filename	
preUserFunc	function-name	Calling a PHP-function or method in a class, passing the current content to the function as first parameter and any properties as second parameter. See .postUserFunc	
Override / Conditions:			
override	string /stdWrap	if "override" returns something else than "" or zero (trimmed), the content is loaded with this!	
preIfEmptyListNum	(as "listNum" below)	(as "listNum" below)	
ifEmpty	string /stdWrap	if the content is empty (trimmed) at this point, the content is loaded with "ifEmpty". Zero's are an empty value.	
listNum	int +calc +"last"	Explodes the content with "," (comma) and the content is set to the item [*value*]. **Special keyword:** "last" is set to the last element of the array! **Property:** .splitChar = string; Defines the string used to explode the value. If splitChar is an integer, the character with that number is used (eg. "10" to split lines...) .stdWrap = stdWrap properties of the listNum... Default = "," (comma) **Examples:** We have a value of "item 1, item 2, item 3, item 4": This would return "item 3": `.listNum = last - 1`	
trim		PHP-function trim(); Removes whitespace around value	
required	boolean	This flag requires the content to be set to some value after any content-import and treatment that might have happend now (data, field, current, listNum, trim). Zero's is NOT regarded as empty! Use "if" instead! If the content i empty, "" is returned immediately.	
if	->if	If the if-object returns false, stdWrap returns "" immediately	
fieldRequired	*fieldname*	value in this field MUST be set	
Parse data:			

TYPO3
get.content.right

Property:	Data type:	Description:	Default:
csConv	string	Convert the charset of the string from the charset given as value to the current rendering charset of the frontend (renderCharset).	
parseFunc	object path reference / ->parseFunc	Processing instructions for the content. **Notice:** If you enter a string as value this will be taken as a reference to an object path globally in the TypoScript object tree. This will be the basis configuration for parseFunc merged with any properties you add here. It works exactly like references does for content elements. **Example:** parseFunc = < lib.parseFunc_RTE parseFunc.tags.myTag = TEXT parseFunc.tags.myTag.value = This will be inserted when <myTag> is found!	
HTMLparser	boolean / ->HTMLparser	This object allows you to parse the HTML-content and make all kinds of advanced filterings on the content. Value must be set and properties are those of ->HTMLparser. (See adminguide for ->HTMLparser options)	
split	->split		

prioriCalc	boolean	Calculation of the value using operators -+*/%^ plus respects priority to + and - operators and parenthesis levels ().
		. (period) is decimal delimiter.
		Returns a doublevalue.
		If .prioriCalc is set to "intval" an integer is returned.
		There is no errorchecking and division by zero or other invalid values may generate strange results. Also you use a proper syntax because future modifications to the function used may allow for more operators and features.
		
		Examples:
		`100%7 = 2`
		`-5*-4 = 20`
		`+6^2 = 36`
		`6 ^(1+1) = 36`
		`-5*-4+6^2-100%7  = 54`
		`-5 * (-4+6) ^ 2 - 100%7 = 98`
		`-5 * ((-4+6) ^ 2) - 100%7 = -22`	

char	int	Content is set to the chr(*value*). PHP: $content=chr(intval($conf["char"]);	
intval	boolean	PHP function intval(); Returns an integer. PHP: $content=intval($content);	

date	date-conf	The content should be data-type "UNIX-time". Returns the content formatted as a date.
		$content=Date($conf["date"], $content);
		
		Example where a timestamp is imported:
		`.value.field = tstamp`
		`.value.date =`	

strftime	*strftime-conf*	Exactly like "date" above. See the PHP-manual (strftime) for the codes, or datatype "strftime-conf". This formatting is useful if the locale is set in advance in the CONFIG-object. See this. Properties: **.charset** : Can be set to the charset of the output string if you need to convert it to renderCharset. Default is to take the intelligently guessed charset from t3lib_cs.	
age	*boolean or string*	If enabled with a "1" (number, integer) the content is seen as a date (UNIX-time) and the difference from present time and the content-time is returned as one of these four variations: "xx min" or "xx hrs" or "xx days" or "xx yrs" The limits between which layout is used are 60 minutes, 24 hours, 365 days, **NOTE:** If you set this property with a non-integer, it's used to format the four units. This is the default value: `" min| hrs| days| yrs"` Set another string if you want to change the units. You may include the "-signs. They are removed anyway.	
case	case	Converts case Uses "renderCharset" for the operation.	

Property:	Data type:	Description:	Default:
bytes	boolean	Will format the input (an integer) as bytes: bytes, kb, mb If you add a value for the property "labels" you can alter the default suffixes. Labels for bytes, kilo, mega and giga are separated by vertical bar (|) and possibly encapsulated in "". Eg: " | K| M| G" (which is the default value) Thus: `bytes.labels = "	K| M| G"`	
substring	[p1], [p2]	Returns the substring with [p1] and [p2] send as the 2nd and 3rd parameter to the PHP substring function. Uses "renderCharset" for the operation.	
removeBadHTML	boolean	Removes "bad" HTML code based on a pattern that filters away HTML that is considered dangerous for XSS bugs.	
stripHtml	boolean	Strips all html-tags.	
crop		Crops the content to a certain length Syntax: +/- (chars) = from left / from right | [string] | [boolean: keep whole words] **Examples:** **20 | ...** => *max 20 characters. If more, the value will be truncated to **first** 20 chars and prepended with "..."* **-20 | ...** => *max 20 characters. If more, the value will be truncated to **last** 20 chars and appended with "..."* **20 | ... | 1** => *max 20 characters. If more, the value will be truncated to **last** 20 chars and appended with "...". If the division is in the middle of a word, the remains of that word is removed.* Uses "renderCharset" for the operation.	
rawUrlEncode	boolean	Passes the content through rawurlencode()-PHP-function	
htmlSpecialChars	boolean	Passes the content through htmlspecialchars()-PHP-function Additional property ".preserveEntities" will preserve entities so only non-entity chars are affected.	
doubleBrTag	string	All double-line-breaks are substituted with this value.	
br	boolean	PHP function nl2br(); Converts linebreaks to -tags	
brTag	string	All ASCII-codes of "10" (CR) is substituted with *value*	
encapsLines	->encapsLines	Lets you split the content by chr(10) and proces each line independently. Used to format content made with the RTE.	
keywords	boolean	splits the content by characters "," ";" and chr(10) (return), trims each value and returns a comma-separated list of the values.	
innerWrap	wrap /stdWrap		
innerWrap2	wrap /stdWrap		
fontTag	wrap		
addParams	->addParams	Lets you add tag-parameters to the content *if* the content is a tag!	
textStyle	->textStyle	Wraps content in font-tags	
tableStyle	->tableStyle	Wraps content with table-tags	
filelink	->filelink	Used to make lists of links to files.	
preCObject	cObject	cObject prepended the content	
postCObject	cObject	cObject appended the content	
wrapAlign	align /stdWrap	Wraps content with <DIV align=[*value*]> | </DIV> *if* align is set	
typolink	->typolink	Wraps the content with a link-tag	
spaceBefore	int /stdWrap	Pixels space before. Done with a clear-gif; <img ...> 	
spaceAfter	int /stdWrap	Pixels space after. Done with a clear-gif; <img ...> 	
space	space	[spaceBefore] | [spaceAfter] **Additional property:** .useDiv = 1 If set, a clear gif is not used by rather a <div> tag with a style-attribute setting the height. (Affects spaceBefore and spaceAfter as well).	
wrap	wrap /+.splitChar	.splitChar defines an alternative splitting character (default is "|" - the vertical line)	

Property:	Data type:	Description:	Default:
noTrimWrap	"special" wrap	This wraps the content with the values val1 and val2 in the example below - including surrounding whitespace! - without trimming the values. Note that this kind of wrap requires a "|" character to begin and end the wrap. **Example:** `| val1 | val2 |`	
wrap2	wrap /+.splitChar	*see .wrap*	
dataWrap		The content is parsed for sections of {...} and the content of {...} is of the type getText and substituted with the result of getText. **Example:** This should result in a font-tag where the fontsize is decided by the global variable "size": `<font size="{global : size}"> | </font>`	
prepend	cObject	cObject prepended to content (before)	
append	cObject	cObject appended to content (after)	
wrap3	wrap /+.splitChar	*see .wrap*	
outerWrap	wrap /stdWrap		

insertData	boolean	If set, then the content string is parsed like .dataWrap above.
		
		Example:
		Displays the page title:
		`10 = TEXT`
		`10.value = This is the page title: {page:title}`
		`10.insertData = 1`	

offsetWrap	x,y	This wraps the input in a table with columns to the left and top that offsets the content by the values of x,y. Based on the cObject OTABLE. **.tableParams / .tdParams** +stdWrap - used to manipulate tableParams/tdParams (default width=99%) of the offset. Default: See OTABLE. **.stdWrap** - stdWrap properties wrapping the offsetWrap'ed output	

postUserFunc	function-name	Calling a PHP-function or method in a class, passing the current content to the function as first parameter and any properties as second parameter. Please see the description of the cObject USER for in-depth information.
		
		Example:
		You can paste this example directly into a new template record.
		
		`page = PAGE`
		`page.typeNum=0`
		`includeLibs.something =`
		`media/scripts/example_callfunction.php`
		
		`page.10 = TEXT`
		`page.10 {`
		`  value = Hello World`
		`  postUserFunc = user_reverseString`
		`  postUserFunc.uppercase = 1`
		`}`
		
		`page.20 = TEXT`
		`page.20 {`
		`  value = Hello World`
		`  postUserFunc = user_various->reverseString`
		`  postUserFunc.uppercase = 1`
		`  postUserFunc.typolink = 11`
		`}`	

postUserFuncInt	function-name	Calling a PHP-function or method in a class, passing the current content to the function as first parameter and any properties as second parameter. The result will be rendered non-cached, outside the main page-rendering. Please see the description of the cObject USER_INT and PHP_SCRIPT_INT for in-depth information. Supplied by Jens Ellerbrock	

Property:	Data type:	Description:	Default:
prefixComment	string	Prefixes content with a HTML comment with the second part of input string (divided by "|") where first part is an integer telling how many trailing tabs to put before the comment on a new line. The content is parsed through insertData. **Example:** prefixComment = 2 | CONTENT ELEMENT, uid:{field:uid}/{field:CType} Will indent the comment with 1 tab (and the next line with 2+1 tabs) (Added in TYPO3 >3.6.0RC1)	

editIcons	string	If not empty, then insert an icon linking to the typo3/alt_doc.php with some parameters to build and backend user edit form for certain fields.
		The value of this property is a list of fields from a table to edit. It's assumed that the current record of the cObj is the record to be edited.
		Syntax: *optional tablename* : *comma list of fieldnames[list of pallette-field names separated by |]*
		
		.beforeLastTag (1,0,-1): If set (1), the icon will be inserted before the last HTML tag in the content. If -1 the icon will be prepended to the content. If zero (0) the icon is appended in the end of the content.
		
		.styleAttribute (string): Adds a style-attribute to the icon image with this value. For instance you can set "position:absolute" if you want a non-destructive insertion of the icon. Notice: For general styling all edit icons has the class "frontEndEditIcons" which can be addressed from the stylesheet of the site.
		
		.iconTitle (string): The title attribute of the image tag.
		
		.iconImg (HTML): Alternative HTML code instead of the default icon shown. Can be used to set another icon for editing (for instance a red dot or otherwise... :-)
		
		Example:
		This will insert an edit icon which links to a form where the header and bodytext fields are displayed and made available for editing (provided the user has access!).
		`editIcons = tt_content : header, bodytext`
		
		Or this line that puts the header_align and date field into a "palette" which means they are displayed on a single line below the header field. This saves some space.
		`editIcons = header[header_align|date], bodytext`	

editPanel	boolean / editPanel	See cObject EDITPANEL.	
debug	boolean	Prints content with HTMLSpecialChars() and <PRE></PRE>: Usefull for debugging which value stdWrap actually ends up with, if you're constructing a website with TypoScript. Should be used under construction only.	
debugFunc	boolean	Prints the content directly to browser with the debug() function. Should be used under construction only. Set to value "2" the content will be printed in a table - looks nicer.	
debugData	boolean	Prints the current data-array, $cObj->data, directly to browser. This is where ".field" gets data from. Should be used under construction only.	

[tsref:->stdWrap]

imgResource:

imgResource is properties that is used with the data type imgResource.

Example:

This scales the image toplogo.gif to the width of 200 pixels

```
file = toplogo.gif
file.width = 200
```

Property:	Data type:	Description:	Default:
ext	imageExtension / stdWrap		web
width	pixels / stdWrap	If both the width and heigth is set and one of the numbers has an "m" after it, the proportions will be preserved and thus width/height will work as max-dimensions for the image.	
height	pixels / stdWrap		
params		ImageMagick command-line: fx. "-rotate 90" or "-negate"	
sample	boolean	If set, -sample is used to scale images instead of -geometry. Sample does not use antialiasing and is therefore much faster.	
alternativeTempPath	string	Enter an alternative path to use for temp images. Must be found in the list in TYPO3_CONF_VARS[FE][allowedTempPaths]	
frame	int	Chooses which frame in an gif-animation or pdf-file. "" = first frame (zero)	

import	path / stdWrap	*value* should be set to the path of the file
		with stdWrap you get the filename from the data-array
		
		Example:
		This returns the first image in the field "image" from the data-array:
		`.import = uploads/pics/`
		`.import.field = image`
		`.import.listNum = 0`	

maxW	pixels / stdWrap	Max width	
maxH	pixels / stdWrap	Max height	
minW	pixels	Min width (overrules maxW/maxH)	
minH	pixels	Min height (overrules maxW/maxH)	
Masking: (Black hides, white shows)			
m.mask	imgResource	The mask by which the image is masked onto "m.bgImg". Both "m.mask" and "m.bgImg" **is scaled to fit** the size of the imgResource image! **NOTE:** Both "m.mask" and "m.bgImg" must be valid images.	
m.bgImg	imgResource	**NOTE:** Both "m.mask" and "m.bgImg" must be valid images.	
m.bottomImg	imgResource	An image masked by "m.bottomImg_mask" onto "m.bgImg" before the imgResources is masked by "m.mask". Both "m.bottomImg" and "m.bottomImg_mask" **is scaled to fit** the size of the imgResource image! This is most often used to create an underlay for the imgResource. **NOTE:** Both "m.bottomImg" and "m.bottomImg_mask" must be valid images.	
m.bottomImg_mask	imgResource	(optional) **NOTE:** Both "m.bottomImg" and "m.bottomImg_mask" must be valid images.	

[tsref:->imgResource]

imageLinkWrap:

This object wraps the input (an image) with a link to the script "showpic.php" with parameters that define such things as the size of the image, the backgroundcolor of the new window and so on.

An md5-hash of the parameters is generated. The hash is also generated in "showpic.php" and the hashes MUST match in order for the image to be shown. This is a safety feature in order to prevent users from changing the parameters in the url themselves.

PHP-function: $cObj->imageLinkWrap()

Property:	Data type:	Description:	Default:
width	int (1-1000)	If you add "m" to either the width or height, the image will be held in proportions and width/height works as max-dimensions	
height	int (1-1000)	see ".width"	
effects	*see GIFBUILDER / effects. (from stdgraphics-library)*	**Example:** `gamma=1,3	sharpen=80	solarize=70`	
sample	boolean	If set, -sample is used to scale images instead of -geometry. Sample does not use antialiasing and is therefore much faster.	
alternativeTempPath		Enter an alternative path to use for temp images. Must be found in the list in TYPO3_CONF_VARS[FE][allowedTempPaths]	
title	string	page title of the new window (HTML)	
bodyTag	<tag>	Body tag of the new window	
wrap	wrap	Wrap of the image, which is output between the body-tags	
target	<A>-data:target	NOTE: Only if ".JSwindow" is set	
JSwindow	boolean	The image will be opened in a new window which is fitted to the dimensions of the image!	
JSwindow.expand	x,y	x and y is added to the window dimensions.	
JSwindow.newWindow	boolean	Each picture will open in a new window!	
JSwindow.altUrl	string/stdWrap	If this returns anything, the URL shown in the JS-window is NOT showpic.php but the url given here!	
typolink	->typolink	NOTE: This overrides the imageLinkWrap if it returns anything!!	
enable	boolean /stdWrap	**The image is linked ONLY if this is true!!**	0

[tsref:->imageLinkWrap]

Example:

```
1.imageLinkWrap = 1
1.imageLinkWrap {
  enable = 1
  bodyTag = <BODY bgColor=black>
  wrap = <A href="javascript:close();"> | </A>
  width = 800m
  height = 600

  JSwindow = 1
  JSwindow.newWindow = 1
  JSwindow.expand = 17,20
}
```

numRows:

This object return the number of rows

Property:	Data type:	Description:	Default:
table	tablename		
select	->select	Select query for the operation. The property "selectFields" is overridden internally with "count(*)".	

[tsref:->numRows]

select:

This object generates an SQL-select statement needed to select records from the database.

Some records are hidden or timed by start and end-times. This is automatically added to the SQL-select by looking in the tables.php-array (enablefields)

Also, if the "pidInList" feature is used, any page in the pid-list that is not visible for the user of the website IS REMOVED from the pidlist. Thereby no records from hidden, timed or access-protected pages are selected! Nor records from recyclers.

Property:	Data type:	Description:	Default:
uidInList	*list of* page_id		
pidInList	*list of* page_id / stdWrap		this
orderBy	*SQL-orderBy*	without "order by"! Eg. "sorting, title"	
groupBy	*SQL-groupBy*	without "group by"! Eg. "CType"	
max	int +calc +*"total"*	max records **Special keyword:** "total" is substituted with count(*)	
begin	int +calc +*"total"*	begin with record number *value* **Special keyword:** "total" is substituted with count(*)	
where	*SQL-where*	without "where"!, Eg. " (title LIKE '%SOMETHING%' AND NOT doktype) "	
andWhere	*SQL-where /* *stdWrap*	without "AND"!, Eg. "NOT doktype".	
languageField	string	If set, this points to the field in the record which holds a reference to a record in sys_language table. And if set, the records returned by the select-function will be selected only if the value of this field matches the $GLOBALS["TSFE"]->sys_language_uid (which is set by the config.sys_language_uid option)	
selectFields	string	List of fields to select, or "count(*)".	*
join leftjoin rightjoin	string	Enter tablename for JOIN , LEFT OUTER JOIN and RIGHT OUTER JOIN respectively.	

[tsref:->select]

split:

This object is used to split the input by a character and then parse the result onto some functions.

For each iteration the split index starting with 0 (zero) is stored in the register key SPLIT_COUNT.

Example:

This is an example of TypoScript-code that imports the content of field "bodytext" from the $cObj->data-array (ln 2). The content is split by the linebreak-character (ln 4). The items should all be treated with a stdWrap (ln 5) which imports the value of the item (ln 6). This value is wrapped in a tablerow where the first column is a bullet-gif (ln 7). Finally the whole thing is wrapped in the proper table-tags (ln 9)

```
1       20 = TEXT
2       20.field = bodytext
3       20.split {
4         token.char = 10
5         cObjNum = 1
6         1.current = 1
7         1.wrap = <TR><TD valign="top"><IMG src="dot.gif"></TD><TD valign="top"> | </TD></TR>
8       }
9       20.wrap = <TABLE border="0" cellpadding="0" cellspacing="3" width="368"> | </TABLE><BR>
```

Property:	Data type:	Description:	Default:
token	str / stdWrap	string or character (token) used to split the value	
max	int / stdWrap	max number of splits	
min	int / stdWrap	min number of splits.	
cObjNum	*cObjNum* +optionSplit	This is a pointer the array of this object ("1,2,3,4"), that should treat the items, resulting from the split.	
1,2,3,4	->CARRAY / stdWrap	The object that should treat the value. **NOTE:** The "current"-value is set to the value of current item, when the objects are called. See "stdWrap" / current. **Example (stdWrap used):** `1.current = 1` `1.wrap = <B>	</B>`  **Example (CARRAY used):** `1 {` ` 10 = TEXT` ` 10.current = 1` ` 10.wrap = <B>	</B>` ` 20 = CLEARGIF` ` 20.height = 20` `}`	
wrap	wrap +optionSplit	Defines a wrap for each item.	

[tsref:->split]

if:

This function returns true if ALL of the present conditions are met (they are AND'ed). If a single condition is false, the value returned is false.

The returned value may still be negated by the ".negate"-property.

Property:	Data type:	Description:	Default:
isTrue	str / stdWrap	If the content is "true".... (not empty string and not zero)	
isFalse	str / stdWrap	If the content is "false"... (empty or zero)	
isPositive	int / stdWrap + calc	returns false if content is not positive	
isGreaterThan	value / stdWrap	returns false if content is not greater than ".value"	
isLessThan	value / stdWrap	returns false if content is not less than ".value"	
equals	value / stdWrap	returns false if content does not equal ".value"	
isInList	value / stdWrap	returns false if content is not in the comma-separated list ".value". The list in ".value" may not have spaces between elements!!	
value	value / stdWrap	"value" (the comparison value mentioned above)	
negate	boolean	This negates the result just before it exits. So if anything above returns true the overall returns ends up returning false!!	
directReturn	boolean	If this property exists the true/false of this value is returned. Could be used to set true/false by TypoScript constant	

[tsref:->if]

Explanation:

the "if"-function is a very odd way of returning true or false! Beware!

"if" is normally used to decide whether to render an object or return a value (see the cObjects and stdWrap)

Here is how it works:

The function returns true or false. Whether it returns true or false depends on the properties of this function. Say if you set "isTrue = 1" then result is true. If you set "isTrue.field = header" the function returns true if the field "header" in $cObj->data is set!

If you want to compare values, you must load a base-value in the ".value"-property. Example:

```
.value = 10
.isGreaterThan = 11
```

This would return true because the value of ".isGreaterThan" is greater than 10, which is the base-value.

More complex is this:

```
.value = 10
.isGreaterThan = 11
.isTrue.field = header
.negate = 1
```

There are two conditions - isGreaterThan and isTrue. If they are both true, the total is true (AND) BUT (!) the result if the function in total is false because the ".negate"-flag inverts the result!

Example:

This is a GIFBUILDER object that will write "NEW" on a menu-item if the field "newUntil" has a date less than the current date!

```
...
  30 = TEXT
  30.text = NEW!
  30.offset = 10,10
  30.if {
    value.data = date: U
    isLessThan.field = newUntil
    negate = 1
  }
...
```

typolink:

Wraps the incoming value with link.

If this is used from parseFunc the $cObj->parameters-array is loaded with the link-parameters (lowercased)!

Property:	Data type:	Description:	Default:
extTarget	target /stdWrap	target used for external links	_top
target	target /stdWrap	target used for internal links	
no_cache	boolean /stdWrap	Adds a "&no_cache=1"-parameter to the link	
useCacheHash	boolean	If set, the additionalParams list is exploded and calculated into a hashstring appended to the url, like "&cHash=ae83fd7s87". When the caching mechanism sees this value, it calculates the same value on the server based on incoming values in HTTP_GET_VARS, excluding id,type,no_cache,ftu,cHash values. If the incoming cHash value matches the calculated value, the page may be cached based on this. The [SYS][encryptionKey] is included in the hash in order to make it unique for the server and non-predictable.	
additionalParams	string /stdWrap	This is parameters that are added to the end of the url. This must be code ready to insert after the last parameter. **Example:** '&print=1' '&sword_list[]=word1&sword_list[]=word2' **Applications:** This is very useful when linking to pages from a searchresult. The searchwords are stored in the register-key SWORD_PARAMS and can be insert directly like this: .additionalParams.data = register:SWORD_PARAMS **NOTE:** This is only active for internal links!	
wrap	wrap	Wraps the links.	
ATagBeforeWrap	boolean	If set, the link is first wrapped with ".*wrap"* and then the <A>-tag.	

Property:	Data type:	Description:	Default:

parameter	string /stdWrap	This is the data, that ->typolink uses to create the link. The value is trimmed and if it's empty, ->typolink returns the input value untouched.
		
		NOTE: If used from parseFunc, this value should be imported by:
		`typolink.parameter.data = parameters : allParams`
		
		Examples:
		Internal links:
		integers (51): creates a link to page with uid = 51
		filerefs (fileadmin/somedir/thedoc.html): creates a link to the file on the local server.
		strings (some_alias): creates a link to the page with alias = "some_alias"
		
		External links:
		email-adresses (name@email.com): creates a link to the email-addr.
		domains (www.domain.com): creates link to http://-page
		
		The input is parsed like this:
		First the parameter is splitted by character-space. This provides a way to pass more parameters. See "target" below here.
		If a "@" is in the string, it's an email
		If a period (.) is in the string AND if the period (.) is found before a slash (/) is found OR if a doubleslash is found, then it's a URL
		If a slash (/) is found, it's a filereference. If the file/directory does not exist on the server, the link is NOT made!
		
		Now the input can be an alias or page-id. If the input is an integer it's a page-id, if it's two comma separated integers, it's a id/type pair, else it's an alias. For page-id's or aliases you can prepend a "#" mark with a number indication tt_content record number on the page to jump to! (if .section-property is present, it overrides this).
		If you insert only "#234" where "234" is the tt_content record number, it links to the current page-id
		
		Target
		Target is normally defined by the ".extTarget" and "target" properties of typolink. But you may override this target by adding the new target after the parameter separated by whitespace. Thus the target becomes the 'second' parameter.
		If the 'Target' parameter is set to the '-' character, then it's the same as no target passed to the function. This feature enables you to still pass a class as third parameter without setting the target also.
		
		Open in windows with fixed dimensions (JavaScript)
		It is possible to open the link in a window opened by JavaScript (with "window.open"). For this, just set the target value to "123x456" where 123 is the window width and 456 is the window height. You can also specify additional parameters to the function by entering them separated from the width and height with a colon ":". For instance "230x450:resizable=0,location=1" will disable resizing of the window and enable the location bar.
		Also see property "JSwindow".
		
		Class
		If you specify a 'third parameter' separated by whitespace in the parameter value this becomes the class-parameter of the link. This class parameter is inserted in the link-tag before any values from .ATagParams which means this class value will override any class value set in ATagParams (at least for MSIE).
		
		Examples of multiparameters:
		Consider this .parameter value passed to this function:
		
		`51  _blank  blueLink`
		
		This would result in a link approx like this:
		
		`<A href="?id=51" target="_blank" class="blueLink">`	

JSwindow_params	string	Preset values for opening the window. This example lists almost all possible attributes: status=1,menubar=1,scrollbars=1,resizable=1,location=1,directories=1,toolbar=1	
returnLast	string	If set to "url" then it will return the URL of the link ($this->lastTypoLinkUrl) If set to "target" it will return the target of the link. So, in these two cases you will not get the value wrapped but the url or target value returned!	

Property:	Data type:	Description:	Default:
section	string /stdWrap	If this value is present, it's prepended with a "#" and placed after any internal url to another page in Typo3. This is used create a link, which jumps from one page directly the section on another page.	
ATagParams	<A>-params	Additional parameters **Example:** class="board"	
userFunc	function-name	This passes the link-data compiled by the typolink function to a user-defined function for final manipulation. The $content variable passed to the user-function (first parameter) is an array with the keys "TYPE", "TAG", "url", "targetParams" and "aTagParams". TYPE is an indication of link-kind: mailto, url, file, page TAG is the full <A>-tag as generated and ready from the typolink function. The latter three is combined into the 'TAG' value after this formula: `<A HREF="'.$finalTagParts["url"].'"'.$finalTagParts ["targetParams"].$finalTagParts["aTagParams"].'>` The userfunction must return an <A>-tag.	

[tsref:->typolink]

textStyle

This is used to style text with a bunch of standard options + some site-specific.

Property:	Data type:	Description:	Default:
align.field	align	Set to fieldname from the $cObj->data-array	
face.field	string	Set to fieldname from the $cObj->data-array [1] = "Times New Roman"; [2] = "Verdana,Arial,Helvetica,Sans serif"; [3] = "Arial,Helvetica,Sans serif";	
face.default	string /stdWrap	[default] = User defined	
size.field	string	Set to fieldname from the $cObj->data-array [1] = 1; [2] = 2; [3] = 3; [10] = "+1"; [11] = "-1";	
size.default	string /stdWrap	[default] = User defined	
color.field	string	Set to fieldname from the $cObj->data-array See "content.php" for the colors available	
color.default	string /stdWrap	[default] = User defined	
color.1 color.2	string	[1],[2] = User defined	
properties.field	int	Set to fieldname from the $cObj->data-array The property values goes like this: bit 0: <B> bit 1: <I> bit 2: <U> bit 3: (uppercase) Thus a value of 5 would result in bold and underlined text	
properties.default	int /stdWrap	[default] = User defined (This value will be used whenever ".field" is false!)	
altWrap	wrap	If this value is set, the wrapping with a font-tag based on font,size and color is NOT done. Rather the element is wrapped with this value. Use it to assign a stylesheet by setting this value to eg. <div class="text">	</div>	

[tsref:->textStyle]

encapsLines

Property:	Data type:	Description:	Default:

encapsTagList	list of strings	List of tags which qualify as encapsulating tags. Must be lowercase.
		
		Example:
		`encapsTagList = div, p`
		
		This setting will recognize the red line below as encapsulated lines:
		
		`First line of text`
		`Some <div>text</div>`
		`<p>Some text</p>`
		`<div>Some text</div>`
		`<B>Some text</B>`	

remapTag.[*tagname*]	string	Enter a new tag name here if you wish the tagname of any encapsulation to be unified to a single tag name.
		
		For instance, setting this value to "remapTags.P=DIV" would convert:
		
		`<p>Some text</p>`
		`<div>Some text</div>`
		
		to
		
		`<div>Some text</div>`
		`<div>Some text</div>`
		
		([*tagname*] is in uppercase.)	

addAttributes.
[*tagname*]
	array of strings	Attributes to set in the encapsulation tag.
		
		Example:
		`addAttributes.P {`
		`  style=padding-bottom:0px; margin-top:1px; margin-`
		`bottom:1px;`
		`  align=center`
		`}`
		
		([*tagname*] is in uppercase.)
		
		`.setOnly =`
		exists : This will set the value ONLY if the property does not already exist
		blank : This will set the value ONLY if the property does not already exist OR is blank ("")
		
		Default is to always override/set the attributes value.	

removeWrapping	boolen	If set, then all existing wrapping will be removed.
		
		This:
		
		`First line of text`
		`Some <div>text</div>`
		`<p>Some text</p>`
		`<div>Some text</div>`
		`<B>Some text</B>`
		
		becomes this:
		
		`First line of text`
		`Some <div>text</div>`
		`Some text`
		`Some text`
		`<B>Some text</B>`	

wrapNonWrappedLines	wrap	Wrapping for non-encapsulated lines
		
		Example:
		`.wrapNonWrappedLines = <P>|</P>`
		
		This:
		
		`First line of text`
		`<p>Some text</p>`
		
		becomes this:
		
		`<P>First line of text</P>`
		`<p>Some text</p>`	

innerStdWrap_all	->stdWrap	Wraps the content inside all lines, whether they are encapsulated or not.	

Property:	Data type:	Description:	Default:
encapsLinesStdWrap .[*tagname*]	->stdWrap	Wraps the content inside all encapsulated lines. ([*tagname*] is in uppercase.)	
defaultAlign	string / ->stdWrap	If set, this value is set as the default "align" value of the wrapping tags, both from .encapsTagList, .bypassEncapsTagList and .nonWrappedTag	
nonWrappedTag	tagname	For all non-wrapped lines, you can set here which tag it should be wrapped in. Example would be "P". This is an alternative to . wrapNonWrappedLines and has the advantage that it's attributes are set by .addAttributes as well as defaultAlign. Thus you can easier match the wrapping tags used for nonwrapped and wrapped lines.	

[tsref:->encapsLines]

Example:

```
encapsLines {
  encapsTagList = div,p
  remapTag.DIV = P
  wrapNonWrappedLines = <P>|</P>
  innerStdWrap_all.ifEmpty =  
}
```

This example shows how to handle content rendered by Typo3 and stylesheets where the <P> tag is used to encapsulate each line.

Say, you have made this content with the Rich Text Editor:

```
This is line # 1

[Above is an empty line!]
<DIV align=right>This line is right-aligned</DIV>
```

After being processed by encapsLines with the above configuration, the content looks like this:

```
<P>This is line # 1 </P>
<P> </P>
<P>[Above is an empty line!] </P>
<P align="right">This line is right-aligned</P>
```

Each line is nicely wrapped with <P> tags. The line from the database which was *already* wrapped (but in <DIV>-tags) has been converted to <P>, but keeps it's alignment. Overall, notice that the Rich Text Editor ONLY stored the line which was in fact right-aligned - every other line from the RTE was stored without any wrapping tags, so that the content in the database remains as human readable as possible.

Example:

```
# Make sure nonTypoTagStdWrap operates on content outside <typolist> and <typohead> only:
tt_content.text.20.parseFunc.tags.typolist.breakoutTypoTagContent = 1
tt_content.text.20.parseFunc.tags.typohead.breakoutTypoTagContent = 1
# ... and no <BR> before typohead.
tt_content.text.20.parseFunc.tags.typohead.stdWrap.wrap >
# Setting up  nonTypoTagStdWrap to wrap the text with P-tags
tt_content.text.20.parseFunc.nonTypoTagStdWrap >
tt_content.text.20.parseFunc.nonTypoTagStdWrap.encapsLines {
  encapsTagList = div,p
  remapTag.DIV = P
  wrapNonWrappedLines = <P style="margin:0 0 0;">|</P>

  # Forcing these attributes onto the encapsulation-tags if any
  addAttributes.P {
    style=margin:0 0 0;
  }
  innerStdWrap_all.ifEmpty =  
  innerStdWrap_all.textStyle < tt_content.text.20.textStyle
}
# finally removing the old textstyle formatting on the whole bodytext part.
tt_content.text.20.textStyle >
# ... and <BR>-tag after the content is not needed either...
tt_content.text.20.wrap >
```

This is an example of how to wrap traditional tt_content bodytext with <P> tags, setting the line-distances to regular space like that generated by a <BR> tag, but staying compatible with the RTE features such as assigning classes and alignment to paragraphs.

tableStyle

This is used to style a table-tag. The input is wrapped by this table-tag

Property:	Data type:	Description:	Default:
align	align /stdWrap		
border	int /stdWrap		
cellspacing	int /stdWrap		
cellpadding	int /stdWrap		
color.field	string	Set to fieldname from the $cObj->data-array	
color.default color.1 color.2	string	[default],[1],[2] = User defined	
params	<TABLE>-params		

[tsref:->tableStyle]

Example:

```
styles.content.tableStyle {
  align.field = text_align
  border.field = table_border
  cellspacing.field = table_cellspacing
  cellpadding = 1

  color.field = table_bgColor
  color.default = {$styles.content.tableStyle.color}
  color.1 = {$styles.content.tableStyle.color1}
  color.2 = {$styles.content.tableStyle.color2}
}
```

addParams

Property:	Data type:	Description:	Default:
_offset	int	Use this to define which tag you want to manipulate. 1 is the first tag in the input, 2 is the second, -1 is the last, -2 is the second last	1
(array of strings)	string /stdWrap	This defines the content of each added property to the tag. If there is a tag-property with this name already (case-sensitive!) that property will be overridden! If the returned value is a blank string (but not zero!) then the existing (if any) property will not be overridden.	

[tsref:->addParams]

Example:

```
page.13 = HTML
page.13.value = <tr><td valign=top>
page.13.value.addParams.bgcolor = {$menuCol.bgColor}
page.13.value.addParams._offset = -1
```

Result example:

```
<tr><td valign="top" bgcolor="white">
```

(This example adds the 'bgColor' property to the value of the HTML cObject, if the content is not "". (zero counts as a value here!))

filelink

Input is a filename in the path "path".

icon, size and file is rendered in the listed order.

Property:	Data type:	Description:	Default:
path	path /stdWrap	**Example:** "uploads/media/"	
icon	boolean /stdWrap	Set if icon should be shown	
icon_image_ext_list	*list* of imageextensions	This is the extensions that should render as thumbsnails instead of icons.	
iconCObject	cObject	Enter a cObject to use alternatively for the icons, eg. IMAGE type. If this is set, it'll substitute the use of the thumbs-script for display of thumbnails.	
icon_link	boolean	If the icon should be linked also	
labelStdWrap	-> stdWrap	stdWrap options for the label (by default the label is the filename) before being wrapped with the A-tags. Use this to eg. import another label from a database field or such.	
wrap	wrap	Wraps the links.	
ATagBeforeWrap	boolean	If set, the link is first wrapped with ".*wrap"* and then the <A>-tag.	
file	-> stdWrap	stdWrap of the label (by default the label is the filename) after having been wrapped with A-tag!	
size	boolean /stdWrap	Set if size should be shown	
jumpurl	boolean	Decides if the link should call the script with the jumpurl paramter in order to register any clicks in the stat. This has the advantage that any clicks on the file will register in the stat. The disadvantage is, that users cant right-click and select "Save Target As" in the browser. **Properties:** .secure (boolean) If set, then the file pointed to by jumpurl is NOT redirected to, but rather it's read from the file and returned with a correct header. This option adds a hash and locationData to the url and there MUST be access to the record in order to download the file. If the fileposition on the server is furthermore secured by a .htaccess file preventing ANY access, you've got secure download here! .secure.mimeTypes (list of mimetypes, syntax [ext] = [mimetype] **Example:** .secure = 1 .secure.mimeTypes = pdf=application/pdf, doc=application/msword	
target	target		
stdWrap	->stdWrap		
ATagParams	<A>-params	Additional parameters **Example:** class="board"	
removePrependedNumbers	boolean	if set, any 2-digit prepended numbers ("eg _23") in the filename is removed.	
altText titleText	string / stdWrap	For icons (image made with "iconCObject" must have their own properties) If no titltext is specified, it will use the alttext instead If no alttext is specified, it will use an empty alttext	
longdescURL	string / stdWrap	For icons (image made with "iconCObject" must have their own properties) "longdesc" attribute (URL pointing to document with extensive details about image).	

[tsref:->filelink]

Example:

```
1.filelink {
   path = uploads/media/
   icon = 1
   icon.wrap = <td> | </td>
   size = 1
   size.wrap = <td> | </td>
```

```
        file.fontTag = {$styles.content.uploads.wrap}
        file.wrap = <td> | </td>
        jumpurl = 1
        target = _blank
        stdWrap = <tr> | </tr>
    }
```

TYPO3
get.content.right

parseFunc:

This object is used to parse some content for stuff like special typo tags, the "makeLinks"-things and so on...

Example:

This example takes the content of the field "bodytext" and parses it through the makelinks-functions and substitutes all <LINK> and <TYPOLIST>-tags with something else.

```
tt_content.text.default {
  20 = TEXT
  20.field = bodytext
  20.wrap = | <BR>
  20.brTag = <br>
  20.parseFunc {
    makelinks = 1
    makelinks.http.keep = path
    makelinks.http.extTarget = _blank
    makelinks.mailto.keep = path
    tags {
      link = TEXT
      link {
        current = 1
        typolink.extTarget = _blank
        typolink.target={$cLinkTagTarget}
        typolink.wrap = <B><FONT color=red>|</FONT></B>
        typolink.parameter.data = parameters : allParams
      }

      typolist < tt_content.bullets.default.20
      typolist.trim = 1
      typolist.field >
      typolist.current = 1
    }
  }
}
```

20 = TEXT

Property:	Data type:	Description:	Default:

externalBlocks	list of tagnames/+properties	This allows you to pre-split the content passed to parseFunc so that only content outside the blocks with the given tags is parsed.
		Extra properties:
		.[tagname] {
		  callRecursive = [boolean]; If set, the content of the block is directed into parseFunc again. Otherwise the content is just passed through with no other processing than stdWrap (see below)
		  callRecursive.dontWrapSelf = [boolean]; If set, the tags of the block is *not* wrapped around the content returned from parseFunc.
		  callRecursive.alternativeWrap = Alternative wrapping instead of the original tags.
		  callRecursive.tagStdWrap = ->stdWrap processing of the block-tags.
		  stdWrap = ->stdWrap processing of the whole block (regardless of whether callRecursive was set.)
		  stripNLprev = [boolean]; Strips off last linebreak of the previous outside block
		  stripNLnext = [boolean]; Strips off first linebreak of the next outside block
		  stripNL = [boolean]: Does both of the above.
		
		  HTMLtableCells = [boolean]; If set, then the content is expected to be a table and every table-cell is traversed.
		  # Below, default is all cells and 1,2,3... overrides for specific cols.
		  HTMLtableCells.[default/1/2/3/...] {
		    callRecursive = [boolean]; The content is parsed through current parseFunc
		    stdWrap = ->stdWrap processing of the content in the cell
		    tagStdWrap = -> The <TD> tag is processed by ->stdWrap
		  }
		  HTMLtableCells.addChr10BetweenParagraphs = [boolean]; If set, then all </P><P> appearances will have a chr(10) inserted between them
		}
		
		Example:
		This example is used to split regular bodytext content so that tables and blockquotes in the bodytext are processed correctly. The blockquotes are passed into parseFunc again (recursively) and further their top/bottom margins are set to 0 (so no apparent linebreaks are seen)
		The tables are also displayed with a number of properties of the cells overridden.
		<pre>tt_content.text.20.parseFunc.externalBlocks {
		  blockquote.callRecursive=1
		  blockquote.callRecursive.tagStdWrap.HTMLparser = 1
		  blockquote.callRecursive.tagStdWrap.HTMLparser {
		    tags.blockquote.fixAttrib.style.list = margin-
		bottom:0;margin-top:0;
		    tags.blockquote.fixAttrib.style.always=1
		  }
		  blockquote.stripNLprev=1
		  blockquote.stripNLnext=1
		
		  table.stripNL=1
		  table.stdWrap.HTMLparser = 1
		  table.stdWrap.HTMLparser {
		    tags.table.overrideAttribs = border=0 cellpadding=2
		cellspacing=1 style="margin-top:10px; margin-
		bottom:10px;"
		    tags.tr.allowedAttribs=0
		    tags.td.overrideAttribs = valign=top
		bgcolor="#eeeeee" style="font-family : Verdana, Geneva,
		Arial, Helvetica, sans-serif; font-size :
		10px;"
		  }
		}</pre>	

constants	boolean	The toplevel-defined constants will be substituted in the text. The constant-name is wrapped in "###". **Example:** constants.EMAIL = *email@email.com* (NOTE: This is toplevel TypoScript!) All cases of the string ###EMAIL### will be substituted in the text. The constants are defined as a toplevel object.	

Property:	Data type:	Description:	Default:

short	*array of strings*	Like constants above, but local.
		
		Example:
		This substitutes all occurencies of "T3" with "Typo3 CMS" and "T3web" with a link to typo3.com.
		`short {`
		`  T3 = Typo3 CMS`
		`  T3web = <a href="http://typo3.com">typo3</a>`
		`}`	

plainTextStdWrap	->stdWrap	This is stdWrap properties for all non-tag content.	
userFunc	function name	This passes the non-tag content to a function of your own choice. Similar to eg. .postUserFunc in stdWrap. Remember the function name must possibly be prepended "user_"	
nonTypoTagStdWrap	->stdWrap	Like .plainTextStdWrap. Difference: .plainTextStdWrap works an ALL non-tag pieces in the text. .nonTypoTagStdWrap is post processing of all text (including tags) between special TypoTags (unless .breakoutTypoTagContent is not set for the TypoTag)	
nonTypoTagUserFunc	function name	Like .userFunc. Differences is (like nonTypoTagStdWrap) that this is post processing of all content pieces around TypoTags while .userFunc processes all non-tag content. (Notice: .breakoutTypoTagContent must be set for the TypoTag if it's excluded from nonTypoTagContent)	
sword	wrap	Marks up any words from the GET-method send array sword_list[] in the text. The word MUST be at least two characters long! **NOTE:** works only with $GLOBALS["TSFE"]->no_cache==1	`<font color="red">|</font>`
makelinks	boolean / ->makelinks	Convert webadresses prefixed with "http://" and mail-adresses prefixed with "mailto:" to links.	
tags	->*tags*	Here you can define **custom tags** that will parse the content to something.	
allowTags	list of strings	List of tags, which are allowed to exist in code! Highest priority: If a tag is found in allowTags, denyTags is ignored!!	

denyTags	list of strings	List of tags, which may NOT exist in code! (use "*" for all.)
		Lowest priority: If a tag is NOT found in allowTags, denyTags is checked. If denyTags is not "*" and the tag is not found in the list, the tag may exist!
		
		Example:
		This allows <B>, <I>, <A> and <IMG> -tags to exist
		`.allowTags = b,i,a,img`
		`.denyTags = *`	

if	->if	if "if" returns false the input value is not parsed, but returned directly.	

[tsref:->parseFunc]

makelinks:

makelinks substitutes all appearances of

> http://www.webaddress.rld

> mailto:name@email.rld

... to a real linktag

Property:	Data type:	Description:	Default:
http.extTarget	target	The target of the link	_top
http.wrap	wrap	wrap around the link	
http.ATagBeforeWrap	boolean	If set, the link is first wrapped with *http.wrap* and then the <A>-tag.	
http.keep	list: *"scheme","path", "query"*	As default the link-text will be the full domain-name of the link. **Examples**: http://www.webaddress.rld/test/doc.php?id=3 "": www.webaddress.rld "scheme": http://www.webaddress.rld "scheme,path": http://www.webaddress.rld/test/doc.php "scheme,path,query": http://www.webaddress.rld/test/doc.php?id=3	
http.ATagParams	<A>-params	Additional parameters **Example:** class="board"	
mailto.wrap	wrap	wrap around the link	
mailto.ATagBeforeWrap	boolean	If set, the link is first wrapped with mailto.*wrap* and then the <A>-tag.	

Property:	Data type:	Description:	Default:
mailto.ATagParams	`<A>-params`	Additional parameters **Example:** class="board"	

[tsref:->makelinks]

tags:

Used to create custom tags and define how they should be parsed. This is used in conjunction with *parseFunc*.

Property:	Data type:	Description:	Default:
Array...	cObject +*stripNL* + *breakoutTypoTagContent*	Every entry in the *Array...* corresponds to a tag, that will be parsed. The elements MUST be in lowercase. Every entry must be set to a content-object. "current" is set to the content of the tag, eg <TAG>content</TAG>: here "current" is set to "content". **Parameters:** Parameters of the tag is set in $cObj->parameters (key is lowercased): <TAG COLOR="red">content</TAG> => $cObj->parameters[color] = red **Special added properties to the content-object:** $cObj->parameters[allParams]: this is automatically set to the whole parameter-string of the tag, eg ' color="red"' [cObject].stripNL: is a boolean option, which tells *parseFunc* that NewLines before and after content of the tag should be stripped. [cObject].breakoutTypoTagContent: is a boolean option, which tells parseFunc that this block of content is breaking up the nonTypoTag content and that the content after this must be re-wrapped. **Examples:** `tags.bold = TEXT` `tags.bold {` `  current = 1` ` wrap = <B>	</B>` `}` `tags.bold.stripNL = 1`	

[tsref:->tags]

Example:

This example creates 4 custom tags. The <LINK>-, <TYPOLIST>-, <GRAFIX>- and <PIC>-tags
<LINK> is made into a typolink and provides an easy way of creating links in text
<TYPOLIST> is used to create bullet-lists
<GRAFIX> will create a gif-file 90x10 pixels where the text is the content of the tag.
<PIC> lets us place an image in the text. The content of the tag should be the image-reference in "fileadmin/"

```
tags {
  link = TEXT
  link {
    current = 1
    typolink.extTarget = _blank
    typolink.target={$cLinkTagTarget}
    typolink.wrap = <B><FONT color=red>|</FONT></B>
    typolink.parameter.data = parameters : allParams
  }

  typolist < tt_content.bullets.default.20
  typolist.trim = 1
  typolist.field >
  typolist.current = 1

  grafix = IMAGE
  grafix {
    file = GIFBUILDER
    file {
      XY = 90,10
      100 = TEXT
      100.text.current = 1
      100.offset = 5,10
      100.nicetext = 1
    }
  }
  pic = IMAGE
  pic.file.import = fileadmin/
  pic.file.import.current = 1
}
```


HTMLparser:

Property:	Data type:	Description:
allowTags	list of tags	Default allowed tags

Property:	Data type:	Description:
tags.[tagname]	boolean/->HTMLparser_tags	Either set this property to 0 or 1 to allow or deny the tag. If you enter ->HTMLparser_tags properties, those will automatically overrule this option, thus it's not needed then. [tagname] in lowercase.
localNesting	list of tags, must be among preserved tags	List of tags (among the already set tags), which will be forced to have the nesting-flag set to true
globalNesting	(ibid)	List of tags (among the already set tags), which will be forced to have the nesting-flag set to "global"
rmTagIfNoAttrib	(ibid)	List of tags (among the already set tags), which will be forced to have the rmTagIfNoAttrib set to true
noAttrib	(ibid)	List of tags (among the already set tags), which will be forced to have the allowedAttribs value set to zero (which means, all attributes will be removed.
removeTags	(ibid)	List of tags (among the already set tags), which will be configured so they are surely removed.
keepNonMatchedTags	boolean / "protect"	If set (true=1), then all tags are kept regardless of tags present as keys in $tags-array. If "protect", then the preserved tags have their <> converted to < and > Default is to REMOVE all tags, which are not specifically assigned to be allowed! So you might probably want to set this value!
htmlSpecialChars	-1 / 0 / 1 / 2	This regards all content which is NOT tags: "0" means "disabled" - nothing is done "1" means the content outside tags is htmlspecialchar()'ed (PHP-function which converts &"<> to &...;) "2" is the same as "1" but entities like "&" or "&#234" are untouched. "-1" does the opposite of "1" - converts < to <, > to >, " to " etc.
xhtml_cleaning	boolean	Cleans up the content for XHTML compliance. Still slightly experimental and supports only some clean up operations (like convertion tags and attributes to lower case).

[page:->HTMLparser; tsref:->HTMLparser]

HTMLparser_tags:

Property:	Data type:	Description:
overrideAttribs	string	If set, this string is preset as the attributes of the tag.
allowedAttribs		'0' (zero) = no attributes allowed, '[commalist of attributes]' = only allowed attributes. If blank/not set, all attributes are allowed.
fixAttrib.[attribute].set	string	Force the attribute value to this value.
fixAttrib.[attribute].unset	boolean	If set, the attribute is unset.
fixAttrib.[attribute].default	string	If no attribute exists by this name, this value is set as default value (if this value is not blank)
fixAttrib.[attribute].always	boolean	If set, the attribute is always processed. Normally an attribute is processed only if it exists
fixAttrib.[attribute].trim fixAttrib.[attribute].intval fixAttrib.[attribute].upper fixAttrib.[attribute].lower	boolean	If any of these keys are set, the value is passed through the respective PHP-functions.
fixAttrib.[attribute].range	[low],[high]	Setting integer range.
fixAttrib.[attribute].list	list of values, trimmed	Attribute value must be in this list. If not, the value is set to the first element.
fixAttrib.[attribute].removeIfFalse	boolean/"blank" string	If set, then the attribute is removed if it is "false". If this value is set to "blank" then the value must be a blank string (that means a "zero" value will not be removed)
fixAttrib.[attribute].removeIfEquals	string	If the attribute value matches the value set here, then it is removed.
fixAttrib.[attribute].casesensitiveComp	boolean	If set, the comparison in .removeIfEquals and .list will be case-sensitive. At this point, it's insensitive.
fixAttrib.[attribute].prefixLocalAnchors	integer	If the first char is a "#" character (anchor of fx. <a> tags) this will prefix either a relative of absolute path. If the value is "1" you will get the absolute path (t3lib_div::getIndpEnv('TYPO3_REQUEST_URL')) If the value is "2" you will get the relative path (stripping of t3lib_div::getIndpEnv('TYPO3_SITE_URL')) **Example:** ...fixAttrib.href.prefixLocalAnchors = 1

Property:	Data type:	Description:
fixAttrib.[attribute].prefixRelPathWith	string	If the value of the attribute seems to be a relative URL (no scheme like "http" and no "/" as first char) then that value of this property will be prefixed the attribute. **Example:** ...fixAttrib.src.prefixRelPathWith = http://192.168.230.3/typo3/32/dummy/
fixAttrib.[attribute].userFunc	function reference	User function for processing of the attribute. **Example:** ...fixAttrib.href.userFunc = tx_realurl->test_urlProc
protect	boolean	If set, the tag <> is converted to < and >
remap	string	If set, the tagname is remapped to this tagname
rmTagIfNoAttrib	boolean	If set, then the tag is removed if no attributes happend to be there.
nesting		If set true, then this tag must have starting and ending tags in the correct order. Any tags not in this order will be discarded. Thus '</B><B><I></B></I></B>' will be converted to '<B><I></B></I>'. Is the value "global" then true nesting in relation to other tags marked for "global" nesting control is preserved. This means that if <B> and <I> are set for global nesting then this string '</B><B><I></B></I></B>' is converted to '<B></B>'

[page:->HTMLparser_tags; tsref:->HTMLparser_tags]

Constants

What are constants?

Constants are values defined in the "Constants"-field of a template. They follow the syntax of ordinary TypoScript!

NOTE, reserved name: The object or property "file" is always interpreted as data type "resource".

NOTE: Toplevel "object" TSConstantEditor cannot be used. It's reserved for configuration of the ConstantEditor module (Changed from beta4)
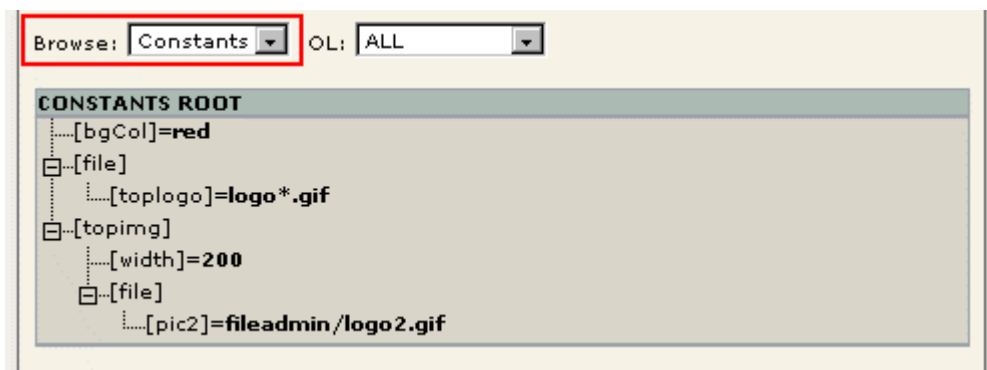
Example:

Here "bgCol" is set to "red" and "file.toplogo" is set to "logo*.gif" which is found in the resource-field of the template.

```
bgCol = red
topimg.width = 200
topimg.file.pic2 = fileadmin/logo2.gif
file.toplogo = logo*.gif
```

This could also be defined like this:

```
bgCol = red
file {
  toplogo = logo*.gif
}
topimg {
  width = 200
  file.pic2 = fileadmin/logo2.gif
}
```

(The objects in bold is the reserved word "file" and the properties are always of data type "resource"

Inserting constants

Constants are inserted in the template-setup by performing an ordinary str_replace operation! You insert them like this:

```
{$bgCol}
{$topimg.width}
{$topimg.file.pic2}
{$file.toplogo}
```

Example:

```
page = PAGE
page.typeNum = 0

page.bodyTag = <body bgColor="{$bgCol}">
page.10 = IMAGE
page.10.file = {$file.toplogo}
```

Only defined constants are substituted.

Constants in included templates are also substituted as the whole template is just on large chunk of text.

Constants are case sensitive.

You should use a systematic naming of constants. Seek inspiration in the code-examples around.

(default)

test ▾ ▶ [r

Include static AFTER basedOn: ☐

TEMPLATE HIERARCHY:

Title   Roo

- plugin.meta
  - styles.hmenu.tu
    - styles.content (d...
    - styles.sitemap.text
    - styles.header.gfx1
  - content (default)
- template; TU
- plugin.meta
- content.postit1
- cSet (default)
- Example template
- test

---

test ▾ ▶ [m

Include static AFTER basedOn: ☑

TEMPLATE HIERARCHY:

Title   Roo

- styles.hmenu.tu
  - styles.content (d...
  - styles.sitemap.text
  - styles.header.gfx1
- content (default)
- template; TU
- plugin.meta
- content.postit1
- cSet (default)
- Example template
- plugin.meta
- test

(default)

test ▾ ▶ [r

Include static AFTER basedOn: ☐

TEMPLATE HIERARCHY:

Title   Roo

- plugin.meta
  - styles.hmenu.tu
    - styles.content (d...
    - styles.sitemap.text
    - styles.header.gfx1
  - content (default)
- template; TU
- plugin.meta
- content.postit1
- cSet (default)
- Example template
- test

---

test ▾ ▶ [m

Include static AFTER basedOn: ☑

TEMPLATE HIERARCHY:

Title   Roo

- styles.hmenu.tu
  - styles.content (d...
  - styles.sitemap.text
  - styles.header.gfx1
- content (default)
- template; TU
- plugin.meta
- content.postit1
- cSet (default)
- Example template
- plugin.meta
- test

Notice how the constants in the setup code is substituted. In the Object Browser, you can monitor the constants with or without substitution. Also notice that the value "logo*.gif" was resolved to the resource "uploads/tf/logo_01.gif"

(Note: The "Constants display" function is not available if you select "Crop lines")

Setup:

Toplevel objects:

Property:	Data type:	Description:	Default:
types	readonly	Types (internal) type=99 reserved for plaintext display	
resources	readonly	Resources in list (internal)	
sitetitle	readonly	SiteTitle (internal)	
config	->CONFIG	Global configuration. These values are stored with cached pages which means they are also accessible when retrieving a cached page.	
constants	->CONSTANTS	Site-specific constants, eg. a general email-adresse. These constants may be substituted in the text throughout the pages. The substitution is done by parseFunc. (Option: constants=1)	
FEData	->*FE_DATA*	Here you can configure how data submitted from the front-end should be processed, which script and so on.	
includeLibs	*array of strings*	With this you can include php-files with function libraries for use in your includescript in Typo3. Please see the PAGE-object, which has the same property.	
Other reserved TLO's: plugin tt_* temp styles lib _GIFBUILDER		These toplevel object names are reserved. That means you can risk static_templates to use them: "plugin" is used for rendering of special content like boards, ecommerce solutions, guestbooks and so on. Normally set from static_templates. *Please see separate description below!* "tt_*", eg tt_content (from "content (default)") is used to render content from tables. "temp" and "styles" are used for conde-libraries you can copy during parse-time, but they are not saved with the template in cache. **"temp" / "styles" are unset** before the template is cached! Therefore use these names to store temporary data. "lib" can be used for a "library" of code, you can reference in TypoScript (unlike "styles" which is unset)	

...	PAGE	Start a new page
		
		Example:
		`page = PAGE`
		`page.typeNum = 1`
		
		Guidelines:
		Good, general PAGE object names to use are such as:
		page for the main page with content
		frameset, frameset2 for framesets.
		top, left, menu, right, bottom, border for top and menu frames etc.
		This is just recommandations. Especially the name 'page' for the content bearing page is very common.	

...	*(whatever)*	If a toplevel-object is not a PAGE-object it could be used as a temporary repository for setup. In this case you should use the "temp" or "styles" objects. "tt_..." is normally used to define the setup of content-records. Eg. "tt_content" would be used for the tt_content-table as default. See the "CONTENT"-cObject	

[tsref:(TLO)]

The "plugin" TLO:

This is used for extensions in Typo3 set up as frontend plugins. Typically you can set configuration properties of the plugin here. Say you have an extension with the key "tx_myext" and it has a frontend plugin named "tx_myext_pi1" then you would find the TypoScript configuration at the position "plugin.tx_myext_pi1" in the object tree!

Most plugins are USER or USER_INT objects which means that they have at least 1 or 2 reserved properties. Furthermore this table outlines some other default properties. Generally system properties are prefixed with an underscore:

Property:	Data type:	Description:	Default:
userFunc		*Property setting up the USER / USER_INT object of the plugin*	
includeLibs		*Property setting up the USER / USER_INT object of the plugin*	

Property:	Data type:	Description:	Default:
_CSS_DEFAULT_STYLE	string	Use this to have some default CSS styles inserted in the header section of the document. Most likely this will provide a default acceptable display from the plugin, but should ideally be cleared and moved to an external stylesheet. This value is for all plugins read by the pagegen script when making the header of the document.	
_DEFAULT_PI_VARS.[piVar-key]	string	Allows you to set default values of the piVars array which most plugins are using (and should use) for data exchange with themselves. This works only if the plugin calls $this->pi_setPiVarDefaults().	
_LOCAL_LANG.[lang-key].[label-key]	String	Can be used to override the default locallang labels for the plugin.	

[tsref:plugin]

"CONFIG":

In tslib/ this is known as $GLOBALS["TSFE"]->config["config"], thus the property "debug" below is accessible as $GLOBALS["TSFE"]->config["config"]["debug"].

Property:	Data type:	Description:	Default:
linkVars	list	HTTP_GET_VARS, which should be passed on with links in Typo. This is compiled into a string stored in $GLOBALS["TSFE"]->linkVars **Example:** `config.linkVars = noframes, print` This will add "&noframes=[noframes-value]&print=[print-value]" to all links in Typo3. The values are rawurlencoded in PHP.	
MP_defaults	string	Allows you to set a list of page id numbers which will always have a certain "&MP=..." parameter added. **Syntax:** [id],[id],... : [MP-var] | [id],[id],... : [MP-var] | ... **Example:** `config.MP_defaults = 36,37,48 : 2-207` This will by default add "&MP=2-207" to all links pointing to pages 36,37 and 48	
MP_mapRootPoints	list of PIDs/string	Defines a list of ID numbers from which the MP-vars are automatically calculated for the branch. The result is used just like MP_defaults are used to find MP-vars if none has been specified prior to the call to t3lib_tstemplate::linkData(). You can specify "root" as a special keyword in the list of IDs and that will create a map-tree for the whole site (but this may be VERY processing intensive if there are many pages!). The order of IDs specified may have a significance; Any ID in a branch which is processed already (by a previous ID root point) will not be processed again.	
MP_disableTypolinkClosestMPvalue	boolean	If set, the typolink function will not try to find the closest MP value for the id.	
renderCharset	string	Charset used for rendering internally of the page content. It is highly recommended that this value is the same as the charset of the content coming from the main data source (eg. the database). Thus you don't need to do any other conversion. All strings from locallang files and locale strings are (and should be) converted to "renderCharset" during rendering. If you need another output charset than the render charset, see "metaCharset" below. If you set TYPO3_CONF_VARS['BE']['forceCharset'] that value is used by default for "renderCharset". It is highly recommended to use TYPO3_CONF_VARS['BE']['forceCharset'] for multilingual websites in TYPO3. If you set that you don't have to worry about renderCharset and metaCharset - the same charset is used in the whole system.	TYPO3_CONF_VARS[BE][forceCharset] if found, otherwise "iso-8859-1"

Property:	Data type:	Description:	Default:
metaCharset	string	Charset used for the output document. For example in the meta tag: `<meta http-equiv="Content-Type" content="text/html; charset=...>` Is used for a) HTML meta-tag, b) HTTP header (unless disabled with .disableCharsetHeader) and c) xhtml prologues (if available) If renderCharset and metaCharset are different the output content is automatically converted to metaCharset before output and likewise are values posted back to the page converted from metaCharset to renderCharset for internal processing. This conversion takes time of course so there is another good reason to use the same charset for both.	value of ".renderCharset"
disableCharsetHeader	boolean	By default a header "content-type:text/html; charset..." is sent. This option will disable that.	

doctype	string	If "**xhtml_frames**" then a header for XHTML-frames is outputted.
		If value is "**xhtml_trans**" then a header for XHTML transitional is outputted:
		
		`<?xml version="1.0" encoding="[here goes config.metaCharset!]"?>`
		`<!DOCTYPE html`
		`    PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"`
		`    "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">`
		
		Keyword list:
		keyword "**xhtml_trans**" for XHTML 1.0 Transitional doctype.
		keyword "**xhtml_frames**" for XHTML 1.0 Frameset doctype.
		keyword "**xhtml_strict**" for XHTML 1.0 Strict doctype.
		keyword "**xhtml_11**" for XHTML 1.1 doctype.
		keyword "**xhtml_2**" for XHTML 2 doctype.
		keyword "**none**" for NO doctype at all.
		
		Any other value than these keywords will be outputted as the doctype itself!
		Notice that some doctype keywords (all the xhtml* sorts) will set appropriate attributes for the `<html>` tag as well. See "config.htmlTag_setParams" and "config.htmlTag_langKey" below for more details.
		
		Default is a DOCTYPE like this:
		`<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">`	

doctypeSwitch	boolean / string	If set, the order of `<?xml...>` and `<!DOCTYPE...>` will be reversed. This is needed for MSIE to be standards compliant with XHTML.
		
		Background:
		By default TYPO3 outputs the XML/DOCTYPE in compliance with the standards of XHTML. However a browser like MSIE will still run in "quirks-mode" unless the `<?xml>` and `<DOCTYPE>` tags are ordered opposite. But this breaks CSS validation...
		With this option designers can decide for themselves what they want then.
		
		If you want to check the compatibility-mode of your webbrowser you can do so with a simple JavaScript that can be inserted on a TYPO3 page like this:
		
		`page.headerData.1 = TEXT`
		`page.headerData.1.value = <script>alert`
		`(document.compatMode);</script>`
		
		If your browser has detected the DOCTYPE correctly it will report "CSS1Compat"
		If you are not running in compliance mode you will get some other message. MSIE will report "BackCompat" for instance - this means it runs in quirks-mode, supporting all the old "browser-bugs".	

xmlprologue	string	If "none" then no XML prologue is created when "config.doctype" is set to one of the XHTML modes.	
htmlTag_setParams	string	Sets the attributes for the `<html>` tag on the page. If you set "config.doctype" to a keyword enabling XHTML then some attributes are already set. This property allows you to override any preset attributes with you own content if needed. **Special:** If you set it to "none" then no attributes will be set at any event. **Example:** `config.htmlTag_setParams = xmlns="http://www.w3.org/1999/xhtml" xml:lang="en-US"`	

Property:	Data type:	Description:	Default:
htmlTag_langKey	string	Allows you to set the language value for the attributes "xml:lang" and "lang" in the <html> tag (when using "config.doctype = xhtml*"). The values must follow the format specified in IETF RFC 3066 **Example:** config.htmlTag_langKey = en-US	en
htmlTag_dir	string	Sets text direction for whole document (useful for display of Arabic, Hebrew pages). Basically the value becomes the attribute value of "dir" for the <html> tag. **Values:** rtl = Right-To-Left (for Arabic / Hebrew) ltr = Left-To-Right (Default for other languages) **Example:** config.htmlTag_dir = rtl	
ATagParams	<A>-params	Additional parameters to all links in Typo3 (excluding menu-links) **Example:** To blur links, insert: `onFocus="blurLink(this)"`	
setJS_openPic	boolean	If set, the openPic JavaScript function is forced to be included	
setJS_mouseOver	boolean	If set, the over() and out() JavaScript functions are forced to be included	
removeDefaultJS	boolean / string	if set, the default JavaScript in the header will be removed. The default JavaScript is the blurLink function and browser detection variables. **Special case:** if the value is "**external**" then the default JavaScript is written to a temporary file and included from that file. See "inlineStyle2TempFile" below. **Example:** config.removeDefaultJS = external config.removeDefaultJS = 1	
inlineStyle2TempFile	boolean	If set, the inline styles TYPO3 controls in the core are written to a file, typo3temp/stylesheet_[hashstring].css, and the header will only contain the link to the stylesheet. The file hash is based solely on the content of the styles. **Example:** config.inlineStyle2TempFile = 1	
ftu	boolean	If set, the "&ftu=...." GET-fallback identification is inserted. "&ftu=[hash]" is always inserted in the links on the first page a user hits. If it turns out in the next hit that the user has cookies enabled, this variable is not set anymore as the cookies does the job. If no cookies is accepted the "ftu" remains set for all links on the site and thereby we can still track the user. **You should not set this feature if grabber-spiders like Teleport are going to grab your site!** **You should not set this feature if you want search-engines to index your site (in conjunction with the simulateStaticDocuments feature!)** You can also ignore this feature if you're certain, website users will use cookies. "ftu" means fe_typo_user ("fe" is "frontend").	false
mainScript	string	This lets you specify an alternative "mainScript" which is the document that Typo3 expects to be the default doc. This is used in form-tags and other places where Typo3 needs to refer directly to the main-script of the application	index.php

Property:	Data type:	Description:	Default:
pageGenScript	resource	Alternative page generation script for applications using index_ts.php for initialization, caching, stating and so on. This script is included in the global scope of index_ts.php-script and thus you may include libraries here. Always use include_once for libraries. Remember not to output anything from such an included script. **All content must be set into $TSFE->content.** Take a look at tslib/pagegen.php **NOTE:** This option is ignored if `$TYPO3_CONF_VARS["FE"]["noPHPscriptInclude"]=1;` is set in localconf.php.	tslib/pagegen.php
debug	boolean	If set any debug-information in the TypoScript code is output. Currently this applies only to the menu-objects	
message_page_is_being_generated	string	Alternative message in HTML that appears if a page is being generated. Normally when a page is being generated a temporary copy is stored in the cache-table with an expire-time of 30 seconds.	
message_preview	string	Alternative message in HTML that appears when the preview function is active!	
locale_all	string	PHP: setlocale("LC_ALL", [value]); value-examples: deutsch, de_DE, danish, portuguese, spanish, french, norwegian, italian. See www.php.net for other value. Also on linux, look at /usr/share/locale/ TSFE->localeCharset is intelligently set to the assumed charset of the locale strings. This is used in stdWrap.strftime to convert locale strings to the renderCharset of the frontend. **Example:** This will render dates in danish made with stdWrap/strftime: locale_all = danish locale_all = da_DK	
sword_standAlone	boolean	Used by the parseFunc-substitution of search Words (sword): If set, the words MUST be surrounded by whitespace in order to be marked up.	
sword_noMixedCase	boolean	Used by the parseFunc-substitution of search Words (sword): If set, the words MUST be the exact same case as the search word was.	
intTarget	target	default internal target. Used by typolink if no target is set	
extTarget	target	default external target. Used by typolink if no extTarget is set	_top
spamProtectEmailAddresses	boolean / -5 to 5	If set, then all email addresses in typolinks will be encrypted so spam bots cannot detect them. Encryption is a simply offset of character values. If you set this value to "3" then all characters will have their ASCII value offset by "3". The reason for supplying this is... well just a way you can choose another offset than everyone else... Hmm. Does it make sense?	
spamProtectEmailAddresses _atSubst	string	Substitute label for the at-sign (@).	(at)
forceTypeValue	int	Force the &type value of all Typo3 generated links to a specific value (except if overruled by local forceTypeValue values). Useful if you run a template with special content at - say &type=95 - but still wants to keep your targets neutral. Then you set your targets to blank and this value to the type value you wish.	
frameReloadIfNotInFrameset	boolean	If set, then the current page will check if the page object name (eg. "page" or "frameset") exists as "parent.[name]" (eg. "parent.page") and if not the page will be reloaded in top frame. This secures that links from search engines to pages inside a frameset will load the frameset. Works only with type-values different from zero.	
jumpurl_enable	boolean	Jumpurl is a concept where external links are redirected from the index_ts.php script, which first logs which url it was. This logging of external links is only interesting if you use the internal stat-table in Typo3.	0
jumpurl_mailto_disable	boolean	Disables the use of jumpUrl when linking to email-adresses.	0

Property:	Data type:	Description:	Default:

compensateFieldWidth	double	this floating point value will be used by the FORMS cObject to compensate the length of the formfields text and input.
		This feature is useful, if the page-option "smallFormFields" is set. In that case Netscape renders formfields much longer than IE. If you want the two browsers to display the same size formfields, use a value of approx "0.6" for netscape-browsers.
		
		Example:
		`[browser = netscape]`
		`  config.compensateFieldWidth = 0.6`
		`[global]`
		
		This option may be overridden in the FORMS-cObject.	

includeLibrary	resource	This includes a phpfile.	
incT3Lib_htmlmail	boolean	Include t3lib/class.t3lib_htmlmail.php	
lockFilePath	string	This is used to lock paths to be "inside" this path. Used by "filelist" in stdWrap	fileadmin/
noScaleUp	boolean	Normally images are scaled to the size specified via TypoScript. This also forces small images to be scaled to a larger size. This is not always a good thing. If this property is set, images are **not** allowed to be scaled up in size. This parameter clears the $this->mayScaleUp var of the class t3lib_stdgraphics (often "gifbuilder").	
USERNAME_substToken	string	The is the token used on the page, which should be substituted with the current username IF a front-end user is logged in! If no login, the substitution will not happen.	<!--###USERNAME###-->
cache_period	int, seconds	The number of second a page may remain in cache. This value is overridden by the value set in the page-record (field="cache_timeout") if this value is greater than zero.	86400 (=24H)
cache_clearAtMidnight	boolean	With this setting the cache always expires at midnight of the day, the page is scheduled to expire.	false
no_cache	boolean	If this is set to true, the page will not be cached. If set to false, it's ignored. Other parameters may have set it to true of other reasons.	-
disableAllHeaderCode	boolean	If this is set, none of the features of the PAGE-object is processed and the content of the page will be the result of the cObject array (1,2,3,4...) of the PAGE-object. This means that the result of the cObject should include everything from the <HTML> .... to the </HTML> tag !! Use this feature in templates supplying other content-types than HTML. That could be an image or a WAP-page!	false
additionalHeaders	strings divided by "|"	This is additional headers. You separate each header by a vertical line "|". Normally Typo3 does *not* send any headers with the Header()-function in PHP. **Examples:** Content-type: text/vnd.wap.wml (this will sent a content-header for a WAP-site) Content-type: image/gif | Expires: Mon, 26 Jul 1997 05:00:00 GMT (this will sent a content-header for a GIF-file and a Expires header) Location: www.typo3.com (This redirects the page to www.typo3.com)	
disablePageExternalUrl	boolean	If set, pages with doktype "External Url" will not trigger jumpUrl in TSFE. This may help you to have external urls open inside you framesets.	
stat	boolean	Enable stat logging at all.	true
stat_typeNumList	int/list	List of pagetypes that should be registered in the statistics table, sys_stat. If no types are listed, all types are logged. Default is "0,1" which normally logs all hits on framesets and hits on content keeping pages. Of course this depends on the template design.	0,1
stat_excludeBEuserHits	boolean	If set a pagehit is not logged if a user is logged in into Typo3.	false
stat_excludeIPList	list of strings	If the REMOTE_ADDR is in the list of IP-addresses, it's also not logget.	
stat_mysql	boolean	Enable logging to the MySQL table sys_stat.	false
stat_apache	boolean	Enable logging to the logfile "stat_apache_logfile"	false
stat_apache_logfile	filename	This the filename of the logfile in the directory $TYPO3_CONF_VARS ["FE"]["logfile_dir"] (localconf) on the server where Typo3 writes to the apachelogfile. The directory defined by $TYPO3_CONF_VARS["FE"] ["logfile_dir"] must exist and be writable. The logfile also must exist.	

Property:	Data type:	Description:	Default:
stat_apache_pagenames	string	The "pagename" simulated for apache. Default: "[path][title]--[uid].html" Codes: [title] = inserts title, no special characters and shortend to 30 chars. [uid] = the id [alias] = any alias [type] = the type (typeNum) [path] = the path of the page.	
stat_apache_notExtended	boolean	If true the logfile is NOT written in Apache extended format	
stat_apache_noHost	boolean	If true the HTTP_HOST is - if available - NOT inserted instead of the IP-address	
stat_titleLen	int 1-100	The lenght of the page names in the path written to logfile/database	20

simulateStaticDocuments	boolean / string: "PATH_INFO"	If set Typo3 makes all links in another way than usual. This can be used with **Apache compiled with mod_rewrite and configured in httpd.conf for use of this in the ".htaccess"-files.**
		Include this in the .htaccess file
		     RewriteEngine On
		     RewriteRule ^[^/]*\.html$ index.php
		
		This means that any "*.html"-documents should be handled by index.php. Now if is done, Typo3 will interpret the url of the html-document like this:
		     [title].[id].[type].html
		Title is optional and only usefull for the entries in the apache log-files. You may omit both [title] and [type] but if title is present, type must also be there!.
		
		Example:
		Typo3 will interpret this as page with uid=23 and type=1 :
		     Startpage.23.1.html
		
		Typo3 will interpret this as the page with alias = "start" and the type is zero (default):
		     start.html
		_____
		Alternative option (PATH_INFO):
		Instead of using the rewrite-module in apache (eg. if you're running Windows!) you can use the PATH_INFO variable from PHP.
		It's very simple. Just set simulateStaticDocuments to "PATH_INFO" and you're up and running!
		
		Also: See below, .absRefPrefix
		
		Example (put in Setup-field of your template):
		config.simulateStaticDocuments = PATH_INFO
			default is defined by a configuration option in localconf.php. It's $TYPO3_CONF_VARS["FE"]["simulateStaticDocuments"] = 1; This affects all sites in the database.
			You can also set this value to the string "PATH_INFO"

simulateStaticDocuments_addTitle	int	If not zero, Typo3 generates urls with the title in, limited to the first [simulateStaticDocuments_addTitle] number of chars. **Example:** Startpage.23.1.html instead of the default, "23.1.html", without the title.	
simulateStaticDocuments_noTypeIfNoTitle	boolean	If set, then the type-value will not be set in the simulated filename if the type value is zero anyways. However the filename must be without a title. **Example:** "Startpage.23.0.html" would _still_ be "Startpage.23.0.html" "23.0.html" would be "23.html" (that is without the zero) "23.1.html" would _still_ be "23.1.html"	
simulateStaticDocuments_dontRedirectPathInfoError	boolean	Regarding PATH_INFO mode: When a page is requested by "PATH_INFO" method it must be configured in order to work properly. If PATH_INFO is not configured, the index_ts.php script sends a location header to the correct page. However if you better like an error message outputted, just set this option.	

Property:	Data type:	Description:	Default:
simulateStaticDocuments_pEnc	string	Allows you to also encode additional parameters into the simulated filename. **Example:** You have a news-plugin. The main page has the url "Page_1.228.0.html" but when one clicks on a news item the url will be "Page_1.228.0.html?&tx_mininews_pi1[showUid]=2&cHash=b8d239c224" instead. Now, this URL will not be indexed by external search-engines because of the query-string (everything after the "?" mark). This property avoids this problem by encoding the parameters. These are the options: **Value set to "base64":** This will transform the filename used to this value: "Page_1.228+B6JnR4X21pbmluZXdzX3BpMVtzaG93VWlkXT0yJmNIYXNoPWI4ZDIzOWMyMjQ_.0.html". The querystring has simply been base64-encoded (and some more...) and added to the HTML-filename (so now external search-engines will find this!). The really great thing about this that the filename is self-reliant because the filename contains the parameters. The downside to it is the very very long filename. **Value set to "md5":** This will transform the filename used to this value: "Page_1.228+M57867201f4a.0.html". Now, what a lovely, short filename! Now all the parameters has been hashed into a 10-char string inserted into the filename. At the same time an entry has been added to a cache table in the database so when a request for this filename reaches the frontend, then the REAL parameter string is found in the database! The really great thing about this is that the filename is very short (opposite to the base64-method). The downside to this is that IF you clear the database cache table at any time, the URL here does NOT work until a page with the link has been generated again (re-inserting the parameter list into the database). **NOTICE:** From TYPO3 3.6.0 the encoding will work only on parameters that are manually entered in the list set by . simulateStaticDocuments_pEnc_onlyP (see right below) or those parameters that various plugins might allow in addition. This is to limit the run-away risk when many parameters gets combined.	
simulateStaticDocuments_pEnc_onlyP	string	A list of variables that may be a part of the md5/base64 encoded part of a simulate_static_document virtual filename (see property in the row above). **Example:** simulateStaticDocuments_pEnc_onlyP = tx_maillisttofaq_pi1[pointer], L, print -> this will allow the "pointer" parameter for the extension "maillisttofaq" to be included (in addition to whatever vars the extension sets itself) and further the parameter "L" (could be language selection) and "print" (could be print-version).	
content_from_pid_allowOutsideDomain	boolean	Using the "Show content from this page instead" feature allows you to insert content from the current domain only. Setting this option will allow content included from anywhere in the page tree!	
absRefPrefix	string	If this value is set, then all relative links in TypoScript are prepended with this string. Used to convert relative paths to absolute paths. **Note:** This values is automatically set to the dirname of the index.php script in case simulateStaticDocuments is set to "PATH_INFO". If you're working on a server where you have both internal and external access, you might to yourself a favour and set the absRefPrefix to the url and path of you site, eg. http://www.typo3.com/. If you do not, you risk to render pages to cache from the internal network and thereby prefix image-references and links with a non-accesible path from outside.	
noPageTitle	boolean	If set, the page title will not be printet as the <TITLE> of the doc.	
pageTitleFirst	boolean	If set (and the page title is printed) then the page-title will be printed BEFORE the template title.	
titleTagFunction	function-name	Passes the default <title>-tag content to this function. No typoScript parameters are passed though.	
headerComment	string	The content is added before the "Typo3 Content Management Framework" comment in the <head> section of the page. Use this to insert a note like that "Programmed by My-Agency" ...	

Property:	Data type:	Description:	Default:
language	string	Language key. See stdWrap.lang for more information. Select between: English (default) = [empty] Danish = dk German = de Norwegian = no Italian = it Value must correspond with the key used for backend system language if there is one. If not, use TLD (because if some day this language is added it will get that key)). See translation page on TYPO3.org for the official 2-byte key for a given language. Notice that selecting the official key is important if you want labels in the correct language from "locallang" files.	
sys_language_uid	int	This value points to the uid of a record from the "sys_language" table and if set, this means that various parts of the frontend display code will select records which are assigned to this language. See ->SELECT Internally, the value is depending on whether a Alternative Page Language record can be found with that language. If not, the value will default to zero (default language) except if "sys_language_mode" is set to a value like "content_fallback".	
sys_language_mode	string	Setting various modes of handling localization. The syntax is "[keyword] ; [value]". Possible keywords are: [default] - The system will look for a translation of the page (from "Alternative Page Language" table) and if it is not found it will fall back to the default language and display that. **content_fallback** - [Recommended] The system will always operate with the selected language even if the page is not translated with a page overlay record. This will keep menus etc. translated. The content on the page can still fall back to another language, defined by the value of this keyword, eg. "content_fallback ; 1,0" to fall back to the content of sys_language_uid 1 and if that is not present either, to default (0) **strict** - The system will report an error if the requested translation does not exist. Basically this means that all pages with gray background in the Web>Info / Localization overview module will fail (they would otherwise fall back to default language in one or another way)	
sys_language_overlay	boolean / keyword	If set, records from certain tables selected by the CONTENT cObject using the "languageField" setting will select the default language (0) instead of any language set by sys_language_uid / sys_language_mode. In addition the system will look for a translation of the selected record and overlay configured fields. The requirements for this is that the table is configured with "languageField" and "transOrigPointerField" in the [ctrl] section of $TCA. Also, exclusion of certain fields can be done with the "l10n_mode" directive in the field-configuration of $TCA. For backend administration this requires that you configure the "Web>Page" module to display content elements accordingly; That each default element is shown and next to it any translation found. This configuration can be done with Page TSconfig for a section of the website using the object path "mod.web_layout.defLangBinding = 1". Keyword: **hideNonTranslated** : If this keyword is used a record that has no translation will not be shown. The default is that records with no translation will show up in the default language.	

Property:	Data type:	Description:	Default:
sys_language_softMergeIfNotBlank	string	Setting additional "mergeIfNotBlank" fields from TypoScript. Background: In TCA you can configure "l10n_mode" - localization mode - for each field. Two of the options affect how the frontend displays content; The values "exclude" and "mergeIfNotBlank" (see "TYPO3 Core API" document for details). The first ("exclude") simply means that the field when found in a translation of another record will not be overlaid the default records field value. The second ("mergeIfNotBlank") means that it will be overlaid *only* if it has a non-blank value. Since it might be practical to set up fields for "mergeIfNotBlank" on a per-site basis this options allows you to override additional fields from tables. **Syntax:** [table]:[field], [table]:[field], [table]:[field], ... **Example:** config.sys_language_softMergeIfNotBlank = tt_content:image , tt_content:header This setting means that the header and image field of content elements will be used from the translation only if they had a non-blank value. For the image field this might be very practical because it means that the image(s) from the default translation will be used unless other images are inserted!	
typolinkCheckRootline	boolean	If set, then every "typolink" is checked whether it's linking to a page within the current rootline of the site. If not, then Typo3 searches for the first found domain record (without redirect) in that rootline from out to in. If found (another domain), then that domain is prepended the link, the external target is used instead and thus the link jumps to the page in the correct domain.	
insertDmailerBoundaries	boolean	If set, boundary marks will be set around all records inserted on the page with cObjects CONTENT and RECORD. They are inserted as HTML-comments and do no harm. Used by the Direct Mail module in Typo3 to segmentize a page by categories.	
notification_email_urlmode	string	This option allows you to handle URL's in plain text emails so long URLS of more than 76 chars are not broken. This option can be either empty or "76" or "all". If the string is blank, all links in plaintext emails are untouched. If it's set to 76 then all links longer then 76 characters are stored in the database and a hash is sent in the GET-var ?RDCT=[md5/20] to the index.php script which finds the proper link in the database and issues a location header (redirection). If the value is "all" then ALL "http://" links in the message are converted.	
notification_email_encoding	string	This sets the encoding of plaintext emails (notification messages). Normally no encoding is applied at all. But setting this to eg. "base64" will encode the content with base64 encoding (charset ISO-8859-1). **Values possible:** base64 quoted-printable 8bit	
notification_email_charset	string	Alternative charset for the notification mails. Is set only if an encoding type is entered.	ISO-8859-1
admPanel	boolean / ->ADMPANEL properties	If set, the admin panel appears in the bottom of pages. **NOTE:** In addition the panel must be enabled for the user as well, using the TSconfig for the user! See adminguide documentation. **SEE:** Admin Panel section	
beLoginLinkIPList	[IP-number]	If set and REMOTE_ADDR matches one of the listed IP-numbers (Wild-card, *, allowed) then a link to the typo3/ login scrip with redirect pointing back to the page is shown. **NOTE:** beLoginLinkIPList_login and/or beLoginLinkIPList_logout (see below) must be defined if the link should show up!	
beLoginLinkIPList_login	HTML	HTML code wrapped with the login link, see 'beLoginLinkIPList' **Example:** <HR><B>LOGING</B>	
beLoginLinkIPList_logout	HTML	HTML code wrapped with the logout link, see above	
index_enable	boolean	Enables cached pages to be indexed.	

Property:	Data type:	Description:	Default:
index_externals	boolean	If set, external media linked to on the pages is indexed as well.	
index_descrLgd	int	This indicates how many chars to preserve as description for an indexed page. This may be used in the search result display.	200
xhtml_cleaning	string	Tries to clean up the output to make it XHTML compliant and a bit more. THIS IS NOT COMPLETE YET, but a "pilot" to see if it makes sense anyways. For now this is what is done: **What it does at this point:** - All tags (img,br,hr) is ended with "/>" - others? - Lowercase for elements and attributes - All attributes in quotes - Add "alt" attribute to img-tags if it's not there already. **What it does NOT do (yet) according to XHTML specs.:** - Wellformedness: Nesting is NOT checked - name/id attribute issue is not observed at this point. - Certain nesting of elements not allowed. Most interesting, <PRE> cannot contain img, big,small,sub,sup ... - Wrapping scripts and style element contents in CDATA - or alternatively they should have entitites converted. - Setting charsets may put some special requirements on both XML declaration/ meta-http-equiv. (C.9) - UTF-8 encoding is in fact expected by XML!! - stylesheet element and attribute names are NOT converted to lowercase - ampersands (and entities in general I think) MUST be converted to an entity reference! (&amps;). This may mean further conversion of non-tag content before output to page. May be related to the charset issue as a whole. - Minimized values not allowed: Must do this: selected="selected" Please see the class t3lib_parsehtml for details. You can enable this function by the following values: **all** = the content is always processed before it may be stored in cache. **cached** = only if the page is put into the cache, **output** = only the output code just before it's echoed out.	
prefixLocalAnchors	string keyword	If set to one of the keywords, the content will have all local anchors in links prefixed with the path of the script. Basically this means that <a href="#"> will be transformed to <a href="path/path/script?params#">. This procedure is necessary if the <base> tag is set in the script (eg. if "realurl" extension is used to produce Speaking URLs). Keywords are the same as for "xhtml_cleaning", see above.	
disablePrefixComment	boolean	If set, the stdWrap property "prefixComment" will be disabled, thus preventing any revealing and spaceconsuming comments in the HTML source code. (ver. > 3.6.0RC1)	
baseURL	boolean or string	If set to "1" this will automatically write out the <base> tag with current sites URL in the header of the document. If you set the value to a string that is expected to be the URL. Examples: config.baseURL = http://typo3.org/sub_dir/ config.baseURL = 1	
tx_[extension key with no underscores]_[*]	-	Configuration space for plugins	

[tsref:config/->CONFIG]

"CONSTANTS":

Property:	Data type:	Description:	Default:
Array...	*string*	Constants. **Examples:** .EMAIL = *email@email.com* Now if parseFunc anywhere is configured with constants=1 then all cases of the string ###EMAIL### will be substituted in the text. see ->parseFunc	

[tsref:constants]

"PAGE":
Pages are referenced by two main values. The "id" and "type".

The "id" points to the uid of the page (or the alias). Thus the page is found.

The "type" is used to define how the page should be rendered. This is primarily used with framesets. Here the frameset normally has the type=0 (or not set) and the documents in the frameset would be defined with another type, eg. type=1 for the content-page.

You should explore the framesets of the Typo3-sites around. Also look in the standard-templates for framesets.

It's a good habit to use type=1 for the main-page of a website with frames. With no-frames sites type is normally zero.

Another good habit is to use "page" as the toplevel-objectname for the content-page on a website.

Most of this codes is executed in the PHP-script *pagegen.php*

Property:	Data type:	Description:	Default:
typeNum	*typeNumber*	This decides the the typeId of the page. **NOTE:** This value MUST be set and be unique!	
1,2,3,4...	cObject		
wrap	wrap	Wraps the content of the the cObject array	
stdWrap	->stdWrap	Wraps the content of the the cObject array with stdWrap options	
bodyTagCObject	cObject	This is default bodytag overridden by ".bodyTag" if that is set.	
bodyTag	<tag>	Bodytag on the page **Example:** <body bgcolor="{$bgCol}">	<body bgcolor="#FFFFFF">
bodyTagMargins	int	margins in the bodytag. **Property:** .useCSS = 1 (boolean) - will set a "BODY {margin: ...}" line in the in-document style declaration - for XHTML compliance. **Example:** value *4* adds *leftmargin="4" topmargin="4" marginwidth="4" marginheight="4"* to the bodyTag.	
bodyTagAdd	string	This content is added to the end of the bodyTag.	
bgImg	imgResource	Background image on the page. This is automatically added to the body-tag.	
frameSet	->FRAMESET	if any properties is set to this property, the page is made into a frameset.	
meta	->META		
shortcutIcon	resource	Favicon of the page. Create a reference to an icon here! Browsers that support favicons display them in the browser's address bar, next to the site's name in lists of bookmarks, and next to the page's title in a Tabbed Document Interface. **Note:** This must be a valid ".ico"-file (iconfile)	
headerData	->CARRAY	Inserts content in the header-section. Could be JavaScripts, meta-tags, other stylesheet references. Is inserted after all the style-definitions.	
config	->CONFIG	configuration for the page. Any entries override the same entries in the toplevel-object "config".	

includeLibs	*array of strings*	With this you may include php-files. This does the same as "includeLibrary" in ->CONFIG but this can include more than one file. These files are included *after* the file of includeLibrary.
		
		NOTE:
		The toplevel object "includeLibs" and the scripts defined with this property is added to each other. Script-keys (that is the "array of strings"-value, like below "ts_address") from this property of the page overrides any scripts-keys from the toplevel "includeLibs" property! The script-filenames are of the datatype "resource".
		
		Example:
		`includeLibs.ts_address = lib_filename.php`
		`includeLibs.ts_shop = lib_filename.php`
		
		Please do not use the prefix shown above ("ts_") as this will probably be used by the standard Typo3 libraries that will appear in the future.	

CSS Stylesheets:			

Property:	Data type:	Description:	Default:
stylesheet	resource	Inserts a stylesheet in the <HEAD>-section of the page; *<link rel="stylesheet" href="[resource]">*	

includeCSS.[array]	resource	Inserts a stylesheet (just like the .stylesheet property) by allows to setting up more than a single stylesheet, because you can enter files in an array.
		
		The file definition must be a valid "resource" datatype, otherwise nothing is inserted.
		
		Each file has *optional properties*:
		.media - setting the media attribute of the <style> tag.
		.title - setting the title of the <style> tag.
		.alternate - If set (boolean) then the rel-attribute will be "alternate stylesheet"
		.import - If set (boolean) then the @import way of including a stylesheet is used instead of <link>
		
		Example:
		`includeCSS {`
		`  file1 = fileadmin/mystylesheet1.css`
		`  file2 = stylesheet_uploaded_to_template*.css`
		`  file2.title = High contrast`
		`  file2.media = print`
		`}`	

CSS_inlineStyle	string	This value is just passed on as inline css (in-document css encapsulated in <style>-tags)	
insertClassesFromRTE	boolean	If set, the classes for the Rich Text Editor configured in Page TSconfig is inserted in as the first thing in the Style-section right after the setting of the stylesheet. **.add_mainStyleOverrideDefs** = [* / list of tags] - will add all the "RTE.default. mainStyleOverride_add" - tags configured as well. *Might be depreciated soon. Most likely the RTE should be configured by the stylesheet instead. Stay tuned...*	
noLinkUnderline	boolean	Disables link-underlining. Uses in-document stylesheet. *Depreciated. Use stylesheet instead.*	
hover	HTML-color	The color of a link when the mouse moves over it! (only MSIE). Uses in-document stylesheet. *Depreciated. Use stylesheet instead.*	
hoverStyle	string	Additional style information to the hover-color. **Example:** page.hoverStyle = font: bold; text-decoration: none; *Depreciated. Use stylesheet instead.*	
smallFormFields	boolean	Renders formfields like textarea, input and select-boxes small with "verdana size 1" font. Uses in-document stylesheet. **Tip:** Use this together with the config-option "compensateFieldWidth" set to "0.6" for netscape-browsers in order to render the small form fields in the same width! *Depreciated. Use stylesheet instead.*	
adminPanelStyles	boolean	Will include CSS styles for the Admin Panel.	

[tsref:(page)]

"FE_DATA":

Property:	Data type:	Description:	Default:
array of tableNames	->FE_TABLE		

[tsref:FEData]

"FE_TABLE":

Property:	Data type:	Description:	Default:

default.[field]	string	This property is in charge of which default-values is used for the table:
		
		Example:
		This defines the default values used for new records. These values will be overridden with any value submitted instead (as long as the submitted fields are allowed due to "allowNew")
		<pre>default {
		  subject = This is the default subject value!
		  hidden = 1
		  parent = 0
		}</pre>	

allowNew.[field]	string	This property is in charge of which fields that may be written from the frontend.
		
		Example:
		This defines that subject is a field, that may be submitted from the frontend. If a value is not submitted, subject is filled with the default value (see above).
		The field "hidden" on the other hand cannot be changed from the frontend. "hidden" will gain the value from the default definition (see above). If fields are set to "0" (zero) it's the same as if they were not defined in this array.
		<pre>allowNew {
		  subject = 1
		  hidden = 0
		}</pre>	

allowEdit.[field]	string	Same as above ("allowNew") but this controls which fields that may be written in case of an update of a record (and not a new submission) Please pay attension to the property below! ("overrideEdit")	

overrideEdit.[field]	string	This works like default-values above but is values inserted after the submitted values has beed processed. This means that opposite to default-values overwritten by the submitted values, these values override the submitted values.
		
		Example:
		In this case overrideEdit secures that if a user updates his record (if he "own" it) the "hidden"-field will be set no matter what.
		<pre>overrideEdit {
		  hidden = 1
		}</pre>	

userIdColumn	string (field)	This is a string that points to the column of a record where the user-id of the current fe_user should be inserted. This fe_user-uid is inserted/updated both by "new" and "edit"	
autoInsertPID	boolean	Works with new records: Insert automatically the PID of the page, where the submitted data is sent to. Any "pid" supplied from the submitted data will override. This is for convenience.	
processScript	resource	Include-script to be used for processing of incoming data to the table. The script is included from a function in the class tslib_fetce This is the really important option, because whether or not you are going to utilize the "cleaning"/"authorization" features of the properties above depend on how you write your script to process data and put it in the database. A very good example is to look at "media/scripts/guest_submit.inc", included from static_template "plugin.tt_guest" (Used for the default guestbook feature)	
separator	string	Separator character used when the submitted data is an array from eg. a multiple selector box.	chr(10) (linebreak)
doublePostCheck	string (fieldname)	Specifies a fieldname (integer) into which an integer-hash compiled of the submitted data is inserted. If the field is set, then submissions are checked whether another record with this value already exists. If so, the record is NOT inserted, because it's expected to be a "double post" (posting the same data more than once)	

[tsref:FEData.(tablename)/->FE_TABLE]

"FRAMESET":

Property:	Data type:	Description:	Default:
1,2,3,4...	frameObj	Configuration of frames and nested framesets.	
cols	<frameset>-data:cols	Cols	
rows	<frameset>-data:rows	Rows	
params	<frameset>-params	**Example:** border="0" framespacing="0" frameborder="NO"	

[tsref:(page).frameSet/->FRAMESET]

"FRAME":

Property:	Data type:	Description:	Default:
obj	*pointer to toplevel object-name*	toplevel object-name of a PAGE / FRAMESET **Example:** "left", "page", "frameset"	
options	*url-parameters*	**Example:** print=1&othervar=anotherthing would add '*&print=1&othervar=anotherthing*' to the ".src"-content (if not ".src" is set manually!!)	
params	<frame>-params	**Example:** scrolling="AUTO" noresize frameborder="NO"	
name	<frame>-data:name	Manually set name of frame **NOTE:** Is set automatically and should not be overridden under normal conditions!	value of ".obj"
src	<frame>-data:src	Manually set the src of the frame **NOTE:** Is set automatically and should not be overridden under normal conditions!	could be index.php?$id&$type

[tsref:(page).frameSet.(number)/->FRAMESET.(number)]

Example of a simple frameset with a topframe and content-frame:

```
frameset = PAGE
frameset.typeNum = 0

page = PAGE
page.typeNum = 1

top = PAGE
top.typeNum = 3

frameset.frameSet.rows = 150,*
frameset.frameSet.params = border="0" framespacing="0" frameborder="NO"
frameset.frameSet {
  1 = FRAME
  1.obj = top
  1.params = scrolling="NO" noresize frameborder="NO" marginwidth="0" marginheight="0"
  2 = FRAME
  2.obj = page
  2.params = scrolling="AUTO" noresize frameborder="NO"
}
```

"META":

Property:	Data type:	Description:	Default:
Array...	*string /stdWrap*	Metatags If value is empty (after trimming) the metatag is not generated. If the "key" (eg. "REFRESH" or "DESCRIPTION") is "REFRESH" (caseinsensitive), then the "http-equiv"-attribute is used in the metatag instead of "name". **Examples:** .REFRESH = *[sec]; [url, leave blank for same page]* .DESCRIPTION = This is the description of the content in this document .KEYWORDS = This is the keywords...	

[tsref:->META]

"CARRAY":

Property:	Data type:	Description:	Default:
1,2,3,4...	cObject	This is a numerical "array" of content-objects (cObjects). The order by which you specific the objects is not important as the array will be sorted before it's parsed!	
Occational properties:			
(stdWrap properties...)		**NOTE:** This applies ONLY if "CARRAY /stdWrap" is set to be data type If you specify any non-integer properties to a CARRAY, stdWrap will be invoked with all properties of the CARRAY. **Example:** This will return '<B>This will be rendered before "10"testing</B>' 10 = TEXT 10.value = testing 5 = HTML 5.value = This will be rendered before "10" wrap = <B> |</B>	
(TDParams)	<TD>-params	**NOTE:** This applies ONLY if "CARRAY +TDParams" is set to be data type This property is used only in some cases where CARRAY is used. Please look out for a note about that in the various cases.	

[tsref:->CARRAY]

Content Objects (cObject)

PHP-information:

The content objects (cObjects) are primarily controlled by the PHP-script "tslib/content.php". The PHP-class is named "tslib_cObj" and often this is also the variable-name of the objects ($cObj)

The $cObj in PHP has an array, $this->data, which holds records of various kind. See data type "getText".

This record is normally "loaded" with the record from a table depending on the situation. Say if your creating a menu it's often loaded with the page-record of the actual menuitem or if it's about content-rendering it'll be the content-record.

IMPORTANT NOTE :

When dealing with "cObjects", you're allowed to use a special syntax in order to reuse cObjects without actually creating a copy. This has the advantage of minimizing the size of the cached template. But on the other hand it doesn't give you the flexibility of overriding values.

This example will show you how it works:

```
#
# Temporary objects are defined:
#
lib.stdheader = COA
lib.stdheader {
  stdWrap.wrapAlign.field = header_position
  stdWrap.typolink.parameter.field = header_link
  stdWrap.fieldRequired = header

  1 = TEXT
  1.current = 1
  1.fontTag = {$content.wrap.header1}

  stdWrap.space = {$content.headerSpace}
}


#
# CType: header
#
tt_content.header = COA
tt_content.header {
  10 < lib.stdheader
  10.stdWrap.space >

  20 = TEXT
  20.field = subheader
  20.fontTag = {$content.wrap.subheader1}
}


#
# CType: bullet
#
tt_content.bullets = COA
tt_content.bullets {
  10 = < lib.stdheader
  20 < styles.content.bulletlist_gr
}
```

Comment: First lib.stdheader is defined. This is (and must be) a cObject ! (in this case, COA).

Now *lib.stdheader* is copied to *tt_content.header.10* with the "<" operator. This means that an actual copy of *lib.stdheader* is created at *parsetime*.

But this is not the case with *tt_content.bullets.10*. Here lib.stdheader is just pointed to and lib.stdheader will be used as the cObject at *runtime*.

The reason why lib.stdheader was copied in the first case is the fact that it's needed to unset ".stdWrap.space" inside the cObject ("10.stdWrap.space >"). This could NOT be done in the second case where only a pointer is created.

NOTE:

If *lib.stdheader* was *temp.stdheader* instead, the pointer would not work! This is due to the fact that the runtime-reference would find nothing in "temp." as this is unset before the template is stored in cache!

This goes for "temp." and "styles." (see the toplevel object definition elsewhere)

Overriding values anyway:

Although you can not override values TypoScript-style (using the operators and all) the properties of the object which has the reference will be merged with the config of the reference.

Example:
```
page.10 = TEXT
page.10.value = kasper
page.10.case = upper

page.20 = < page.10
page.20.case = lower
page.20.value >
page.20.field = pages
```

The result is this config:

Notice that .value was not cleared (the red line), because it's simply two arrays which are joined:

So hence the red line in the above example is useless.

HTML:

Property:	Data type:	Description:	Default:
value	HTML / stdWrap	Raw HTML-code.	

[tsref:(cObject).HTML]

Example:
```
10 = HTML
10.value = This is a text in uppercase
10.value.case = upper
```

Example:
```
10 = HTML
10.value.field = bodytext
10.value.br = 1
```

TEXT:

TEXT is very similar to the cObject "HTML". But the stdWrap is on the very rootlevel of the object. This is non-standard. Check the example.

Property:	Data type:	Description:	Default:
value	value	text, wrap with stdWrap properties	
(stdWrap properties...)			

[tsref:(cObject).TEXT]

Example:
```
10 = TEXT
10.value = This is a text in uppercase
```

```
10.case = upper
```

Example:
```
10 = TEXT
10.field = bodytext
10.br = 1
```

COBJ_ARRAY (COA, COA_INT):

This cObject has the alias COA. You can use this instead of COBJ_ARRAY.

You can also create this object as a COA_INT in which case it works exactly like the PHP_SCRIPT_INT object does: It's rendered non-cached! The COA_INT provides a way to utilize this feature not only with PHP_SCRIPT cObjects but any cObject.

Property:	Data type:	Description:	Default:
1,2,3,4...	cObject		
if	->if	if "if" returns false the COA is NOT rendered	
wrap	wrap		
stdWrap	->stdWrap		
includeLibs	*list of* resource	**(This property is used only if the object is COA_INT!, See introduction.)** This is a comma-separated list of resources that are included as PHP-scripts (with include_once() function) if this script is included. This is possible to do because any include-files will be known before the scripts are included. That's not the case with the regular PHP_SCRIPT cObject.	

[tsref:(cObject).COA/(cObject).COA_INT/(cObject).COBJ_ARRAY]

Example:
```
temp.menutable = COBJ_ARRAY
temp.menutable {
  10 = HTML
  10.value = <table border=0 cellpadding=0 cellspacing=0>

  20 = HMENU
  20.entryLevel = 0
  20.1 = GMENU
  20.1.NO {
    wrap = <tr><td> | </td></tr>
    XY = {$menuXY}
    backColor = {$bgCol}
    20  = TEXT
    20 {
      text.field = title
      fontFile = media/fonts/hatten.ttf
      fontSize = 23
      fontColor = {$menuCol}
      offset = |*| 5,18 || 25,18
    }
  }

  30 = HTML
  30.value = </table>
}
```

FILE:

PHP-function: $this->fileResource()

Property:	Data type:	Description:	Default:
file	resource	If the resource is **jpg,gif,jpeg,png** the image is inserted as an image-tag. Al other formats is read and inserted into the HTML-code. The maximum filesize of documents to be read is set to 1024 kb internally!	
linkWrap	linkWrap	(before ".wrap")	
wrap	wrap		
altText titleText	string / stdWrap	**For <img> output only!** If no titltext is specified, it will use the alttext instead If no alttext is specified, it will use an empty alttext	
longdescURL	string / stdWrap	**For <img> output only!** "longdesc" attribute (URL pointing to document with extensive details about image).	

[tsref:(cObject).FILE]

Example:

In this example a page is defined, but the content between the body-tags comes directly from the file "gs.html":

```
page.10 = FILE
page.10.file = fileadmin/gs/gs.html
```

IMAGE:

PHP-function: $cObj->cImage();

The array $GLOBALS["TSFE"]->lastImageInfo is set with the info-array of the returning image (if any) and contains width, height and so on.

Property:	Data type:	Description:	Default:
file	imgResource		
params	<IMG>-params		
border	integer	Value of the "border" attribute of the image tag.	0
altText titleText (alttext)	string / stdWrap	If no titltext is specified, it will use the alttext instead If no alttext is specified, it will use an empty alttext ("alttext" is the old spelling of this attribute. It will be used only if "altText" does not specify a value or properties)	
longdescURL	string / stdWrap	"longdesc" attribute (URL pointing to document with extensive details about image).	
linkWrap	linkWrap	(before ".wrap")	
imageLinkWrap	boolean/ ->imageLinkWrap	**NOTE:** ONLY active if linkWrap is NOT set and file is NOT GIFBUILDER (as it works with the original imagefile)	
if	->if	if "if" returns false the image is not shown!	
wrap	wrap		
stdWrap	->stdWrap		

[tsref:(cObject).IMAGE]

Example:

```
10 = IMAGE
10.file = toplogo*.gif
10.params = hspace=5
10.wrap = |<BR>
```

IMG_RESOURCE:

Returns only the image-reference, possibly wrapped with stdWrap. May be used for putting background images in tables or table-rows or to import a image in your own include-scripts.

Property:	Data type:	Description:	Default:
file	imgResource		

Property:	Data type:	Description:	Default:
stdWrap	->stdWrap		

[tsref:(cObject).IMG_RESOURCE]

CLEARGIF:

Inserts a transparent gif-file.

Property:	Data type:	Description:	Default:
height	<img>-data:height / stdWrap		1
width	<img>-data:width / stdWrap		1
wrap	wrap		|

[tsref:(cObject).CLEARGIF]

Example:

```
    20 = CLEARGIF
    20.height=20
```

CONTENT:

Generating content.

The register-key SYS_LASTCHANGED is updated with the tstamp-field of the records selected which has a higher value than the current.

Property:	Data type:	Description:	Default:
select	->select	The SQL-statement is set here!	
table	*tableName*	The table, the content should come from. In standard-configurations this will be "tt_content" **NOTE:** Only tables allowed are "pages" or tables prefixed with one of these: "tt_", "tx_", "ttx_", "fe_", "user_"	
renderObj	cObject		< [tablename]
wrap	wrap	Wrap the whole content-story...	
stdWrap	->stdWrap		

[tsref:(cObject).CONTENT]

Example (of the CONTENT-obj):

```
  1 = CONTENT
  1.table = tt_content
  1.select {
    pidInList = this
    orderBy = sorting
  }
```

Example (of record-renderObj's):

```
// Configuration for records with the typeField-value (often "CType") set to "header"
tt_content.header.default {
  10 = TEXT
  10.field = header
  .....
}

// Configuration for records with the typeField-value (often "CType") set to "bullets"
// If field "layout" is set to "1" or "2" a special configuration is use, else default
tt_content.bullets.subTypeField = layout
tt_content.bullets.default {
  .....
}
tt_content.bullets.1 {
  .....
}
tt_content.bullets.2 {
  .....
}

// This is what happens if the typeField-value does not match any of the above
tt_content.default.default {
  .....
```

```
}
```

RECORDS:

The register-key SYS_LASTCHANGED is updated with the tstamp-field of the records selected which has a higher value than the current.

NOTE: Records with parent ids (pid's) for non-accessible pages (that is hidden, timed or access-protected pages) are normally not selected. Pages may be of any type, except recycler. Disable the check with the "dontCheckPid"-option.

Property:	Data type:	Description:	Default:
source	*records-list* / stdWrap	List of record-id's, optionally with appended table-names. **Example:** tt_content_34, 45, tt_links_56	
tables	list /	List of accepted tables. If any items in the ".source"-list is not prepended with a tablename, the first table in this list is assumed to be the table for such records. Also tablenames configured in .conf is allowed. **Example:** tables = tt_content, tt_address, tt_links conf.tx_myexttable = TEXT conf.tx_myexttable.value = Hello world This adds the tables tt_content, tt_address, tt_links, tx_myexttable	
conf.[*tablename*]	cObject	Config-array which renders records from table *tablename*	If this is NOT defined, the rendering of the records is done with the toplevel-object [tablename] - just like the cObject, CONTENT!
wrap	wrap		
dontCheckPid	boolean	Normally a record cannot be selected, if it's parent page (pid) is not accessible for the website user. This option disables that check.	

[tsref:(cObject).RECORDS]

Example:

```
20 = RECORDS
20.source.field = records
20.tables = tt_address
20.conf.tt_address < tt_address.default
```

HMENU:
Generates hierarchical menus.

Property:	Data type:	Description:	Default:

(1 / 2 / 3 /...)	menuObj	**Required!**
		Defines which menuObj that should render the menuitems on the various levels.
		1 is the first level, 2 is the second level, 3 is the third level, 4 is ....
		
		Example:
		`temp.sidemenu = HMENU`
		`temp.sidemenu.1 = GMENU`
			(no menu)

entryLevel	int	Defines at which level in the rootLine, the menu should start. Default is "0" which gives us a menu of the very first pages on the site. If the value is < 0, entryLevel is chosen from "behind" in the rootLine. Thus "-1" is a menu with items from the outermost level, "-2" is the level before the outermost...	0
special	*"directory" / "list" / "updated" / "browse" / "rootline" / "keywords" / "language"*	(See separate table below)	
special.value	list of page-uid's / stdWrap	See above	
minItems	int	The minimum items in the menu. If the number of pages does not reach this level, a dummy-page with the title "..." and uid=[currentpage_id] is inserted. **Notice:** Affects all sub menus as well. To set the value for each menu level individually, set the properties in the menu objects (see "Common properties" table).	
maxItems	int	The maximum items in the menu. More items will be ignored. **Notice:** Affects all sub menus as well. (See "minItems" for notice)	
begin	int +calc	The first item in the menu. **Example:** This results in a menu, where the first two items are skipped starting with item number 3: `begin = 3` **Notice:** Affects all sub menus as well. (See "minItems" for notice)	
excludeUidList	list of int	This is a list of page uid's to exclude when the select statement is done. Comma-separated. **Example:** The pages with these uid-number will NOT be within the menu!! 34,2	
alwaysActivePIDlist	list of integers	This is a list of page UID numbers that will always be regarded as active menu items and thereby automatically opened regardless of the rootline.	
protectLvar	boolean / keyword	If set, then for each page in the menu it will be checked if the GET variable "&L" has a value different from the current sys_language_uid and if an Alternative Page Language record for that sys_language_uid exists for the page. If that is not the case and the pages "Localization settings" have the "Hide page if no translation for current language exists" flag set, then the menu item will link to a non accessible page that will yield an error page to the user. Setting this option will prevent that situation by simply adding "&L=0" for such pages, meaning that they will switch to the default language rather than keeping the current language. **Keyword: "all"** When set to "all" the same check is carried out but it will not look if "Hide page if no translation for current language exists" is set - it always reverts to default language if no translation is found. For these options to make sense, they should only be used when "config.sys_language_mode" is not set to "content_fallback".	
if	->if	If "if" returns false, the menu is not generated	
wrap	wrap		
stdWrap	->stdWrap		

[tsref:(cObject).HMENU]

Example:

```
temp.sidemenu = HMENU
temp.sidemenu.entryLevel = 1
temp.sidemenu.1 = TMENU
temp.sidemenu.1 {
  target = page
  NO.afterImg = media/bullets/dots2.gif |*||*| _
  NO.afterImgTagParams = hspace="4"
  NO.linkWrap = {$fontTag}
  NO.ATagBeforeWrap = 1

  ACT < .NO
  ACT = 1
  ACT.linkWrap = <b>{$fontTag}</b>
}
```

The .special property

This property makes it possible to create menus that are not strictly reflecting the current page-structure, but rather creating menus with links to pages like "next/previous", "last modified", "pages in a certain page" and so on.

NOTE: Don't set .entryLevel for a HMENU when using this option! Also be aware that this selects pages for the first level in the menu. Submenus by menuPbjects 2+ will be created as usual.

.special.value always has stdWrap-properties!

Their properies are put in this tables:

Type:	Description:	Default:

directory	This will generate a menu of all pages with pid = 35 and pid = 56.
	`20 = HMENU`
	`20.special = directory`
	`20.special.value = 35, 56`
	
	If .value is not set, the default pid is the current page.
	
	Support for Mount Pages: Yes.	

list	This will generate a menu with the two pages (uid=35 and uid=36) listed:
	`20 = HMENU`
	`20.special = list`
	`20.special.value = 35, 56`
	
	If .value is not set, the default uid is the .entryLevel uid.
	
	Support for Mount Pages: Yes.	

Type:	Description:	Default:

updated	This will generate a menu of the most recently updated pages from the branches in the tree starting with the uid's (uid=35 and uid=36) listed. Furthermore the field "tstamp" is used (default is SYS_LASTCHANGED) and the treedepth is 2 levels. Also there will be shown a maximum of 8 pages and they must have been updated within the last three days (3600*24*3):
	
	``
	20 = HMENU
	20.special = updated
	20.special.value = 35, 56
	20.special {
	  mode = tstamp
	  depth = 2
	  maxAge = 3600*24*3
	  limit = 8
	}
	``
	
	Ordering is by default done in reverse order (desc) with the field specified by "mode" , but setting "alternativeSortingField" for the menu object (eg GMENU, see later) will override that. Properties "mode", "depth", "maxAge" and "limit" is only used with special="updated".
	mode: Which field in the pages-table to use. Default is "SYS_LASTCHANGED" (which is updated when a page is generated to the youngest tstamp of the records on the page), "manual" or "lastUpdated" will use the field "lastUpdated" (set manually in the page-record) and "tstamp" will use the "tstamp"-field of the pagerecord, which is set automatically when the record is changed. "crdate" will use "crdate"-field of the pagerecord. "starttime" will use the starttime field.
	Fields with zero value is not selected anyway.
	depth: By default (if the value is not an integer) the depth is 20 levels. The range is 1-20. A depth of 1 means only the start id, depth of 2 means start-id + first level. NOTE: depth is relative to beginAtLevel.
	beginAtLevel: Integer. Determines starting level for the pagetrees generated based on .value and .depth. Zero is default and includes the start id. 1=starts with the first row of subpages, 2=starts with the second row of subpages. Depth is relative to this starting point.
	maxAge: Seconds+calc. Pages with update-dates older than currenttime minus this number of seconds will not be shown in the menu no matter what. Default is "not used". You may use +-*/ for calculations.
	limit: Max number of items in the menu. Default is 10, max is 100.
	excludeNoSearchPages: Boolean. If set, pages marked "No search" is not included into special-menus.
	
	Support for Mount Pages: Yes.	

rootline	Creates a menu with pages from the "rootline" (see earlier in this reference)
	.range = [begin-level]
		[end-level] (same way as you reference the .entryLevel for HMENU)
		.target_[0-x] targets
		
		This...
		
		``
		page.2 = HMENU
		page.2.special = rootline
		page.2.special.range = 1|-2
		page.2.special.targets.3 = page
		page.2.1 = TMENU
		page.2.1.target = _top
		page.2.1.wrap = <HR>
			<HR>
			page.2.1.NO {
			  linkWrap =
				>
				}
				``
				
				... creates a menu like this:
				Page level 1 > Page level 2 > Page level 3 > Page level 4 >
				(The menu starts at level 1 and does NOT link to the current page (-2 is the level before).
				Further all pages on level 3 will have "page" as target and all other "_top")
				
				Support for Mount Pages: Yes.	

Type:	Description:	Default:

browse	This kind of menu is built of items given by a list from the property ".item". Each element in the list (sep. by "|") is either a reserved itemname (see list) with a predefined function or a userdefined name which you can assign a link to any page. Note that the current page cannot be the root-page of a site.
	
	Support for Mount Pages: No!
	
	## Main properties:
	.items ("|" separated list of "itemnames")
	.[itemnames].target (target) - optional/alternative target of the item
	.[itemnames].uid (uid of page) - optional/alternative page-uid to link to
	.[itemnames].fields.[fieldname] (string) - override field "fieldname" in pagerecord.
	.prevnextToSection (boolean) - if set, the "prev" and "next" navigation will jump to the next section when it reaches the end of pages in the current section
	.value (page-uid) - default is current page id. Seldomly you might want to override this value with another page-uid which will then act as the basepoint for the menu and the predefined items.
	Ordering is by default done in reverse order (desc) with the field specified by "mode" , but setting "alternativeSortingField" for the menu object (eg GMENU, see later) will override that.
	Reserved itemnames:
	next / prev : links to next page / previous page. Next and previous pages are from the same "pid" as the current page id (or "value") - that is the next item in a menu with the current page. Also referred to as current level.
	If ".prevnextToSection" is set then next/prev will link to the first page of next section / last page of previous section.
	nextsection / prevsection : links to next section / previous section. A section is defined as the subpages of a page on the same level as the parent (pid) page of the current page. Will not work if parent page of current page is the root page of the site.
	nextsection_last | prevsection_last: Where nextsection/prevsection links to the first page in a section, these links to the last pages. If there is only one page in the section that will be both first and last. Will not work if parent page of current page is the root page of the site.
	first / last : First / Last page on current level. If there is only one page on the current level that page will be both first and last.
	up : Links to the parent (pid) page of the current page. (up 1 level) Will always be available
	index : Links to the parent of the parent page of the current page (up 2 levels). May not be available if that page is out of the rootline.
	
	Examples:
	If id=20 is current page then:
	21= prev and first, 19 = next, 18 = last, 17 = up, 1=index, 10 = nextsection, 11 = nextsection_last
	prevsection and prevsection_last is not present because id=3 has no subpages!
	
	TypoScript (only "browse"-part, needs also TMENU/GMENU):
	`xxx = HMENU`
	`xxx.special = browse`
	`xxx.special {`
	`   items = index|up|next|prev`
	`   items.prevnextToSection = 1`
	`   index.target = _blank`
	`   index.fields.title = INDEX`
	`   index.uid = 8`
	`}`	

keywords	Makes a menu of pages with one or more keywords also found on the current page. **.value** = page for which keywords to find similar pages. **.mode:** Which field in the pages-table to use for sorting. Default is "SYS_LASTCHANGED" (which is updated when a page is generated to the youngest tstamp of the records on the page), "manual" or "lastUpdated" will use the field "lastUpdated" (set manually in the page-record) and "tstamp" will use the "tstamp"-field of the pagerecord, which is set automatically when the record is changed. "crdate" will use "crdate"-field of the pagerecord. "starttime" will use the starttime field. **.entryLevel** = where in the rootline the search begins. Standard rootline syntax (-x to x) **.depth, .limit, .excludeNoSearchPages, .beginAtLevel** (like "updated" menu) **.setKeywords** (+stdWrap) = lets you define the keywords manually by defining them as a commaseparated list. If this property is defined, it overrides the default, which is the keywords of the current page. **.keywordsField** = defines the field in the pages-table in which to search for the keywords. Default is the fieldname "keyword". No check is done to see if the field you enter here exists, so enter an existing field, OK?! **.keywordsField.sourceField** = defines the field from the current page from which to take the keywords being matched. The default is "keyword". (Notice that ".keywordsField" is only setting the page-record field to *search in* !) Support for Mount Pages: Yes.	

Type:	Description:	Default:

language	Creates a language selector menu. Typically this is made as a menu with flags for each language a page is translated to and when the user clicks any element the same page id is hit but with a change to the "&L" parameter in the URL.
	
	The "language" type will create menu items based on the current page record but with the language record for each language overlaid if available. The items all link to the current page id and only "&L" is changed.
	
	Item states:
	When "TSFE->sys_language_uid" matches the sys_language uid for an element the state is set to "ACT", otherwise "NO". However, if a page is not available due to the pages "Localization settings" (which can disable translations) or if no Alternative Page Language record was found (can be disabled with ".normalWhenNoLanguage", see below) the state is set to "USERDEF1" for non-active items and "USERDEF2" for active items. So in total there are four states to create designs for. It is recommended to disable the link on menu items rendered with "USERDEF1" and "USERDEF2" in this case since they are disabled exactly because a page in that language does not exist and might even issue an error if tried accessed (depending on site configuration).
	
	.value = comma list of sys_language uids to construct the menu with. The number of elements in this list determines the number of menu items.
	.normalWhenNoLanguage = boolean, which if set will render the button for a language as a non-disabled button even if no translation is found for the language.
	
	Example:
	Creates a language menu with flags (notice that some lines break):
	
	lib.langMenu = HMENU
	lib.langMenu.special = language
	lib.langMenu.special.value = 0,1,2
	lib.langMenu.1 = GMENU
	lib.langMenu.1.NO {
	  XY = [5.w]+4, [5.h]+4
	  backColor = white
	  5 = IMAGE
	  5.file = media/flags/flag_uk.gif || media/flags/flag_fr.gif || media/flags/flag_es.gif
	  5.offset = 2,2
	}
	
	lib.langMenu.1.ACT < lib.langMenu.1.NO
	lib.langMenu.1.ACT=1
	lib.langMenu.1.ACT.backColor = black
	
	lib.langMenu.1.USERDEF1 < lib.langMenu.1.NO
	lib.langMenu.1.USERDEF1=1
	lib.langMenu.1.USERDEF1.5.file = media/flags/flag_uk_d.gif || media/flags/flag_fr_d.gif ||
	media/flags/flag_es_d.gif
	lib.langMenu.1.USERDEF1.noLink = 1	

Type:	Description:	Default:

userdefined	Lets you write your own little PHP-script that generates the array of menuitems.
	.file [resource] = filename of the php-file to include. (Just like cObject PHP_SCRIPT)
	.[any other key] = your own variables to your script. They are all accessible in the array $conf in your script
	
	Howto:
	You must populate an array called $menuItemsArray with page-records of the menuitems you want to be in the menu.
	It goes like this:
	
	`$menuItemsArray[] = pageRow1;`
	`$menuItemsArray[] = pageRow2;`
	`$menuItemsArray[] = pageRow3;`
	`...`
	
	A "pageRow" is a record from the table "pages" with all fields selected (SELECT * FROM...)
	If you create fake page rows, make sure to add at least "title" and "uid" field values.
	
	Notice:
	If you work with mount-points you can set the MP param which should be set for the page by setting the internal field "_MP_PARAM" in the page-record (xxx-xxx).
	
	Overriding URLs:
	You can also use the internal field "_OVERRIDE_HREF" to set a custom href-value (eg. "http://www.typo3.org") which will in any case be used rather than a link to the page that the page otherwise might represent. If you use "_OVERRIDE_HREF" then "_OVERRIDE_TARGET" can be used to override the target value as well (See example below).
	
	Other reserved keys:
	"_ADD_GETVARS" can be used to add get parameters to the URL, eg. "&L=xxx".
	"_SAFE" can be used to protect the element to make sure it is not filtered out for any reason.
	
	Creating submenus:
	You can create submenus for the next level easily by just adding an array of menu items in the internal field "_SUB_MENU" (See example below).
	
	Presetting element state
	If you would like to preset an element to be recognized as a SPC, IFSUB, ACT, CUR or USR mode item, you can do so by specifying one of these values in the key "ITEM_STATE" of the page record. This setting will override the natural state-evaluation.	

userfunction	Calls a user function/method in class which should (as with "userdefined" above) return an array with page records for the menu. **.userFunc** = function-name	

[tsref:(cObject).HMENU.special]

Example: Creating hierarchical menus of custom links

By default the HMENU object is designed to create menus from pages in TYPO3. Such pages are represented by their page-record contents. Usually the "title" field is used for the title and the "uid" field is used to create a link to that page in the menu.

However the HMENU and sub-menu objects are so powerful that it would be very useful to use these objects for creating menus of links which does not relate to pages in TYPO3 by their ids. This could be a menu reflecting a menu structure of a plugin where each link might link to the same page id in TYPO3 but where the difference would be in some parameter value.

This can be easily done with the special-type "userdefined" (see table above) where you can return an array of menu items customly build in a PHP-script you write.

First, this listing creates a menu in three levels where the first two are graphical items:

```
  0:  # ************************
  1:  # MENU LEFT
  2:  # ************************
  3:  lib.leftmenu.20 = HMENU
  4:  lib.leftmenu.20.special = userfunction
  5:  lib.leftmenu.20.special.userFunc = user_3dsplm_pi2->makeMenuArray
  6:  lib.leftmenu.20.1 = GMENU
  7:  lib.leftmenu.20.1.NO {
  8:    wrap = <tr><td>|</td></tr><tr><td class="bckgdgrey1" height="1"></td></tr>
  9:    XY = 163,19
 10:    backColor = white
 11:    10 = TEXT
 12:    10.text.field = title
 13:    10.text.case = upper
 14:    10.fontColor = red
 15:    10.fontFile = fileadmin/fonts/ARIALNB.TTF
 16:    10.niceText = 1
 17:    10.offset = 14,12
```

```
18:     10.fontSize = 10
19: }
20: lib.leftmenu.20.2 = GMENU
21: lib.leftmenu.20.2.wrap = | <tr><td class="bckgdwhite" height="4"></td></tr><tr><td
class="bckgdgrey1" height="1"></td></tr>
22: lib.leftmenu.20.2.NO {
23:   wrap = <tr><td class="bckgdwhite" height="4"></td></tr><tr><td>|</td></tr>
24:   XY = 163,16
25:   backColor = white
26:   10 = TEXT
27:   10.text.field = title
28:   10.text.case = upper
29:   10.fontColor = #666666
30:   10.fontFile = fileadmin/fonts/ARIALNB.TTF
31:   10.niceText = 1
32:   10.offset = 14,12
33:   10.fontSize = 11
34: }
35: lib.leftmenu.20.2.RO < lib.leftmenu.20.2.NO
36: lib.leftmenu.20.2.RO = 1
37: lib.leftmenu.20.2.RO.backColor = #eeeeee
38: lib.leftmenu.20.2.ACT < lib.leftmenu.20.2.NO
39: lib.leftmenu.20.2.ACT = 1
40: lib.leftmenu.20.2.ACT.10.fontColor = red
41: lib.leftmenu.20.3 = TMENU
42: lib.leftmenu.20.3.NO {
43:   allWrap = <tr><td>|</td></tr>
44:   linkWrap (
45:    <table border="0" cellpadding="0" cellspacing="0" style="margin: 2px; 0px; 2px; 0px;">
46:       <tr>
47:         <td><img src="clear.gif" width="15" height="1" /></td>
48:         <td><img src="fileadmin/arrow_gray.gif" height="9" width="9" hspace="3" /></td>
49:         <td>|</td>
50:       </tr>
51:    </table>
52:   )
53: }
```

The menu looks like this on a webpage:

The TypoScript code above generates this menu, but the items does not link straight to pages as usual. This is because the *whole* menu is generated from this array, which was returned from the function "menuMenuArray" called in TypoScript line 4+5

```
1:      function makeMenuArray($content,$conf)     {
2:          return array(
3:              array(
4:                  'title' => 'Contact',
5:                  '_OVERRIDE_HREF' => 'index.php?id=10',
6:                  '_SUB_MENU' => array(
7:                      array(
8:                          'title' => 'Offices',
9:                          '_OVERRIDE_HREF' => 'index.php?id=11',
10:                         '_OVERRIDE_TARGET' => '_top',
11:                         'ITEM_STATE' => 'ACT',
12:                         '_SUB_MENU' => array(
13:                             array(
14:                                 'title' => 'Copenhagen Office',
15:                                 '_OVERRIDE_HREF' => 'index.php?id=11&officeId=cph',
16:                             ),
17:                             array(
18:                                 'title' => 'Paris Office',
19:                                 '_OVERRIDE_HREF' => 'index.php?id=11&officeId=paris',
20:                             ),
21:                             array(
22:                                 'title' => 'New York Office',
```

```
23:                              '_OVERRIDE_HREF' => 'http://www.newyork-office.com',
24:                              '_OVERRIDE_TARGET' => '_blank',
25:                          )
26:                      )
27:                  ),
28:                  array(
29:                      'title' => 'Form',
30:                      '_OVERRIDE_HREF' => 'index.php?id=10&cmd=showform',
31:                  ),
32:                  array(
33:                      'title' => 'Thank you',
34:                      '_OVERRIDE_HREF' => 'index.php?id=10&cmd=thankyou',
35:                  ),
36:              ),
37:          ),
38:          array(
39:              'title' => 'Products',
40:              '_OVERRIDE_HREF' => 'index.php?id=14',
41:          )
42:      );
43:  }
```

Notice how the array contains "fake" page-records which has *no* uid field, only a "title" and "_OVERRIDE_HREF" as required and some other fields as it fits.

- The first level with items "Contact" and "Products" contains "title" and "_OVERRIDE_HREF" fields, but "Contact" extends this by a "_SUB_MENU" array which contains a similar array of items.
- The first item on the second level, "Offices", contains a field called "_OVERRIDE_TARGET". Further the item has its state set to "ACT" which means it will render as an "active" item (you will have to calculate such stuff manually when you are not rendering a menu of real pages!). Finally there is even another sub-level of menu items.

CTABLE:
Creates a standard-table where you can define the content of the the various cells

Property:	Data type:	Description:	Default:
offset	x,y	Offset from upper left corner	0,0 = intet
tm	->CARRAY +TDParams	topMenu	
lm	->CARRAY +TDParams	leftMenu	
rm	->CARRAY +TDParams	rightMenu	
bm	->CARRAY +TDParams	bottomMenu	
c	->CARRAY +TDParams	content-cell	
cMargins	margins	Distance around the content-cell "c"	0,0,0,0
cWidth	pixels	Width of the content-cell "c"	
tableParams	<TABLE>-params		border=0 cellspacing=0 cellpadding=0

[tsref:(cObject).CTABLE]

Example:
```
page.10 = CTABLE
page.10 {
  offset = 5, 0
  tableParams = border=0 width=400
  cWidth=400
  c.1 = CONTENT
  c.1.table = tt_content
  c.1.select {
    pidInList = this
    orderBy = sorting
  }

  tm.10 < temp.sidemenu
  tm.TDParams = valign=top
}
```

OTABLE:

Property:	Data type:	Description:	Default:
offset	x,y	Offset from upper left corner **Note:** Actually the datatype is "x,y,r,b,w,h": x,y is offset from upperleft corner r,b is offset (margin) to right and bottom w is the required width of the content field h is the required height of the content field All measures is in pixels.	
1,2,3,4...	cObject		
tableParams	<TABLE>-params		border=0 cellspacing=0 cellpadding=0

[tsref:(cObject).OTABLE]

Example:
```
top.100 = OTABLE
top.100.offset = 310,8
top.100.tableParams = border=0 cellpadding=0 cellspacing=0
top.100.1 < temp.topmenu
```

COLUMNS:

Property:	Data type:	Description:	Default:
tableParams	<TABLE>-params		border=0 cellspacing=0 cellpadding=0
TDparams	<TD>-params		valign=top
rows	int (Range: 2-20)	The number of rows in the columns.	2
totalWidth	int	The total-width of the columns+gaps	
gapWidth	int /stdWrap +optionSplit	Width of the gap between columns. 0 = no gap	
gapBgCol	HTML-color / stdWrap +optionSplit	background-color for the gap-tablecells	
gapLineThickness	int /stdWrap +optionSplit	lineThickness of the dividerline in the gap between cells 0 = no line	
gapLineCol	HTML-color / stdWrap +optionSplit	Line color	black
[column-number] 1,2,3,4...	cObject	This is the content-object for each column!!	
after	cObject	This is a cObject placed after the columns-table!!	
if	->if	if "if" returns false the columns are not rendered!	
stdWrap	->stdWrap		

[tsref:(cObject).COLUMNS]

HRULER:

Property:	Data type:	Description:	Default:
lineThickness	int /stdWrap	Range: 1-50	1
lineColor	HTML-color	The color of the ruler.	black
spaceLeft	pixels	space before the line (to the left)	
spaceRight	pixels	space after the line (to the right)	
tableWidth	string	Width of the ruler ("width" attribute in a table)	99%
stdWrap	->stdWrap		

[tsref:(cObject).HRULER]

IMGTEXT:

This object is designed to align images and text. This is normally used to render text/picture records from the tt_content table.

The image(s) are placed in a table and the table is placed before, after or left/right relative to the text.

See code-examples.

Property:	Data type:	Description:	Default:
text	->CARRAY / stdWrap	Use this to import / generate the content, that should flow around the imageblock.	
textPos	int / stdWrap	Textposition: bit[0-2]: 000 = centre, 001 = right, 010 = left bit[3-5]: 000 = over, 001 = under, 010 text 0 - Above: Centre 1 - Above: Right 2 - Above: Left 8 - Below: Centre 9 - Below: Right 10 - Below: Left 17 - In Text: Right 18 - In Text: Left 25 - In Text: Right (no wrap) 26 - In Text: Left (no wrap)	
textMargin	pixels /stdWrap	margin between the image and the content	
textMargin_outOfText	boolean	If set, the textMargin space will still be inserted even if the image is placed above or below the text. This flag is only for a kind of backwards compatibility because this "feature" was recently considered a bug and thus corrected. So if anyone has depended on this way things are done, you can compensate with this flag.	
imgList	list of imagefiles / stdWrap	list of images from ".imgPath" **Example:** This imports the list of images from tt_content's image-field "imgList.field = image"	
imgPath	path /stdWrap	Path to the images **Example:** "uploads/pics/"	
imgMax	int /stdWrap	max number of images	
imgStart	int /stdWrap	start with image-number ".imgStart"	
imgObjNum	*imgObjNum* +optionSplit	Here you define, which IMAGE-cObjects from the array "1,2,3,4..." in this object that should render the images. "current" is set to the image-filename. **Example:** "imgObjNum = 1 |*||*| 2": This would render the first two images with "1. ..." and the last image with "2. ...", provided that the ".imgList" contains 3 images.	
1,2,3,4	->IMAGE (cObject)	Rendering of the images The register "IMAGE_NUM" is set with the number of image being rendered for each rendering of a image-object. Starting with zero. The image-object should not be of type GIFBUILDER! **Important:** "file.import.current = 1" fetches the name of the images!	
caption	->CARRAY / stdWrap	Caption	
captionAlign	align /stdWrap	Caption alignment	default = ".textPos"
captionSplit	boolean	If this is set, the caption text is split by the character (or string) from ".token" , and every item is displayed under an image each in the image block. .token = (string/stdWrap) Character to split the caption elements (default is chr(10)) .cObject = cObject, used to fetch the caption for the split .stdWrap = stdWrap properties used to render the caption.	
altText titleText	string / stdWrap	Default altText/titleText if no alternatives are provided by the ->IMAGE cObjects If no titltext is specified, it will use the alttext instead If no alttext is specified, it will use an empty alttext	

TYPO3
get.content.right

Property:	Data type:	Description:	Default:
longdescURL	string / stdWrap	Default longdescURL if no alternatives are provided by the ->IMAGE cObjects "longdesc" attribute (URL pointing to document with extensive details about image).	
border	boolean /stdWrap	If true, a border i generated around the images.	
borderCol	HTML-color / stdWrap	Color of the border, if ".border" is set	black
borderThick	pixels /stdWrap	Width of the border around the pictures	1
cols	int /stdWrap	Columns	
rows	int /stdWrap	Rows (higher priority thab "cols")	
noRows	boolean /stdWrap	If set, the rows are not divided by a table-rows. Thus images are more nicely shown if the height differs a lot (normally the width is the same!)	
noCols	boolean /stdWrap	If set, the columns are not made in the table. The images are all put in one row separated by a clear giffile to space them apart. If noRows is set, noCols will be unset. They cannot be set simultaneously.	
colSpace	int /stdWrap	space between columns	
rowSpace	int /stdWrap	space between rows	
spaceBelowAbove	int /stdWrap	Pixelsspace between content an images when position of image is above or belox text (but not in text)	
tableStdWrap	->stdWrap	This passes the final <table> code for the image block to the stdWrap function.	
maxW	int /stdWrap	max width of the image-table. This will scale images not in the right size! Takes the number of columns into account! **NOTE:** Works ONLY if IMAGE-obj is NOT GIFBUILDER	
maxWInText	int /stdWrap	max width of the image-table, if the text is wrapped around the image-table (on the left or right side). This will scale images not in the right size! Takes the number of columns into account! **NOTE:** Works ONLY if IMAGE-obj is NOT GIFBUILDER	50% of maxW
equalH	int /stdWrap	If this value is greater than zero, it will secure that images in a row has the same height. The width will be calculated. If the total width of the images raise above the "maxW"-value of the table the height for each image will be scaled down equally so that the images still have the same height but is within the limits of the totalWidth. Please note that this value will override the properties "width", "maxH", "maxW", "minW", "minH" of the IMAGE-objects generating the images. Furthermore it will override the "noRows"-property and generate a table with no columns instead!	
colRelations	string /stdWrap	This value defines the width-relations of the images in the columns of IMGTEXT. The syntax is "[int] : [int] : [int] : ..." for each column. If there are more imagecolumns than figures in this value, it's ignored. If the relation between two of these figures exceeds 10, this function is ignore. It works only fully if all images are downscaled by their maxW-definition. **Example:** If 6 images are placed in three columns and their width's are high enough to be forcibly scaled, this value will scale the images in the to be eg. 100, 200 and 300 pixels from left to right 1 : 2 : 3	

Property:	Data type:	Description:	Default:
image_compression	int /stdWrap	Image Compression: 0= Default 1= Dont change! (removes all parameters for the image_object!!) (adds gif-extension and color-reduction command) 10= GIF/256 11= GIF/128 12= GIF/64 13= GIF/32 14= GIF/16 15= GIF/8 (adds jpg-extension and quality command) 20= IM: -quality 100 21= IM: -quality 90 <=> Photoshop 60 (JPG/Very High) 22= IM: -quality 80 (JPG/High) 23= IM: -quality 70 24= IM: -quality 60 <=> Photoshop 30 (JPG/Medium) 25= IM: -quality 50 26= IM: -quality 40 (JPG/Low) 27= IM: -quality 30 <=> Photoshop 10 28= IM: -quality 20 (JPG/Very Low) The default ImageMagick quality seems to be 75. This equals Photoshop quality 45. Images compressed with ImageMagick with the same visual quality as a Photoshop-compressed image seems to be largely 50% greater in size!! **NOTE:** Works ONLY if IMAGE-obj is NOT GIFBUILDER	
image_effects	int /stdWrap	Adds these commands to the parameteres for the scaling. This function has no effect if "image_compression" above is set to 1!! 1 => "-rotate 90", 2 => "-rotate 270", 3 => "-rotate 180", 10 => "-colorspace GRAY", 11 => "-sharpen 70", 20 => "-normalize", 23 => "-contrast", 25 => "-gamma 1.3", 26 => "-gamma 0.8" **NOTE:** Works ONLY if IMAGE-obj is NOT GIFBUILDER	
image_frames	Array + .key /stdWrap	**Frames:** .key points to the frame used. ".image_frames.x" is imgResource-mask (".m")properties which will override to the [imgResource].m properties of the imageObjects. This is used to mask the images into a frame. See how it's done in the default configuration and IMGTEXT in the static_template-table. **Example:** <code>1 {</code> <code> mask = media/uploads/darkroom1_mask.jpg</code> <code> bgImg = GIFBUILDER</code> <code> bgImg {</code> <code> XY = 100,100</code> <code> backColor = {$bgCol}</code> <code> }</code> <code> bottomImg = GIFBUILDER</code> <code> bottomImg {</code> <code> XY = 100,100</code> <code> backColor = black</code> <code> }</code> <code> bottomImg_mask = media/uploads/darkroom1_bottom.jpg</code> <code>}</code> **NOTE:** This cancels the jpg-quality settings sent as ordinary ".params" to the imgResource. In addition the output of this operation will always be jpg or gif! **NOTE:** Works ONLY if IMAGE-obj is NOT GIFBUILDER	

Property:	Data type:	Description:	Default:
netprintApplicationLink	boolean / +properties	Set true, if you would like to enable the netprint feature. "Netprint" is a Typo3 application which is still in development. 'Netprint' allows film development stores to offer online submission of Photos to a Typo3 / Netprint server in-house from which the store offer enlargements of digital photos on eg Kodak Royal paper. The application will be released under GNU/GPL sometime in 2002. 'Netprint' offers a feature to fetch images from peoples online fotoalbums. This property enables links from your site images to a Netprint server. **Properties:** **.url** = Url of the Netprint site page, eg. http://netprint.photo.com/?id=999&type=2 **.thisUrl** = Url of this site, eg. http://mysite.com/typo3site/ (default is calculated from HTTP_HOST) **.cObject** = (cObject) The link content **.ATagParams** = <A>-tag parameters **.outerStdWrap** = (->stdWrap) stdWrap of the total linked cObject. **.before** = (boolean) If the link code should be inserted before the image (default is after) **.linkOnlyPixelsAbove** = (int) If set, only images with a total number of pixels greater than this number will be linked. This secures that small images will not be linked. Eg. if images above 640x480 is supposed to be linked, the limit should be approx. 300000	
editIcons	string	(See stdWrap.editIcons)	
noStretchAndMargin Cells	boolean	If set (1), the cells used to add left and right margins plus stretch out the table will not be added. You will loose the ability to set margins for the object if entered "in text". So it's not recommended, but it has been requested by some people for reasons.	
stdWrap	->stdWrap		

[tsref:(cObject).IMGTEXT]

Example:

```
tt_content.textpic.default {
  5 = IMGTEXT
  5 {
    text < tt_content.text.default
    imgList.field = image
    textPos.field = imageorient
    imgPath = uploads/pics/
    imgObjNum = 1
    1 {
      file.import.current = 1
      file.width.field = imagewidth
      imageLinkWrap = 1
      imageLinkWrap {
        bodyTag = <BODY bgColor=black>
        wrap = <A href="javascript:close();"> | </A>
        width = 800m
        height = 600m
        JSwindow = 1
        JSwindow.newWindow = 1
        JSwindow.expand = 17,20
      }
    }
    maxW = 450
    maxWInText = 300
    cols.field = imagecols
    border.field = imageborder
    caption {
      1 = TEXT
      1.field = imagecaption
      1.wrap = <font size="1"> |</font>
      1.wrap2 = {$cBodyTextWrap}
    }
    borderThick = 2
    colSpace = 10
    rowSpace = 10
    textMargin = 10
  }
  30 = HTML
  30.value = <br>
}
```

CASE:

This provides something alike a switch-construct in PHP. The property "key" is supposed to equal the name of another property in the object (*Array...*) which is a cObject. If the property .[key] is defined, "default" will be used.

Strings is Array... can be anything except the reserved words "key", "default", "stdWrap", "if"

Property:	Data type:	Description:	Default:
setCurrent	string /stdWrap	Sets the "current"-value.	
key	string /stdWrap	This is the	
default	cObject		
Array...	cObject		
stdWrap	->stdWrap		
if	->if	if "if" returns false nothing is returned	

[tsref:(cObject).CASE]

Example:

This example chooses between two different renderings of some content depending on whether the field "layout" is "1" of not ("default"). The result is in either case wrapped with "|
". If the field "header" turns out not to be set ("false") an empty string is returned anyway.

```
stuff = CASE
stuff.key.field = layout
stuff.if.isTrue.field = header
stuff.stdWrap.wrap = |<BR>

stuff.default = TEXT
stuff.default {
  ....
}
stuff.1 = TEXT
stuff.1 {
  ....
}
```

LOAD_REGISTER:

This provides a way to load the array $GLOBALS["TSFE"]->register[] with values. It doesn't return anything! The usefullness of this is, that some predefined configurations (like the page-content) can be used in various places but use different values as the values of the register can change during the page-rendering.

Property:	Data type:	Description:	Default:
Array... *[fieldname]*	string /stdWrap	**Example:** (This sets "contentWidth", "label" and "head") `page.27 = LOAD_REGISTER` `page.27 {` `  contentWidth = 500` `  label.field = header` `  head = some text` ` head.wrap = <B>	</B>` `}`	

[tsref:(cObject).LOAD_REGISTER]

Example:

RESTORE_REGISTER:

This unsets the latest changes in the register-array as set by LOAD_REGISTER.

Internally this works like a stack there the original register is saved when LOAD_REGISTER is called. Then a RESTORE_REGISTER cObject is called the last element is pulled of that stack the register is replaced with it.

RESTORE_REGISTER has no properties.

FORM:

This provides a way to create forms

```
textarea:  Label | [* = required][fieldname =] textarea[,cols,rows,"wrap= [eg. "OFF"]"] | [defaultdata]
| Special evaluation configuration (see note below)
```

```
input:          Label | [* = required][fieldname =] input[,size,max] | [defaultdata] | Special
evaluation configuration (see note below)
password:  Label | [* = required][fieldname =] input[,size,max] | [defaultdata]
file:      Label | [* = required][fieldname (*1)=] file[,size]
check:          Label | [fieldname =]check | [checked=1]
select:         Label | [* = required][fieldname =]select[,size (int/"auto"), "m"=multiple] | label
[=value] , ...
radio:          Label | [* = required][fieldname =]radio | label [=value] , ...
hidden:         |[fieldname =]hidden | value
submit:         Label |[fieldname =]submit | Caption
label:          Label | label | Label value
property:       [Internal, see below]
-----------------
```

Preselected item with type "select" and "radio":

This is an example, where "Brown" is the preselected item of a selectorbox:

```
Haircolor: | *haircolor=select| Blue=blue , Red=red , *Brown=brown
```

You can enter multiple items to be preselected by placing a asterisk in front of each preselected item.

Property override:

This can be done with the following properties from the table below:

type, locationData, goodMess, badMess, emailMess

syntax:

```
|[property] =property | value
```

*(*1) (fieldname for files)*

In order for files to be attached the mails, you must use the fieldnames:

attachment, attachment1, ... , attachment10

Correct return-email:

In order for the mails to be attached with the email address of the people that submits the mails, please use the fieldname "email", eg: *Email: | *email=input |*

Special evaluation

By prefixing a "*" before the fieldname of most types you can have the value of the field required. The check is done in JavaScript; It will only submit the form if this field is filled in.

Alternatively you can evaluate a field value against a regular expression or as an email address for certain types (textarea, password, input).

This is done by specifying the "Special evaluation configuration" for those types as part 4 in the configuration line (see examples above).

The special evaluation types are divided by a semicolon (":").

The first part defines the evaluation **keyword**. Current options are "EREG" (for regular expression) and "EMAIL" (for evaluation to an email address).

If the "EREG" keyword is specified the 2nd and 3rd parts are error message and regular expression respectively.

Examples:

```
Your address: | address=textarea,40,10 |  | EREG : You can enter only characters A to Z : ^[a-zA-Z]*$
Your email: | *email=input |  | EMAIL
```

Property:	Data type:	Description:	Default:
data	string/->stdWrap	This is the data that sets up the form. See above. "||" can be used instead of linebreaks	

Property:	Data type:	Description:	Default:

dataArray	*[array of form elements]*	This is an alternative way to define the form-fields. Instead of using the syntax with vertical separator bars suggested by the .data property, you can define the elements in regular TypoScript style arrays.
		.dataArray is *added* to the input in .data if any.
		Every entry in the dataArray is numeric and has three main properties, *label*, *type*, *value* and *required*. 'label' and 'value' has stdWrap properties. There is an alternative property to .value, which is .valueArray. This is also an array in the same style with numeric entries which has properties *label*, *value* and *selected*. 'label' has stdWrap properties.
		
		Example:
		<pre>dataArray {
		  10.label = Name:
		  10.type = name=input
		  10.value = [Enter name]
		  10.required = 1
		  20.label = Eyecolor
		  20.type = eyecolor=select
		  20.valueArray {
		    10.label = Blue
		    10.value = 1
		    20.label = Red
		    20.value = 2
		    20.selected = 1
		  }
		  40.type = submit=submit
		  40.value = Submit
		}</pre>
		This is the same as this line in the .data property:
		
		<pre>Name:
			*name=input	[Enter name]
				Eyecolor:
					eyecolor=select	Blue=1, *Red=2
						| submit=submit
							Submit</pre>
							Why do it this way? Good question, but doing it this way has a tremendous advantage because labels are all separated from the codes. In addition it's much easier to pull out or insert new elements in the form. Inserting an email-field after the name field would be like this:
							<pre>dataArray {
							  15.label = Email:
							  15.type = input
							  15.value = <u>your@email.com</u>
							  15.specialEval = EMAIL
							}</pre>
							Or translating the form to danish (setting config.language to 'dk'):
							
							<pre>dataArray {
							  10.label.lang.dk = Navn:
							  10.value.lang.dk = [Indtast dit navn]
							  20.label.lang.dk = Øjenfarve
							  20.valueArray {
							    10.label.lang.dk = Blå
							    20.label.lang.dk = Rød
							  }
							  40.value.lang.dk = Send
							}</pre>	

radioWrap	->stdWrap	Wraps the labels for radiobuttons	
type	int	Type (action="" of the form): **Integer:** this is regarded to be a page in Typo3 **String:** this is regarded to be a normal url (eg. "formmail.php" or "fe_tce_db.php") **Empty;** the current page is chosen. **NOTE:** If type is integer/empty the form will be submitted to a page in Typo3 and if this page has a value for target/no_cache, then this will be used instead of the default target/no_cache below. **NOTE:** If the redirect-value is set, the redirect-target overriddens the target set by the action-url **NOTE:** May be overridden by the property override feature of the formdata (see above)	
target	target	Default target of the form.	
method	form-method	**Example:** GET	POST

Property:	Data type:	Description:	Default:
no_cache	string	Default no_cache-option	
noValueInsert	boolean	By default values that are submitted to the same page (and thereby same form, eg. at searchforms) are re-inserted in the form instead of any default-data that might be set up. This, however, applies ONLY if the "no_cache=1" is set! (a page being cached may not include user-specific defaults in the fields of course...) If you set this flag, "noValueInsert", the content will always be the default content.	
compensateFieldWidth	double	Overriding option to the config-value of the same name. See "CONFIG" above.	
locationData	boolean / string	If this value is true, then a hidden-field called "locationData" is added to the form. This field wil be loaded with a value like this: [page id]:[current record table]:[current record id] For example, if a formfield is inserted on page with uid = "100", as a page-content item from the table "tt_content" with id "120", then the value would be "100:tt_content:120". The value is use by eg. the cObject SEARCHRESULT. If the value $GLOBALS["HTTP_POST_VARS"]["locationData"] is detected here, the search is done as if it was performed on this page! This is very usefull if you want a search functionality implemented on a page with the "stype" field set to "L1" which means that the search is carried out from the first level in the rootline. Suppose you want the search to submit to a dedicated searchpage where ever. This page will then know - because of locationData - that the search was submittet from another place on the website. If "locationData" is not only true but also set to "HTTP_POST_VARS" then the value will insert the content of $GLOBALS["HTTP_POST_VARS"]["locationData"] instead of the true location data of the page. This should be done with search-fields as this will carry the initial searching start point with. **NOTE:** May be overridden by the property override feature of the formdata (see above)	
redirect	string /stdWrap	url to redirect to (generates the hiddenfield "redirect") **Integer:** this is regarded to be a page in Typo3 **String:** this is regarded to be a normal url **Empty;** the current page is chosen. **NOTE:** If this value is set the target of this overriddes the target of the "type".	
recipient	(list of) string/stdWrap	email to formmail content to (generates the hiddenfield "recipient")	No email
goodMess	string	Message for the formevaluation function in case of correctly filled form. **NOTE:** May be overridden by the property override feature of the formdata (see above)	No message
badMess	string	Prefixed Message for the formevaluation in case of missing required fields. This message is shown above the list of fields. **NOTE:** May be overridden by the property override feature of the formdata (see above)	No message
emailMess	string	Message if a field evaluated to be an email adresse did not validate. **NOTE:** May be overridden by the property override feature of the formdata (see above)	
image	->IMAGE (cObject)	If this is a valid image the submitbutton is rendered as this image!! **NOTE:** CurrentValue is set to the caption-label before generating the image.	
layout	string	This defines how the label and the field are placed towards each other. **Example:** This substitutes the "###FIELD###" with the field data and the "###LABEL###' with labeldata. `<tr><td>###FIELD###</td><td> ###LABEL###</td></tr>` You can also use the marker ###COMMENT### which is ALSO the label value inserted, but wrapped in .commentWrap stdWrap-properties (see below)	
fieldWrap	->stdWrap	Field: Wraps the fields	
labelWrap	->stdWrap	Labels: Wraps the label	

Property:	Data type:	Description:	Default:
commentWrap	->stdWrap	Comments: Wrap for comments IF you use ###COMMENT###	
REQ	boolean	Defines if required-fields should be checked and marked up	
REQ.fieldWrap	->stdWrap	Field: Wraps the fields, but for reuired fields	the "fieldWrap"-property
REQ.labelWrap	->stdWrap	Labels: Wraps the label, but for reuired fields	the "labelWrap"-property
REQ.layout	string	The same as "layout" above, but for reuired fields	the "layout"-property
COMMENT.layout	string	Alternative layout for comments.	the "layout"-property
CHECK.layout	string	Alternative layout for checkboxes	the "layout"-property
RADIO.layout	string	Alternative layout for radiobuttons	the "layout"-property
LABEL.layout	string	Alternative layout for label types	the "layout"-property
stdWrap	->stdWrap	Wraps the hole form (before formtags is added)	

hiddenFields	[array of cObject]	Used to set hiddenFields from TS.
		
		Example:
		`hiddenFields.pid = TEXT`
		`hiddenFields.pid.value = 2`
		
		This makes a hidden-field with the name "pid" and value "2".	

params	form-element tag parameters	Extra parameters to form elements
		
		Example:
		`params = style="width:200px;"`
		`params.textarea = style="width:300px;"`
		`params.check =`
		
		This sets the default to 200 px width, but excludes check-boxes and sets textareas to 300.	

wrapFieldName	wrap	This wraps the fieldnames before they are applied to the form-field tags. **Example:** If value is *tx_myextension[input][	]* then the fieldname "email" would be wrapped to this value: *tx_myextension[input][email]*	
arrayReturnMode	boolean	If set, the <form> tags and the form content will be returned in an array as separate elements including other pratical values. This mode is for use in extensions where the array return value can be more useful.	
accessibility	boolean	If set, then the form will be compliant with accessibility guidelines. This includes: ● label string will be wrapped in <label for="[fieldname]"> ... </label> ● All form elements will have an id-attribute carrying the same value as the name-attribute (if you need unique ID-numbers you can use "wrapFieldName" to prefix all field names with a string of some kind).	

[tsref:(cObject).FORM]

Example: Login

In order to creating a loginform supply these fields:

"username" = username

"userident" = password

"login_status" = "logout" for logout, "login" for login.

If you insert "<!--###USERNAME###-->" somewhere in your document this will be substituted by the username if a user is logged in!

If you want the login-form to change into a logout form you should use conditions to do this. See this TS-example (extract from the static_template *"styles.content (default)"*):

```
  # loginform
styles.content.loginform {
  data = Username:|*username=input || Password:|*userident=password
}
[usergroup = *]
styles.content.loginform.data = Username: <!--###USERNAME###-->  || |submit=submit| Logout
[global]
```

(shortend a bit...)

Example: Mailform
This creates a simple mailform (this is not TypoScript, but the setup code that you should put directly into the "bodytext"-field of a pagecontent record of the type "FORMMAIL":

```
Name: | *the name = input | Enter your name here
Email: | *email=input |
Like TV: | tv=check |
|formtype_mail = submit | Send this!

|html_enabled=hidden | 1
|subject=hidden| This is the subject
|recipient_copy=hidden | copy@email.com
|auto_respond_msg=hidden|  Hello / This is an autoresponse. //We have received your mail.
|tv=hidden | 0
```

- "NAME" is required (the asterisk, *) and the fieldname will be "the_name". A default value is set ("Enter your...")

- "Email" is also required, the name will be "email" (which it should always be for the address to shown up properly in the real email!) and theres no default value here.

- "Like TV" is a checkbox. Default is "unchecked".

- "formtype_mail" is the name of the submit-button. And it should be if you use the build-in formmail of Typo3. Then this var makes Typo3 react on the input at interpret it as formmail-input!

- "html_enabled" will let the mail be rendered in nice HTML

- "subject": Enter the subject of your mail

- "recipient_copy" : A copy is sent to this mail-address. You may supply more addresses by separating with a comma ",". The mail sent to recicipent_copy is a the same, but a separate message from the one sent to the 'recipient' and furthermore the copy-mail is sent only if the 'recipient' mail is sent.

- "auto_respond_msg": This is a autoresponder message. This is sent if the email of the "submitter" is known (field: "email"). The value of this is the message broken up in to lines by a slash "/". Each slash is a new line in the email. The first line is used for the subject.

- "tv" (again, but hidden). Repeating this field may be smart as the value "tv" is normally NOT submitted with the value "false" if not checked. Inserting this line will secure a default value for "tv".

SEARCHRESULT:
Searchwords are loaded into the register in a form ready for linking to pages:

Example:
register:SWORD_PARAMS = '&sword_list[]=word1&sword_list[]=word2 .....'

See typolink for more info!

SEARCHRESULT returns results only from pages with of doktype "Standard" (1), "Advanced" (2) and "Not in menu" (5)

Property:	Data type:	Description:	Default:
allowedCols	string	List (separated by ":") of allowed table-cols. **Example:** pages.title:tt_content.bodytext	

layout	string	This defines how the search content is shown.
		
		Example:
		This substitutes the following fields:
		`###RANGELOW###:   The low result range, eg. "1"`
		`###RANGEHIGH###:  The high result range, eg. "10"`
		`###TOTAL###:         The total results`
		`###RESULT###:    The result itself`
		`###NEXT###:      The next-button`
		`###PREV###:      The prev-button`	

next	cObject	This cObject will be wrapped by a link to the next searchresult. This is the code substituting the "###NEXT###"-mark	

Property:	Data type:	Description:	Default:
prev	cObject	This cObject will be wrapped by a link to the prev searchresult. This is the code substituting the "###PREV###"-mark	
target	target	target til next/prev links!	
range	int	The number of results at a time!	20
renderObj	cObject	the cObject to render the searchresults $cObj->data array is set to the resulting record from the search. Please note, that in all fields are named [tablename]_[fieldnam]. Thus the pagetitle is in the field "pages_title". Apart from this, these fields from the pages-table are also present: `uid`	
renderWrap	wrap		
resultObj	cObject	the cObject prepended in the search results returns rows	
noResultObj	cObject	the cObject used if the search results in no rows.	
noOrderBy	boolean	If this is set, the result is NOT sorted after lastUpdated, tstamp for the pages-table.	
wrap	wrap	Wrap the whole content...	
stdWrap	->stdWrap	Wrap the whole content...	
addExtUrlsAndShort Cuts	boolean	If set, then the doktypes 3 and 4 (External URLS and Shortcuts) are added to the doktypes being searched. However at this point in time, no pages will be select if they do not have at least one tt_content record on them! That is because the pages and tt_content (or other) table is joined. So there must at least one occurance of a tt_content element on a External URL / Shortcut page for them to show up.	
languageField.[2nd table]	string	Setting a field name to filter language on. This works like the "languageField" setting in ->select **Example:** languageField.tt_content = sys_language_uid	

[tsref:(cObject).SEARCHRESULT]

NOTE: "sword" and "scols" MUST be set in order for the search to be engaged.

var "sword" = search word(s)

var "scols" = search columns separated by ":". Eg: pages.title:pages.keywords:tt_content.bodytext

var "stype" = the starting point of the search: false = current page, L-2 = page before currentPage, L-1 = current page, L0 = rootlevel, L1 = from first level, L2 = from second level

var $GLOBALS["HTTP_POST_VARS"]["locationData"]: If this is set, the search is done as was it from another page in the website given by the value of "locationData" here. See the description at the cObject "FORMS".

Only if the page locationData is pointing to, is inside the real rootLine of the site, the search will take this into account.

internal:

var "scount": If this is set this is used as the searchCount - the total rows in the search. This way we don't need to reconstruct this number!

var "spointer": This points to the start-record in the search.

LATER:

var "alldomains" :   boolean: If set the search will proceed into other domains

var "allsites" :   boolean: If set the search will proceed into other sites (defined by the "root" setting of an active template.)

var "depth":   The depth

Search syntax
When you search, you can use three operatortypes

- AND: "+", "and" (UK), "og" (DK)

- OR: "or" (UK), "eller" (DK)

- NOT: "-", "not" (UK), "uden" (DK)

Default operator is AND. If you encapsulate words in "" they are searched for as a whole string. The search is case

insensitive and matches parts of words also.

Examples:

1. *menu backend* - will find pages with both 'menu' and 'backend'.

2. *"menu backend"* - will find pages with the phrase "menu backend".

3. *menu or backend* - will find pages with either 'menu' or 'backend'

4. *menu or backend not content* - will find pages with either 'menu' or 'backend' but not 'content'

Queries to the examples:

In this case "pagecontent" is chosen as the fields to search. That includes *tt_content.header*, *tt_content.bodytext* and *tt_content.imagecaption.*

Prefixed to these queries is this:

SELECT pages.title AS pages_title, pages.subtitle AS pages_subtitle, pages.keywords AS pages_keywords, pages.description AS pages_description, pages.uid, tt_content.header AS tt_content_header, tt_content.bodytext AS tt_content_bodytext, tt_content.imagecaption AS tt_content_imagecaption FROM pages, tt_content WHERE(tt_content.pid=pages.uid) AND (pages.uid IN (2,5,6,20,21,22,29,30,31,3,4,8,9,16,1) AND pages.doktype in (1,2,5) AND pages.no_search=0 AND NOT tt_content.deleted AND NOT tt_content.hidden AND (tt_content.starttime<=985792797) AND (tt_content.endtime=0 OR tt_content.endtime>985792797) AND tt_content.fe_group IN (0,-1) AND NOT pages.deleted AND NOT pages.hidden AND (pages.starttime<=985792797) AND (pages.endtime=0 OR pages.endtime>985792797) AND pages.fe_group IN (0,-1)) ...

The part "... pages.uid IN (2,5,6,20,21,22,29,30,31,3,4,8,9,16,1)... " is a list of pages-uid's to search. This list is based on the page-ids in the website-branch of the pagetree and confines the search to that branch and not the whole page-table.

1. ... AND ((tt_content.header LIKE '%menu%' OR tt_content.bodytext LIKE '%menu%' OR tt_content.imagecaption LIKE '%menu%') AND (tt_content.header LIKE '%backend%' OR tt_content.bodytext LIKE '%backend%' OR tt_content.imagecaption LIKE '%backend%')) GROUP BY pages.uid

2. ... AND ((tt_content.header LIKE '%menu backend%' OR tt_content.bodytext LIKE '%menu backend%' OR tt_content.imagecaption LIKE '%menu backend%')) GROUP BY pages.uid

3. ... AND ((tt_content.header LIKE '%menu%' OR tt_content.bodytext LIKE '%menu%' OR tt_content.imagecaption LIKE '%menu%') OR (tt_content.header LIKE '%backend%' OR tt_content.bodytext LIKE '%backend%' OR tt_content.imagecaption LIKE '%backend%')) GROUP BY pages.uid

4. ... AND ((tt_content.header LIKE '%menu%' OR tt_content.bodytext LIKE '%menu%' OR tt_content.imagecaption LIKE '%menu%') OR (tt_content.header LIKE '%backend%' OR tt_content.bodytext LIKE '%backend%' OR tt_content.imagecaption LIKE '%backend%') AND NOT (tt_content.header LIKE '%content%' OR tt_content.bodytext LIKE '%content%' OR tt_content.imagecaption LIKE '%content%')) GROUP BY pages.uid

Notice that upper and lowercase does not matter. Also 'menu' as searchword will find 'menu', 'menus', 'menuitems' etc.

USER and USER_INT:

This calls either a PHP-function or a method in a class. This is very useful if you want to incorporate you own data processing or content.

Basically this a userdefined cObject, because it's just a call to a function or method you control!

An important thing to know is that if you call a method in a class (which is of course instantiated as an object) the internal variable 'cObj' of that class is set with a *reference* to the parent cObj. See the example_callfunction.php file for an example of how this may be usefull for you. Basically it offers you an API of functions which are more or less relevant for you. Refer to the "Include PHP scripts" section in this document.

It's a little like the PHP_SCRIPT concept but this is somehow cleaner, because it's a call to a function previously defined and not an inclusion of a PHP-script file. So this is recommended.

If you create this object as USER_INT, it'll be rendered non-cached, outside the main page-rendering. See the PHP_SCRIPT_INT for details as this is the same concept used there.

Property:	Data type:	Description:	Default:

userFunc	function-name	The name of the function. If you specify the name with a '->' in, it's intepreted as a call to a method in a class.
		Two parameters are sent: A content variable (which is empty in this case, but not when used from stdWrap function .postUserFunc and .preUserFunc) and the second parameter is an array with the properties of this cObject if any.
		
		Example:
		This TypoScript will display all content element headers of a page in reversed order. Please take a look in media/scripts/example_callfunction.php!! (Also demonstrated on the testsite, page
		
		`page = PAGE`
		`page.typeNum=0`
		`includeLibs.something =`
		`media/scripts/example_callfunction.php`
		
		`page.30 = USER`
		`page.30 {`
		`  userFunc = user_various->listContentRecordsOnPage`
		`  reverseOrder = 1`
		`}`	

Property:	Data type:	Description:	Default:
includeLibs	*list of* resource	**(This property applies only if the object is created as USER_INT)** This is a comma-separated list of resources that are included as PHP-scripts (with include_once() function) if this script is included. This is possible to do because any include-files will be known before the scripts are included. That's not the case with the regular PHP_SCRIPT cObject.	

[tsref:(cObject).USER/(cObject).USER_INT]

PHP_SCRIPT:

This includes a PHP-script. You should not name these script ".php" but rather ".inc" as it's meant to be included and not executed on it's own.

NOTE: This options is ignored if $TYPO3_CONF_VARS["FE"]["noPHPscriptInclude"]=1; is set in localconf.php.

Property:	Data type:	Description:	Default:
file	resource	File that will be included. This file must be valid PHP-code! It's included with "include()"; **Directions:** **1) All content must be put into $content.** No output must be echo'ed out! 2) Call $GLOBALS["TSFE"]->set_no_cache(), if you want to disable caching of the page. Set this during development! And set it, if the content you create may not be cached. **NOTE:** If you have a parsing error in your include script the $GLOBALS ["TSFE"]->set_no_cache() function is NOT executed and thereby does not disable caching. Upon a parse-error you must manually clear the page-cache after you have corrected your error! 3) the array $conf contains the configuration for the PHP_SCRIPT cObject. Try debug($conf) to see the content printed out for debugging! *See later in this manual for an introduction to writing your own PHP include-scripts.*	

[tsref:(cObject).PHP_SCRIPT]

PHP_SCRIPT_INT:
(see PHP_SCRIPT)

Property:	Data type:	Description:	Default:
file	resource	File that will be included. This file must be valid PHP-code! It's included with "include()"; **Purpose:** This basically works like PHP_SCRIPT. But the vital difference is that inserting a PHP_SCRIPT_**INT** (internal opposed to external, see below) merely inserts a divider-string in the code and then serializes the current cObj and puts it in the $GLOBALS["TSFE"]->config["INTincScript"]-array. This array is saved with the cached page-content. Now, the point is, that including a script like this lets you avoid disabling pagecaching. The reason is that the cached page contains the divider string and when a "static" page is fetched from cache, it's divided by that string and the dynamic content object is inserted. This is the compromise option of all three PHP_SCRIPT-cObjects, because the page-data is all cached, but still the pagegen.php script is included, which initializes all the classes, objects and so. What you gain here is an environment for your script almost exactly the same as PHP_SCRIPT because your script is called from inside a class tslib_cObj object. You can work with all functions of the tslib_cObj-class. But still all the "static" pagecontent is only generated once, cached and only your script is dynamically rendered. **Rules:** - calls to $GLOBALS["TSFE"]->set_no_cache() and $GLOBALS["TSFE"]->set_cache_timeout_default() makes no sense in this situation. - parsing errors does not interfere with caching - Be aware that certain global variables may not be set as usual and be available as usual when working in this mode. Most scripts should work out-of-the-box with this option though. - Dependence and use of LOAD_REGISTER is fragile because the PHP_SCRIPT_INT is not rendered until *after* the cached content and due to this changed order of events, use of LOAD_REGISTER may not work. - You can not nest PHP_SCRIPT_INT and PHP_SCRIPT_EXT in PHP_SCRIPT_INT. You may nest PHP_SCRIPT cObjects though.	

Property:	Data type:	Description:	Default:
includeLibs	*list of* resource	This is a comma-separated list of resources that are included as PHP-scripts (with include_once() function) if this script is included. This is possible to do because any include-files will be known before the scripts are included. That's not the case with the regular PHP_SCRIPT cObject.	

[tsref:(cObject).PHP_SCRIPT_INT]

PHP_SCRIPT_EXT:
(see PHP_SCRIPT)

Property:	Data type:	Description:	Default:
file	resource	File that will be included. This file must be valid PHP-code! It's included with "include()"; **Purpose:** This works like PHP_SCRIPT_INT, because a divider string is also inserted in the content for this kind of include-script. But the difference is that the content is divided as the very last thing before it's output to the browser. This basically means that PHP_SCRIPT_**EXT** (external, because it's included in the global space in index_ts.php file!!) can output data directly with echo-statements! This is a very "raw" version of PHP_SCRIPT because it's not included from inside an object and you have only very few standard functions from Typo3 to call. This is the fastest option of all three PHP_SCRIPT-cObjects, because the page-data is all cached and your dynamic content is generated by a raw php-script **Rules:** - All content can be either 1) echo'ed out directly, or 2) returned in $content. - calls to $GLOBALS["TSFE"]->set_no_cache() and $GLOBALS["TSFE"]->set_cache_timeout_default() makes no sense in this situation. - parsing errors does not interfere with caching - In the global name-space, the array $REC contains the current record when the file was "inserted" on the page, and $CONF-array contains the configuration for the script. - Don't mess with the global vars named $EXTiS_*	
includeLibs	*list of* resource	This is a comma-separated list of resources that are included as PHP-scripts (with include_once() function) if this script is included. This is possible to do because any include-files will be known before the scripts are included. That's not the case with the regular PHP_SCRIPT cObject.	

[tsref:(cObject).PHP_SCRIPT_EXT]

TEMPLATE:

Property:	Data type:	Description:	Default:
template	cObject	This must be loaded with the template-code. If not the object returns nothing.	

Property:	Data type:	Description:	Default:

subparts	*Array... of* cObject	This is an array of subpart-markers (case-sensitive). A subpart is defined by two markers in the template. The markers must be wrapped by "###" on both sides. You may insert the subpart-markers inside HTML-comment-tags!!
		
		Example:
		`subparts {`
		`  HELLO = TEXT`
		`  HELLO.value = En subpart er blevet erstattet!!`
		`}`
		
		In the templates:
		`<!--  start of subpart: ###HELLO### -->`
		`This is the HTML.code, that will be loaded in the`
		`register and replaced with the result...`
		`<!-- end ###HELLO### -->`
		
		NOTE:
		Before the content-objects of each subpart is generated, all subparts in the array are extracted and loaded into the register so that you can load them from there later on.
		The register-key for each subparts code is "SUBPART_[theSubpartkey]".
		In addition the current-value is loaded with the content of each subpart just before the cObject for the subpart is parsed. That makes it quite easy to load the subpart of the cObject (eg: ".current=1")
		Eg. this subpart above has the register-key "SUBPART_HELLO".
		This is valid ONLY if the property .nonCachedSubst is not set! (see below)	

relPathPrefix	*string / properties*	Finds all relative references (eg. to images or stylesheets) and prefixes this value.
		If you specify properties these will match HTML tags (lowercase) and specify alternative paths for them. See example below.
		If the property is named "style" it will set alternative path for the "url()" wrapper that may be in <style> sections.
		
		Example:
		`page.10 = TEMPLATE`
		`page.10 {`
		`  template = FILE`
		`  template.file = fileadmin/template.html`
		`  relPathPrefix = fileadmin/`
		`  relPathPrefix.img = fileadmin/img/`
		`}`
		
		Inthis example all relative paths found are prefixed "fileadmin/" unless it was the src attribute of an img tag in which case the path prefixed is "fileadmin/img/"	

marks	*Array... of* cObject	This is an array of marks-markers (case-sensitive). A mark is defined by one markers in the template. The marker must be wrapped by "###" on both sides. Opposite to subparts, you may NOT insert the subpart-markers inside HTML-comment-tags! (They will not be removed). Marks are substituted bya str_replace-function. The subparts loaded in the register is available also to the cObjects of markers (only if .nonCachedSubst is not set!).	
wraps	*Array... of* cObject	This is an array of wraps-markers (case-sensitive). This is shown best by an example: **Example:** `subparts {` `  MYLINK = TEXT` ` MYLINK.value = <A href="#">	</A>` `}`  In the template: `This is <!--###MYLINK###-->a link to my<!--` `###MYLINK###--> page!` In this example the MYLINK subpart will be substituted by the wrap which is the content returned by the MYLINK cObject.	
workOnSubpart	string	This is an optional definition of a subpart, that we decide to work on. In other words; if you define this value that subpart is extracted from the template and is the basis for this whole templateobject.	
markerWrap	wrap	This is the wrap the markers is wrapped with. The default value is ###	### resulting in the markers to be presented as ###[marker_key]###. Any whitespace around the wrap-items is stripped before they are set around the marker_key.	### | ###

Property:	Data type:	Description:	Default:
substMarksSeparately	boolean	If set, then marks are substituted in the content AFTER the substitution of subparts and wraps. Normally marks are not substituted inside of subparts and wraps when you are using the default cached mode of the TEMPLATE cObject. That is a problem if you have marks inside of subparts! But setting this flag will make the marker-substitution a non-cached, subsequent process. Another solution is to turn of caching, see below.	
nonCachedSubst	boolean	If set, then the substitution mode of this cObject is totally different. Normally the raw template is read and divided into the sections denoted by the marks, subparts and wraps keys. The good thing is high speed, because this "pre-parsed" template is cached. The bad thing is that templates that depends on incremental substition (where the order of substition is important) will not work so well. By setting this flag, markers are first substituted by str_replace in the template - one by one. Then the subparts are substituted one by one. And finally the wraps one by one. Obviously you loose the ability to refer to other parts in the template with the register-keys as described above.	

[tsref:(cObject).TEMPLATE]

Example:

```
page.10 = TEMPLATE
page.10 {
  template = FILE
  template.file = fileadmin/test.tmpl
  subparts {
    HELLO = TEXT
    HELLO.value = This is the replaced subpart-code
  }
  marks {
    Testmark = TEXT
    Testmark.value = This is replacing a simple marker in the HTML-code
  }
  workOnSubpart = DOCUMENT
}
```

In this example a template named test.tmpl is loaded.

MULTIMEDIA:

Property:	Data type:	Description:	Default:
file	resource /stdWrap	The multimedia file. Types are: txt, html, htm: Inserted directly class: Java-applet swf: Flash animation swa, dcr: ShockWave Animation wav,au: Sound avi,mov,asf,mpg,wmv: Movies (AVI, QuickTime, MPEG4)	

params	string /stdWrap	This is parameters for the multimedia-objects. Use this to enter stuff like with and height:
		
		Example:
		`width=200`
		`height=300`
		
		... will generate a tag like '<embed .... width="200" height="300">'
		`height=`
		
		An empty string will remove the parameter from the embed-tag	

stdWrap	->stdWrap		

[tsref:(cObject).MULTIMEDIA]

au, wav:

width of control   (default 200)

height of control   (default 16)

loop = true / false

autostart = true/false

avi,mov,asf,mpg,wmv:

width of control   (default 200)

height of control   (default 200)

autostart = true/false   (not "mov", see below for example)

swf,swa,dcr:

width   (browserdefault approx. 200)

height   (browserdefault approx. 200)

quality   (default "high")

class:

width   (default 200)

height   (default 200)

QuickTime (mov) example:

```
WIDTH=256
HEIGHT=208
autoplay=TRUE
CONTROLLER=true
LOOP=false
PLUGINSPAGE= http://www.apple.com/quicktime/
```

EDITPANEL:

This content object is inserted only if a backend user is logged in and if that user has enabled "Display Edit Icons" in the front end Admin Panel. If the edit panel is inserted, page caching is disabled as the edit panel offers editing feature only available for backend users.

The edit panel inserts icons for moving, editing, deleting, hiding and creating records.

Property:	Data type:	Description:	Default:
label	string	Title for the panel. You can insert the record title with %s **Example:** Section: <B>%s</B>	
allow	string	Define which functions are accessible. Further this list may be reduced, if the BE_USER does not have permission to perform the action Values should be listed separated by comma. This is the options you can choose between: toolbar,edit,new,delete,move,hide (toolbar is a general list of icons regarding the page, so use this for pagerecords only)	
newRecordFromTable	string, tablename	Will display a panel for creation of new element (in the top of list) on the page from that table.	
line	boolean / int	If set, a black line will appear after the panel. This value will indicate the distance from the black line to the panel	
edit.displayRecord	boolean	If set, then the record edited is display above the editing form.	
onlyCurrentPid	boolean	If set, only records with a pid matching the current id (TSFE->id) will be shown with the panel.	
previewBorder	boolean / int	If set, the hidden/starttime/endtime/fe_user elements which are preview will have a border around. The integer value denotes the thickness of the border	

[tsref:(cObject).EDITPANEL]

GIFBUILDER

GIFBUILDER:

GIFBUILDER is a object, which is used in many situations for creating gif-files. Anywhere the ->GIFBUILDER object is mentioned, this is the properties that apply.

NOTE (+calc):

When ever the "+calc"-function is added to a value in the data type of the properties underneath, you can use the dimensions of TEXT and IMAGE-objects from the GifBuilderObj-array. This is done by inserting a tag like this: "[10.w]" or "[10.h]", where "10" is the GifBuilderObj-number in the array and "w"/"h" signifies either width or height of the object.

See this example (cut from "styles.content (default)"):

```
styles.header.gfx1 = IMAGE
styles.header.gfx1 {
  wrap = {$styles.header.gfx1.wrap}
  file = GIFBUILDER
  file {
    XY = [10.w]+10 ,{$styles.header.gfx1.itemH}
    backColor = {$styles.header.gfx1.bgCol}
    reduceColors = {$styles.header.gfx1.reduceColors}
    10 = TEXT
    10 {
      text.current = 1
      text.crop = {$styles.header.gfx1.maxChars}
      fontSize = {$styles.header.gfx1.fontSize}
      fontFile = {$styles.header.gfx1.file.fontFile}
      fontColor = {$styles.header.gfx1.fontColor}
      offset = {$styles.header.gfx1.fontOffset}
    }
  }
}
```

As you see, the gif-image has a width defined as the width of the text printed onto it + 10 pixels. The height is fixed by the value of the constant {$styles.header.gfx1.itemH}

The "_GIFBUILDER" Top Level Object

You can configure some global settings for GIFBUILDER by a top level object named "_GIFBUILDER". One of the available properties of the global GIFBUILDER configuration is "charRangeMap".

.charRangeMap

By this property you can globally configure mapping of font files for certain character ranges. For instance you might need GIFBUILDER to produce gif files with a certain font for latin characters while you need to use another true type font for Japanese glyphs. So what you need is to specify the usage of another font file when characters fall into another range of Unicode values.

In the GIFBUILDER object this is possible with the "splitRendering" option but if you have hundreds of GIFBUILDER objects around your site it is not very efficient to add 5-10 lines of configuration for each time you render text. Therefore this global setting allows you to match the basename of the main font face with an alternative font.

Property:	Data type:	Description:	Default:

[array]	string	Basename of font file to match for this configuration. Notice that only the *filename* of the font file is used - the path is stripped off. This is done to make matching easier and avoid problems when font files might move to other locations in extensions etc.
		
		So if you use the font file "EXT:myext/fonts/arial.ttf" or "t3lib/fonts/arial.ttf" both of them will match with this configuration.
		
		The key:
		The value of the array key will be the key used when forcing the configuration into "splitRendering" configuration of the individual GIFBUILDER objects. In the example below the key is "123". Notice; If the key is already found in the local GIFBUILDER configuration the content of that key is respected and not overridden. Thus you can make local configurations which override the global setting.
		
		Example:
		`_GIFBUILDER.charRangeMap {`
		`  123 = arial.ttf`
		`....`	

Property:	Data type:	Description:	Default:
[array].charMapConfig	TEXT / splitRendering. [array] configuration	splitRendering configuration to set. See GIFBUILDER TEXT object for details. **Example:** ```\n_GIFBUILDER.charRangeMap {\n 123 = arial.ttf\n 123 {\n charMapConfig {\n fontFile = t3lib/fonts/vera.ttf\n value = -65\n fontSize = 45\n }\n fontSizeMultiplicator = 2.3\n }\n}\n``` This example configuration shows that GIFBUILDER TEXT objects with font faces matching "arial.ttf" will have a splitConfiguration that uses "t3lib/fonts/vera.ttf" for all characters that fall below/equal to 65 in Unicode value.	
[array].fontSizeMultiplicator	double	If set, this will take the font size of the TEXT GIFBUILDER object and multiply with this amount (xx.xx) and override the "fontSize" property inside "charMapConfig".	
[array].pixelSpaceFontSizeRef	double	If set, this will multiply the four [x/y]Space[Before/After] properties of split rendering with the relationship between the fontsize and this value. In other words; Since pixel space may vary depending on the font size used you can simply specify by this value at what fontsize the pixel space settings are optimized and for other fontsizes this will automatically be adjusted according to this font size. **Example:** ```\n_GIFBUILDER.charRangeMap {\n 123 = arial.ttf\n 123 {\n charMapConfig {\n fontFile = t3lib/fonts/vera.ttf\n value = 48-57\n color = green\n xSpaceBefore = 3\n xSpaceAfter = 3\n }\n pixelSpaceFontSizeRef = 24\n }\n}\n``` In this example xSpaceBefore and xSpaceAfter will be "3" when the font size is 24. If this configuration is used on a GIFBUILDER TEXT object where the font size is only 16 the spacing values will be corrected by "16/24", effectively reducing the pixelspace to "2" in that case.	

[tsref:_GIFBUILDER.charRangeMap]

Objectnames in this section:

Whenever you see a reference to anything named an "object" in this section it's a reference to a "GifBuilderObj" and not the "cObjects" from the previous section. Confusion could happen, because both "IMAGE" and "TEXT" is a object in both areas.

Property:	Data type:	Description:	Default:
1,2,3,4...	GifBuilderObj + .if (->if)	.if (->if) is a property of all gifbuilder-objects. If the property is present and NOT set, the object is NOT rendered! This corresponds to the functionallity of ".if" of the stdWrap-function.	
XY	x,y +calc	Size of the gif-file.	100,20
format	"gif" / "jpg"	Output type. "jpg"/"jpeg" = jpg-image	gif
reduceColors	posint (1-255)	Reduce the number of colors (if gif-file)	
transparentBackground	boolean	Set this flag to render the background transparent. Typo3 makes the color found at position 0,0 of the image (upper left corner) transparent. If you render text you should leave the niceText option OFF as the result with probably be more precise without the niceText antialiasing hack	

Property:	Data type:	Description:	Default:
transparentColor	HTMLColor / stdWrap	Specify a color that should be transparent **Example-values:** #ffffcc red 255,255,127 **Option:** transparentColor.closest = 1 This will allow for the closest color to be matched instead. You may need this if you image is not garanteed "clean". **NOTE:** You may experience that this doesn't work if you use reduceColors-option or render text with niceText-option.	
quality	posint (10-100)	JPG-quality (if ".format" = jpg/jpeg)	
backColor	GraphicColor +stdWrap	Background color for the gif	white
offset	x,y +calc	Offset all objects on the gif.	0,0
workArea	x,y,w,h + calc	Define the workarea on the giffile. All the GifBuilderObj's will see this as the dimensions of the gif-file regarding alignment, overlaying of images an so on. Only will TEXT-objects exceeding the boundaries of the workarea print outside this area.	
maxWidth	pixels	Maximal width of gif-file	
maxHeight	pixels	Maximal heigth of gif-file	

[tsref:->GIFBUILDER]

TEXT:

Property:	Data type:	Description:	Default:
text	stdWrap	This is text text-string on the gif-file. The item is rendered only if this string is not empty. The cObj->data-array is loaded with the page-record, if for example the GIFBUILDER-object is used by GMENU or IMGMENU	
textMaxLength	int	The maximum length of the text. This is just a natural break that prevents incidental rendering of very long texts!	100
maxWidth	pixels	Sets the maximum width in pixels, the text must be. Reduces the fontSize if the text does not fit within this width. Does not support setting alternative fontSizes in splitRendering options. (By Rene Fritz <r.fritz@colorcube.de>)	
doNotStripHTML	boolean	If set, HTML-tags in the string inserted are NOT removed. Any other way HTML-code is removed by default!	0
fontSize	posint	Font size	12
fontColor	GraphicColor	Font color	black
fontFile	resource	Font face (truetype font you can upload!!)	Nimbus (Arial-clone)
angle	degree	Rotation degrees of the text. **NOTE:** Angle is not available if spacing/wordSpacing is set.	0 Range: -90 til 90
align	align	Alignment of the text	left
offset	x,y +calc	Offset of the text	0,0
antiAlias	Boolean	FreeType antialiasing. Notice, the default mode is "on"! **Note:** This option is not available if .niceText is enabled	1
iterations	posint	How many times the text should be "printed" onto it self. This will add the effect of bold text. **Note:** This option is not available if .niceText is enabled	1
spacing	posint	Pixel-distance between letters. This may render ugly!	0
wordSpacing	posint	Pixel-distance between words.	= ".spacing"*2
hide	boolean	If this is true, the text is NOT printed. This feature may be used if you need a shadow-object to base a shadow on the text, but do not want the text to print.	0

Property:	Data type:	Description:	Default:
hideButCreateMap	boolean	If this option is set, the text will not be rendered. Shadows and emboss will, though, so don't apply these!! But this feature is also meant only to enable a text to generate the imageMap coordinates without rendering itself.	
emboss	GifBuilderObj->EMBOSS		
shadow	GifBuilderObj->SHADOW		
outline	GifBuilderObj->OUTLINE		
imgMap	->IMGMAP + stdWrap properties for "altText" and "titleText" in this case.		
niceText	boolean	This is a very popular feature that helps to render small letters much nicer than the freetype library can normally do. But it also loads the system very much! The principle of this function is to create a black/white giffile in twice or more times the size of the actual gif-file and then print the text onto this is a scaled dimension. Afterwards ImageMagick (IM) scales down the mask and masks the font color down on the original gif-file through the temporary mask. The fact that the font is actually rendered in the double size and scaled down adds a more homogenous shape to the lettes. Some fonts are more critical than others though. If you do not need the quality, then don't use the function. **Some properties:** .before = IM-params before scale .after = IM-params after scale .sharpen = sharpen-value for the mask (after scaling), integer 0-99 (this enables you to make the text crisper if it's too blurred!) .scaleFactor = scaling-factor, int 2-5	

Property:	Data type:	Description:	Default:

splitRendering.comp X
splitRendering.comp Y
splitRendering.[array]
		Split the rendering of a string into separate processes with individual configurations. By this method a certain range of characters can be rendered with another font face or size. This is very useful if you want to use separate fonts for strings where you have latin characters combined with eg. Japanese and there is a separate font file for each.
		You can also render keywords in another font/size/color.
		
		Properties:
		splitRendering.compX = *Additional pixelspace between parts, x direction*
		splitRendering.compY = *Additional pixelspace between parts, y direction*
		splitRendering.[array] = *keyword* [charRange, highlightWord]
		splitRendering.[array] {
		  fontFile = *Alternative font file for this rendering*
		  fontSize = *Alternative font size for this rendering*
		  color = *Alternative color for this rendering, works ONLY without "niceText"*
		  xSpaceBefore = *x-Space before this part*
		  xSpaceAfter = *x-Space after this part*
		  ySpaceBefore = *y-Space before this part*
		  ySpaceAfter = *y-Space after this part*
		}
		
		Keyword: charRange
		splitRendering.[array].value = Commaseparated list of character ranges (eg. "100-200") given as Unicode character numbers. The list accepts optional starting and ending points, eg. " - 200" or " 200 -" and single values, eg. "65, 66, 67"
		
		Keyword: highlightWord
		splitRendering.[array].value = Word to highlight, makes a case sensitive search for this.
		
		Limitations:
		● The pixelcompensation values are not corrected for scale factor used with niceText. Basically this means that when niceText is used, these values will have only the half effect.
		● When word spacing is used the "highlightWord" mode doesn't work.
		● The color override works only without "niceText".
		
		Example:
		`10.splitRendering.compX = 2`
		`10.splitRendering.compY = -2`
		`10.splitRendering.10 = charRange`
		`10.splitRendering.10 {`
		`  value = 200-380 , 65, 66`
		`  fontSize = 50`
		`  fontFile = t3lib/fonts/nimbus.ttf`
		`  xSpaceBefore = 30`
		`}`
		`10.splitRendering.20 = highlightWord`
		`10.splitRendering.20 {`
		`  value = TheWord`
		`  color = red`
		`}`	

[tsref:->GIFBUILDER.(GBObj).TEXT]

SHADOW:

Property:	Data type:	Description:	Default:
textObjNum	pos-int	Must point to the TEXT-object if these shadow-properties are not properties to a TEXT-object directly ("stand-alone-shadow"). Then the shadow needs to know which TEXT-object it should be a shadow of! If - on the other hand - the shadow is a property to a text-object, this property is not needed.	
offset	x,y	Shadow offset	
color	GraphicColor	Shadow color	
blur	posint (1-99)	Blurring of the shadow. Above 40 only values of 40,50,60,70,80,90 means something. **NOTE:** Unfortunately the blurring capabilities of ImageMagick is not very mature in the version 4.2.9. This is addressed in the later version 5.2.0 where a gaussian blur-function is added. BUT as we do cannot use the latest ImageMagick development yet, this is not utilized so far.	

Property:	Data type:	Description:	Default:
opacity	posint (1-100)	Opacity (transparency^-1) 100% opacity = 0% transparency). Only active with a value for blur.	
intensity	posint(0-100)	How "massive" the shadow is. This value can - if it has a high value combined with a blurred shadow - create a kind of soft-edged outline.	

[tsref:->GIFBUILDER.(GBObj).SHADOW]

EMBOSS:

Emboss is actually two shadows offset in opposite directions and with different colors as to create an effect of light casted onto an embossed text.

Property:	Data type:	Description:	Default:
textObjNum	pos-int	Must point to the TEXT-object if these shadow-properties are not properties to a TEXT-object directly ("stand-alone-shadow"). Then the shadow needs to know which TEXT-object it should be a shadow of! If - on the other hand - the shadow is a property to a text-object, this property is not needed.	
offset	x,y	Offset of the emboss	
highColor	GraphicColor	Upper border-color	
lowColor	GraphicColor	lower border-color	
blur	posint (1-99)	Blurring of the shadow. Above 40 only values of 40,50,60,70,80,90 means something.	
opacity	posint (1-100)	Opacity (transparency^-1) 100% opacity = 0% transparency). Only active with a value for blur.	
intensity	posint(0-100)	How "massive" the emboss is. This value can - if it has a high value combined with a blurred shadow - create a kind of soft-edged outline.	

[tsref:->GIFBUILDER.(GBObj).EMBOSS]

OUTLINE:

This outline normally renderes quite ugly as it's done by print 4 or 8 texts underneath the text in question. Try to use a shadow with a high intensity. That works better!

Property:	Data type:	Description:	Default:
textObjNum	pos-int	Must point to the TEXT-object if these shadow-properties are not properties to a TEXT-object directly ("stand-alone-shadow"). Then the shadow needs to know which TEXT-object it should be a shadow of! If - on the other hand - the shadow is a property to a text-object, this property is not needed.	
thickness	x,y	Thickness in each direction, range 1-2	
color	GraphicColor	Outline color	

[tsref:->GIFBUILDER.(GBObj).OUTLINE]

BOX:

Property:	Data type:	Description:	Default:
dimensions	x,y,w,h +calc	Dimensions of a filled box. x,y is the offset. w,h is the dimensions. Dimensions of 1 will result in 1-pixel wide lines!	
color	GraphicColor	fill-color	black
align	VHalign		

[tsref:->GIFBUILDER.(GBObj).BOX]

IMAGE:

Property:	Data type:	Description:	Default:
file	imgResource	The imagefile	
offset	x,y +calc	Offset	0,0
tile	x,y	tile x,y times. Maximum times is 20 each direction. If you need more, use a larger image.	

Property:	Data type:	Description:	Default:
align	VHalign		
mask	imgResource	Optional mask-image for the imagefile.	

[tsref:->GIFBUILDER.(GBObj).IMAGE]

EFFECT:
.value = [Varnavn] = [value] | [Varnavn] = [value]

Example:
20 = EFFECT
20.value = gamme=1.3 | flip | rotate=180

Property:	Data type:	Description:	Default:
gamma	0.5 - 3.0		
blur	1-99		
sharpen	1-99		
solarize	0-99		
swirl	0-100		
wave	ampli , length		
charcoal	0-100		
gray	-		
edge	0-99		
emboss	-		
flip	-	Vertical flipping	
flop	-	Horizontal flipping	
rotate	0-360	Rotation	
colors	2-255		
shear	-90 - 90	Horizontal shearing	
invert	-	invert the colors	

[tsref:->GIFBUILDER.(GBObj).EFFECT]

WORKAREA:
Sets another workarea

Property:	Data type:	Description:	Default:
set	x,y,w,h + calc		
clear	(isset)		

[tsref:->GIFBUILDER.(GBObj).WORKAREA]

CROP:
NOTE: This object resets workArea to the new dimensiosn of the image!

Property:	Data type:	Description:	Default:
backColor	GraphicColor		The original backColor
align	VHalign		
crop	x,y,v,h + calc	x,y is offset of the crop-frame, v,h is the dimensions	

[tsref:->GIFBUILDER.(GBObj).CROP]

SCALE:
NOTE: This object resets workArea to the new dimensiosn of the image!

Property:	Data type:	Description:	Default:
width	pixels + calc		
height	pixels + calc		
params	ImageMagickParams		

[tsref:->GIFBUILDER.(GBObj).SCALE]

ADJUST:

This lets you adjust the input-levels like in Photoshops "levels"-dialog. If you need to adjust gamma, look at the EFFECT-object.

Example:

```
20 = ADJUST
20.value = inputLevels = 13,230
```

Property:	Data type:	Description:	Default:
inputLevels	low,high		
outputLevels	low, high		
autoLevels	-		

[tsref:->GIFBUILDER.(GBObj).ADJUST]

NON-GifBuilderObj:

IMGMAP:

This is used by the GifBuilderObj "TEXT" to create a image-map for the gif-file. This is especially used with the IMGMENU menuobject.

Property:	Data type:	Description:	Default:
url	url	url to link	For IMGMENU menu objects provided automatically
target	target	target for link	For IMGMENU menu objects provided automatically
noBlur	Boolean	Normally graphical links are "blurred" if the browser is msie. This removes the ugly box around a link. If this property is set, the link is NOT blurred with "onFocus".	For IMGMENU menu objects provided automatically
explode	x,y	This "explodes" or "implodes" the image-map. Useful to let the hot area cover a little more than just the letters of the text.	
altText	atring	Value of the alt-attribute. (Used from TEXT Gifbuilding objects, this has stdWrap properties. Otherwise not)	
titleText	string	Value of the title attribute. (Used from TEXT Gifbuilding objects, this has stdWrap properties. Otherwise not)	

[tsref:->IMGMAP]

MENU Objects

Common properties:

These properties are in common for *all* menu objects if not otherways stated!

Property:	Data type:	Description:	Default:
sectionIndex		(see below)	
alternativeSortingField		Normally the menuitems are sorted by the fields "sorting" in the pages- and tt_content-table. Here you can enter a list of fields that is used in the SQL- "ORDER BY" statement instead. **Examples (for "pages" table):** alternativeSortingField = title desc (This will render the menu in reversed alphabetical order) **LIMITATIONS:** This property works with normal menus, sectionsIndex menus and special-menus of type "directory".	
minItems	int	The minimum items in the menu. If the number of pages does not reach this level, a dummy-page with the title "..." and uid=[currentpage_id] is inserted. Takes precedence over HMENU.minItems	
maxItems	int	The maximum items in the menu. More items will be ignored. Takes precedence over HMENU.maxItems	
begin	int +calc	The first item in the menu. **Example:** This results in a menu, where the first two items are skipped starting with item number 3: begin = 3 Takes precedence over HMENU.begin	
JSWindow	boolean	If set, the links of the menu-items will open by JavaScript in a pop-up window. **.newWindow** boolean, that lets every menuitem open in its own window opposite to opening in the same window for each click. **.params** is the list of parameters sent to the JavaScript open-window function, eg: width=200,height=300,status=0,menubar=0 **NOTE:** Does not work with JSMENU's	
imgNamePrefix	string	prefix for the imagenames. This prefix is appended with the uid of the page.	"img"
imgNameNotRandom	boolean	If set, the image names of menuitems is not randomly assigned. Usefull switch if you're manipulating these images with some external JavaScript **NOTE:** Don't set this if you're working with a menu with sectionIndex! In that case you need special unique names of items based on something else than the uid of the parent page of course!	
debugItemConf		Outputs (by the debug()-function) the configuration arrays for each menuitem. Useful to debug optionSplit things and such... Applies to GMENU, TMENU, IMGMENU	
overrideId	integer (page-id)	If set, then all links in the menu will point to this pageid. Instead the real uid of the page is sent by the parameter "&real_uid=[uid]". This feature is smart, if you have inserted a menu from somewhere else, perhaps a shared menu, but wants the menuitems to call the same page, which then generates a proper output based on the real_uid. Applies to GMENU, TMENU, IMGMENU	
addParams	string	Additional parameter for the menu-links. **Example:** "&some_var=some%20value" Must be rawurlencoded. Applies to GMENU, TMENU, IMGMENU	

Property:	Data type:	Description:	Default:
itemArrayProcFunc	function-name	The first variable passed to this function is the "menuArr" array with the menuitems as they are collected based on the type of menu. You're free to manipulate or add to this array as you like. Just remember to return the array again! **Note:** .parentObj property is <u>hardcoded</u> to be a reference to the calling tslib_menu object. Here you'll find eg. ->id to be the uid of the menu item generating a submenu and such. **Presetting element state** You can override element states like SPC, IFSUB, ACT, CUR or USR by setting the key ITEM_STATE in the page records. See cObject HMENU/special=userdefined for more information.	

[tsref:(cObject).HMENU.(mObj)]

Common item states for TMENU, GMENU and IMGMENU series:

These properties are in common for TMENU, GMENU and IMGMENU series. That means they are not used by for instance the JSMENU.

Property:	Data type:	Description:	Default:

NO	Boolean / (config)	The default "Normal" state rendering of Item. This is required for all menus.
		If you specify properties for the "NO" property you do not have to set it "1". Otherwise with no properties setting "NO=1" will render the menu anyways (for TMENU this may make sense).
		
		The simplest menu TYPO3 can generate is then:
		
		`page.20 = HMENU`
		`page.20.1 = TMENU`
		`page.20.1.NO = 1`
		
		That will be pure <a> tags wrapped around page titles.
			1

IFSUB	Boolean / (config)	Enable/Configuration for menu items which has subpages	0
ACT	Boolean / (config)	Enable/Configuration for menu items which are found in the rootLine	0
ACTIFSUB	Boolean / (config)	Enable/Configuration for menu items which are found in the rootLine and has subpages	0
CUR	Boolean / (config)	Enable/Configuration for a menu item if the item is the current page.	0
USR	Boolean / (config)	Enable/Configuration for menu items which are access restricted pages that a user has access to.	0
SPC	Boolean / (config)	Enable/Configuration for 'Spacer' pages. Spacers are pages of the doktype "Spacer". These are not viewable pages but "placeholders" which can be used to divide menuitems. **Note:** Rollovers doesn't work with spacers, if you use GMENU!	0
USERDEF1	Boolean / (config)	Userdefined, see .itemArrayProcFunc for details on how to use this. You can set the ITEM_STATE values USERDEF1 and USERDEF2 (+...RO) from a script/userfunction processing the menu item array. See HMENU/special=userdefined or the property .itemArrayProcFunc of the menu objects.	
USERDEF2	Boolean / (config)	(See above)	

[tsref:(cObject).HMENU.(mObj_itemStates)]

Order of priority: USERDEF2, USERDEF1, SPC, USR, CUR, ACTIFSUB, ACT, IFSUB

[menuObj].sectionIndex

This is a property that all menuObj's share. If it's set, then the menu will not consist of links to pages on the "next level" but rather links to the parent page to the menu, but in addition "#"-links to the cObjects rendered on the page. In other words, the menuitems will be links to the content elements (with colPos=0!) on the page. A section index.

.sectionIndex = [boolean]

If you set this, all content elements (from tt_content table) of "Column" = "Normal" *and* the "Index"-check box clicked are selected. This corresponds to the "Menu/Sitemap" content element when "Section index" is selected as type.

.sectionIndex.type = "all" / "header"

If you set this additional property to "all", then the "Index"-checkbox is not considered and all content elements with colPos=0 is selected.

If this property is "header" then only content elements with a visible header-layout (and a non-empty 'header'-field!) is selected. In other words, if the header layout of an element is set to "Hidden" then the page will not appear in the menu.

The data-record /Behind the scene:

When the menu-records are selected it works like this: The parent page record is used as the "base" for the menu-record. That means that any "no_cache" or "target"-properties of the parent page is used for the whole menu.

But of course some fields from the tt_content records are transfered: This is how it mapped:

```
$temp[$row[uid]]=$basePageRow;
$temp[$row[uid]]["title"]=$row["header"];
$temp[$row[uid]]["subtitle"]=$row["subheader"];
$temp[$row[uid]]["starttime"]=$row["starttime"];
$temp[$row[uid]]["endtime"]=$row["endtime"];
$temp[$row[uid]]["fe_group"]=$row["fe_group"];
$temp[$row[uid]]["media"]=$row["media"];
$temp[$row[uid]]["header_layout"]=$row["header_layout"];
$temp[$row[uid]]["bodytext"]=$row["bodytext"];
$temp[$row[uid]]["image"]=$row["image"];
$temp[$row[uid]]["sectionIndex_uid"]=$row["uid"];
```

Basically this shows that

- the field "header" and "subheader" from tt_content are mapped to "title" and "subtitle" in the pages-record. Thus you shouldn't need to change your standard menu-objects to fit this thing...

- the fields "starttime", "endtime", "fe_group", "media" from tt_content are mapped to the same fields in a pages-record.

- the fields "header_layout", "bodytext" and "image" are mapped to non-existing fields in the page-record

- a new field, "sectionIndex_uid" is introduced in the page-record which is detected by the function t3lib_tstemplate->linkData(). If this field is present in a pagerecord, the linkData()-function will prepend a hash-mark and the number of the field.

NOTE:

You cannot create submenus to sectionIndex-menus. That doesn't make any sense as these elements are not pages and thereby have no "childs".

GMENU:

GMENU works as a object under the cObject "HMENU" and it creates graphical navigation, where each link is a separate gif-file

Property:	Data type:	Description:	Default:
RO	Boolean	RollOver configuration enabled / disabled. If this is true, RO becomes a GIFBUILDER-object defining the layout of the menuitem when the mouse rolls over it	0
expAll	Boolean	If this is true, the menu will always show the menu on the level underneath the menuitem. This corresponds to a situation where a user has clicked a menuitem and the menu folds out the next level. This can enable that to happen on all items as default.	
collapse	Boolean	If set, "active" menuitems that has expanded the next level on the menu will now collapse that menu again.	
accessKey	Boolean	If set access-keys are set on the menu-links	
noBlur	Boolean	Normally graphical links are "blurred" if the browser is MSIE. Blurring removes the ugly box around a clicked link. If this property is set, the link is NOT blurred (browser-default) with "onFocus".	
target	target	Target of the menulinks	self
forceTypeValue	int	If set, the &type parameter of the link is forced to this value regardless of target. Overrides the global equivalent in 'config' if set.	
wrap	wrap	Wraps only if there were items in the menu!	
applyTotalH	objNumsList (offset)	This adds the total height of the previously generated menuitems to the offset of the GifBuilderObj's mentioned in this list. **Example:** This is useful it you want to create a menu with individual items but a common background image that extends to the whole area behind the menu. Then you should setup the background image in each GIFBUILDER-object and include the object-number in this list. Look at the implementation in static_template "styles.gmenu.bug"	

Property:	Data type:	Description:	Default:
applyTotalW	objNumsList (offset)	This adds the total width of the previously generated menuitems to the offset of the GifBuilderObj's mentioned in this list.	
min	x,y (calcInt)	Forces the menu as a whole to these minimum dimensions	
max	x,y (calcInt)	Forces the menu as a whole to these maximum dimensions	
useLargestItemX	boolean	If set, then the width of all menuitems will be equal to the largest of them all.	
useLargestItemY	boolean	If set, then the height of all menuitems will be equal to the largest of them all.	
distributeX	int+	If set, the total width of all the menuitems will be equal to this number of pixels by adding/subtracting an equal amount of pixels to each menu items width. Will overrule any setting for ".useLargestItemX"	
distributeY	int+	If set, the total height of all the menuitems will be equal to this number of pixels by adding/subtracting an equal amount of pixels to each menu items height. Will overrule any setting for ".useLargestItemY"	
removeObjectsOfDummy	objNumsList	If the menu is forced to a certain minimum dimension, this is a list of objects in the gifbuilder-object that is removed for this last item. This is important to do if the menuitems has elements that should only be applied if the item is actually a menuitem!!	
disableAltText	boolean	If set, the alt-parameter of the images are not set. You can do it manually by "imgParams" (see below)	
IProcFunc	function-name	The internal array "I" is passed to this function and expected returned as well. Subsequent to this function call the menu item is compiled by implode()'ing the array $I[parts] in the passed array. Thus you may modify this if you need to. See example on the testsite and in media/scripts/example_itemArrayProcFunc.php	
[Common Item States, see above] + rollover version for all, except SPC	->GIFBUILDER + Additional properties! See table below	This is the GIFBUILDER-options for each category of menuitem that can be generated. **NOTE:** For the GMENU series you can also define the RollOver configuration for the item states. This means that you define the GIFBUILDER object for the 'Active' state by ACT and the RollOver GIFBUILDER object for the 'Active' state by ACTRO. This pattern goes for ALL the states except the SPC state. **SPECIAL:** The ->OptionSplit function is run on the whole GIFBUILDER-configuration before the items are generated.	

[tsref:(cObject).HMENU.(mObj).GMENU]

Additional properties for Menu item states:

These properties are additionally available for the GMENU item states although the main object is declared to be GIFBUILDER.
It is evident that it is an unclean solution to introduce these properties on the same level as the GIFBUILDER object in a single situation like this. However this is how it irreversibly is and has been for a long time.

Property:	Data type:	Description:	Default:
noLink	boolean	If set, the item is NOT linked!	
imgParams	params	Parameters for the <img>-tag	
altTarget	string	Alternative target which overrides the target defined for the GMENU	
altImgResource	imgResouce	Defines an alternative image to use. If an image returns here, it will override any GIFBUILDER configuration.	
ATagTitle	string / stdWrap	which defines the title attribute of the a-tag. (See TMENUITEM also)	
wrap	wrap	Wrap of the menu item	
allWrap /+stdWrap	wrap	Wraps the whole item	
subst_elementUid	boolean	If set, "{elementUid}" is substituted with the item uid.	
allStdWrap	/->stdWrap	stdWrap of the whole item	

[tsref:(cObject).HMENU.(mObj).GMENU.(itemState)]

GMENU_LAYERS / TMENU_LAYERS:

GMENU_LAYERS / TMENU_LAYERS works as an extension to GMENU/TMENU, which means the these properties underneath is additional properties to the ones above.

The purpose of xMENU_LAYERS is to create 2-level (or more!) menus where the 2nd+ level is shown on a DHTML-layer. Most features works with modern browsers including Netscape, Microsoft Internet Explorer, Mozilla, Konqueror and Opera. You can cascade the menus as you like.

NOTE: You must include the library "media/scripts/gmenu_layers.php" (for GMENU_LAYERS) and/or "media/scripts/gmenu_layers.php" (for TMENU_LAYERS) and you must also expand the xMENU_LAYERS to the next for the menu to make sense (use the expAll-flag).

Compatibilty: MSIE 4+, Netscape 4+ and 6+, Opera 5+, Konqueror.

Notes:

• Netscape 4 does not support mouseover on the layers.

• Opera seems to have problems with the mouseout event if you roll from an element to a layer. Then the event may not be fired before entering the layer. It happens only if the layer is placed very close to the trigger element. Problems from this may be that the rollover state of the items are not reset.

• Possible bug; It has been seen with cascaded layers that Opera may suddently refuse any interaction on the page, even clicking normal links. It may be a JavaScript error that makes this happen, but as even normalt links are not clickable anymore, I'm not really sure. Seems to be no problem with single-level menu.

Property:	Data type:	Description:	Default:
layerStyle	<DIV>-tag params	Parameters for the <DIV>-layer-tags in the HTML-document. You might probably not need change this. **Example:** position:absolute; VISIBILITY: hidden;	position:absolute; visibility: hidden;
lockPosition	"x" / "y" / ""	If this is set to "x" or "y" the menu on the layers is locked and does not follow the mouse-cursor (which it does if this is not set). "x" or "y" defines respectively that the summed width (x) or height (y) is added to the x or y offset of the menu. That means that you should set this value to "x" if you have a horizontal GMENU_LAYERS and to "y" if you have a verical menu.	
dontFollowMouse	boolean	If set and lockPosition is blank (so that the menu layer follows the mouse) then the menu will NOT follow the mouse but still it will appear where the mouse cursor hit the trigger-element. Usefull if you don't know the exact positions of elements. **Warning:** You should not set displayActiveOnLoad for menus with this feature enabled (because the absolute position of the layer is not known).	
lockPosition_adjust	int	A number which is added to the width/height of the menuitems in order to compensate for eg. hspace or other things between the images in the GMENU_LAYERS	
lockPosition_addSelf	boolean	Normally the width and height of the items (+lockPosition_adjust) are summed up after the item has been rendered. This is good if the direction of the menulayers is right- og downwards. But if you use directionLeft/directionUp, you might want to add the width of the items before. If so, set this flag.	
xPosOffset	int	The offset of the menu from the point where it's "activated" (if lockPosition is false) / from topleft page corner (if lockPosition is set)	
yPosOffset	int	As above, but for the y-dimension.	
topOffset	int	The offset of menuitems from top of browser. Should be set rather than defining it in the .layerStyle property. Must be set in order to use directionUp. Used with either lockPosition=x or xPosOffset defined.	
leftOffset	int	The offset of menuitems from left border of browser. Should be set rather than defining it in the .layerStyle property. Must be set in order to use directionLeft. Used with either lockPosition=y or yPosOffset defined.	
blankStrEqFalse	boolean	If set, then the properties topOffset,leftOffset, xPosOffset, yPosOffset are considered "blank" if they are really blank strings - not just "zero". You should enable this if you wish to be able to work with zero offsets. This is typically the case if you use relative positioning.	
directionLeft	boolean	Set this, if you want the items to be right-aligned (pop's out towards the left). Does not work with Opera at this time because I don't know how to make Opera read the width of each layer. If you set the width of the menu-layers in .layerStyles this might work no matter what.	

Property:	Data type:	Description:	Default:
directionUp	boolean	Set this, if you want the items to be bottom-aligned (pop's out upwards instead of downwards).	
setFixedWidth	int	For GMENU_LAYERS the width and heights of the element is normally known from the graphical item. For TMENU_LAYERS this cannot be known in the same way. Therefore you can use .setFixedWidth and .setFixedHeight to set these values to a number you find reasonable. Of course this may be blasted by the browsers rendering if the font gets out of proportions etc. Alternatively you may want to use the property "relativeToTriggerItem" which will position your menu layers relative to the item you roll over. This has some drawbacks though. A middle solution is to use a menu with lockPosition set to blank and dontFollowMouse set to true. Then you need only specify either an x or y coordinate to follow and the item will appear where the mouse hits the element. **Notice:** Active if value is NOT a blank str. Setting this value to zero means that no width is calculated for the items in GMENU_LAYERS.	
setFixedHeight	int	See "setFixedWidth". Same, but for height.	
bordersWithin	l,t,r,b,l,t	Keep borders of the layer within these limits in pixels. Zero is 'not set' (Syntax: List of integers, evaluated clockwise: Left, Top, Right, Bottom, Left, Top)	
displayActiveOnLoad	boolean	If set, the submenu-layer of the active menuitem is opened at page-load. If .freezeMouseover is also set and there is RO defined for the main menu items, the menuitem belonging to the displayed submenu is also shown. **Properties:** .onlyOnLoad (boolean) If set, then the display of the active item will happen only when the page is loaded. The display will not be restored on mouseout of other items. **Warning:** If you are cascading GMENU_LAYER objects make sure that all elements before this element (for which you set this attribute) also has this attribute set!	
freezeMouseover	boolean	If set, any mouseout effect of main menuitems is removed not on roll-out but when another element is rolled over (or the layer is hidden/default layer restored) **Properties:** .alwaysKeep (boolean) If set, the freezed element will always stay, even if the submenu is hidden.	
hideMenuWhenNotOver	int+	If set (> 1) then the menu will hide it self whenever a user moves the cursor away from the menu. The value of this parameter determines the width (pixels) of the zone around the element until the mousepointer is considered to be far enough away to hide the layer.	
hideMenuTimer	int+	This is the number of milliseconds to wait before the submenu will disappear if hideMenuWhenNotOver is set	
dontHideOnMouseUp	boolean	If set, the menu will not hide it's layers when the mouse botton is clicked. Usefull if your menuitems loads the pages in another frame	
layer_menu_id	string	If you want to specifically name a menu on a page. Probably you don't need that! **Warning:** Don't use underscore and special characters in this string. Stick to alpha-numeric.	[random 6 char hashstring]

Property:	Data type:	Description:	Default:
relativeToTriggerItem	boolean	This allows you to position the menu layers relative to the item that triggers it. However you should be aware of the following facts: • This does not work with Netscape 4 - the position of the trigger layer will be calculated to zero and thus the offset for all menu layers will be 0,0 + your values. • This feature will wrap the menu item in some <div>-tags right before the whole item is wrapped by the .wrap code (for GMENU_LAYERS) or .allWrap (for TMENU_LAYERS). The bottom line of this is: 1) If your menu is horizontal, always wrap your menu items in a table so linebreaks does not appear because of the <div>-tags and 2) make sure the wrapping of the table cell is done with the .wrap/.allWrap properties respectively. • Works only effectively on the first xMENU_LAYER in a cascade. For succeeding xMENU_LAYERS items please use "relativeToParentLayer". *If set, properties xPosOffset, yPosOffset and lockPosition* are not functional (properties directionLeft, directionUp, topOffset and leftOffset are still active)* **Additional Properties:** **.addWidth** = Adds the width of the trigger element **.addHeight** = Adds the height of the trigger element	
relativeToParentLayer	boolean	If set, then the layer will be positioned relative to the previous layer (parent) in a cascaded series of xMENU_LAYERS. Basically the relative position of the parent layer is just added to the offset of the current menu. **Warning:** This property makes sense only if there really is a previous GMENU_LAYER to get position from! So you must have a cascaded menu! **Additional Properties:** **.addWidth** = Adds the width of the parent layer **.addHeight** = Adds the height of the parent layer	

[tsref:(cObject).HMENU.(mObj).GMENU_LAYERS, (cObject).HMENU.(mObj).TMENU_LAYERS]

Example:

```
page.includeLibs.gmenu_layers = media/scripts/gmenu_layers.php
page.10 = HMENU
page.10.1 = GMENU_LAYERS
page.10.1 {
  layerStyle = position:absolute;VISIBILITY:hidden;
  xPosOffset = -30
  lockPosition = x
  expAll=1
  leftOffset = 15
  topOffset = 30
}
page.10.1.NO {
  backColor = #cccccc
  XY = [10.w]+10, 14
  10 = TEXT
  10.text.field = title
  10.offset = 5,10
}
page.10.2 = GMENU
page.10.2.wrap = <nobr>|</nobr>
page.10.2.NO {
  backColor = #99cccc
  XY = [10.w]+10, 14
  10 = TEXT
  10.text.field = title
  10.offset = 5,10
}
```

(Please refer to the "testsite" application which has a large section with test-examples for a LOT of applications/configurations of the xMENU_LAYERS!)

GMENU_FOLDOUT:

GMENU_FOLDOUT works as an extension to GMENU, which means the these properties underneath is additional properties to the ones above.

The purpose of GMENU_FOLDOUT is to create 2-level menus which are folded out dynamically.

It works with both Netscape, Mozilla, Microsoft internet Explorer and Opera. The menu on the first level is a GMENU because GMENU_FOLDOUT is responsible for this, but the submenu on the next level (refered to as 2nd level) can be both TMENU and another GMENU.

NOTE: You must include the library "media/scripts/gmenu_foldout.php".

The script implemented is taken from http://www9.ewebcity.com/skripts/foldoutmenu_move.htm

Compatibilty: MSIE 4+, Netscape 4+ and 6+, Opera 5+

Property:	Data type:	Description:	Default:
dontLinkIfSubmenu	boolean	If set, items that has a submenu is not linked. Items without a submenu are always linked in the regular ways.	
foldTimer	int	The timeout in the animation, these are milliseconds.	40
foldSpeed	int, range 1-100	How many steps in an animation? Choose 1 for no animation.	1
stayFolded	boolean	Stay open when you click a new toplink? (Level 1)	
bottomHeight	int, pixels	Sets the height of the bottom layer. Is important if the bottomlayer contains either content or a background color: Else the layer will be clipped.	100
menuWidth	int, pixels	Width of the whole menu main layer. Important to set, especially for the bottomlayer as it is clipped by this value. Always try to set this to the width in pixels of the menu	170
menuHeight	int	Height of the whole menulayer. Seems to be not so important.	400
subMenuOffset	x,y	Offset of the submenu for each menuitem. This is important because if you don't set this value the items will appear ontop of their "parent"	
menuOffset	x,y	Offset of the menu main layer on the page. From upperleft corner	
menuBackColor	HTML-color	Background color behind menu. If not set, transparent (which will not work very well in case .foldSpeed is set to something else than 1. But see for yourself)	
dontWrapInTable	boolean	By default every menuitem on the first level is wrapped in a table: <TABLE cellSpacing=0 cellPadding=0 width="100%" border=0><TR><TD> [menu item HTML here..] </TD></TR></TABLE> Doing this ensures that the layers renders equally in the supported browsers. However you might need to disable that which is what you can do by setting this flag. **Note:** Using <TBODY> in this tables seems to break Netscape 4+	0
bottomContent	cObject	Content for the bottom layer that covers the end of the menu.	
adjustItemsH	int	Adjusts the height calculation of the menulayers of the first level (called Top) **Example:** -10 This value will substract 10 pixels from the height of the layer in calculations.	
adjustSubItemsH	int	Adjusts the height calculation of the menulayers of the second level (subitems, called Sub) See above	
arrowNO arrowACT	imgResource	If both arrowNO and arrowACT is defined and valid imgResources then these images are use as "traditional arrows" that indicates whether an item is expanded (active) or not. NO is normal, ACT is expanded The image is inserted just before the menuitem. If you want to change the position, put the marker ###ARROW_IMAGE### into the wrap of the item and the image will be put there instead.	
arrowImgParams	<img> params	Parameters to the arrow-image. **Example:** hspace=5 vspace=7	
displayActiveOnLoad	boolean	If set, then the active menu items will fold out "onLoad"	

[tsref:(cObject).HMENU.(mObj).GMENU_FOLDOUT]

Example:

```
## GMENU_FOLDOUT
```

```
includeLibs.gmenu_foldout = media/scripts/gmenu_foldout.php

temp.foldoutMenu = HMENU
temp.foldoutMenu.1 = GMENU_FOLDOUT
temp.foldoutMenu.1.expAll=1
temp.foldoutMenu.1.NO {
  wrap = | <BR>
  XY = 150,20
  backColor = silver

  10 = TEXT
  10.text.field = title
  10.fontSize = 12
  10.fontColor = Blue
  10.offset = 2,10
}
temp.foldoutMenu.1.RO  < temp.foldoutMenu.1.NO
temp.foldoutMenu.1.RO = 1
temp.foldoutMenu.1.RO {
  10.fontColor = red
}
temp.foldoutMenu.2 = TMENU
temp.foldoutMenu.2.NO {
  linkWrap = <nobr><font face=verdana size=1 color=black><B>|</B></font></nobr><BR>
  stdWrap.case = upper
}
temp.foldoutMenu.1 {
  dontLinkIfSubmenu = 1
  stayFolded=1
  foldSpeed = 6
  subMenuOffset = 10,18
  menuOffset = 100,20
  menuBackColor = silver
  bottomBackColor = silver
  menuWidth = 170

  arrowNO = media/bullets/arrow_no.gif
  arrowACT = media/bullets/arrow_act.gif
  arrowImgParams = hspace=4 align=top

  bottomContent = TEXT
  bottomContent.value = Hello World! Here is some content!
}
```


This creates a menu like this (above). One important point is the line

```
temp.foldoutMenu.1.expAll=1
```

If you don't set this (just like the GMENU_LAYERS) then the second level is not generated!

TMENU:

Property:	Data type:	Description:	Default:
expAll	Boolean	If this is true, the menu will always show the menu on the level underneath the menuitem. This corresponds to a situation where a user has clicked a menuitem and the menu folds out the next level. This can enable that to happen on all items as default.	
collapse	Boolean	If set, "active" menuitems that has expanded the next level on the menu will now collapse that menu again.	
noBlur	Boolean	Normally links are "blurred" if the browser is MSIE. Blurring removes the ugly box around a clicked link. If this property is set, the link is NOT blurred (browser-default) with "onFocus".	
target	target	Target of the menulinks	self
forceTypeValue	int	If set, the &type parameter of the link is forced to this value regardless of target.	
wrap	wrap	Wraps only if there were items in the menu!	
IProcFunc	function-name	The internal array "I" is passed to this function and expected returned as well. Subsequent to this function call the menu item is compiled by implode(')'ing the array $I[parts] in the passed array. Thus you may modify this if you need to. See example on the testsite and in media/scripts/example_itemArrayProcFunc.php	
[Common Item States, see above]	->TMENUITEM	This is the TMENUITEM-options for each category of menuitem that can be generated. **SPECIAL:** The ->OptionSplit function is run on the whole GIFBUILDER-configuration before the items are generated.	

[tsref:(cObject).HMENU.(mObj).TMENU]

TMENUITEM:

The current record is the page-record of the menu item - just like you have it with GMENU/gifbuilder. Now, if you would like to get data from the current page record, use stdWrap.data = page : [fieldname]

Property:	Data type:	Description:	Default:
allWrap /+stdWrap	wrap	Wraps the whole item	
wrapItemAndSub	wrap	Wraps the whole item and any submenu concatenated to it.	
subst_elementUid	boolean	If set, all appearances of the string '{elementUid}' in the total element html-code (after wrapped in .allWrap} is substituted with the uid number of the menu item. This is useful if you want to insert an identification code in the HTML in order to manipulate properties with JavaScript.	
RO_chBgColor	string	If property RO is set (see below) then you can set this property to a certain set of parameters which will allow you to change the background color of eg. the tablecell when the mouse rolls over you text-link. **Syntax:** `[over-color]	[out-color]	[id-prefix]`  **Example:** `page = PAGE` `page.typeNum = 0` `page.10 = HMENU` `page.10.wrap = <table border=1>|</table>` `page.10.1 = TMENU` `page.10.1.NO {` ` allWrap = <tr><td valign=top id="1tmenu{elementUid}" style="background:#eeeeee;">|</td></tr>` ` subst_elementUid = 1` ` RO_chBgColor = #cccccc	#eeeeee	1tmenu` ` RO = 1` `}` This example will start out with the table cells in #eeeeee and change them to #cccccc (and back) when rolled over. The "1tmenu" string is a unique id for the menu items. You may not need it (unless the same menu items are more than once on a page), but the important thing is that the id of the table cell has the exact same label before the {elementUid} (red marks). The other important thing is that you DO set a default background color for the cell with the style-attribute (blue marking). If you do not, Mozilla browsers will behave a little strange by not capturing the mouseout event the first time it's triggered.	

Property:	Data type:	Description:	Default:
before	HTML +stdWrap		
beforeImg	imgResource		
beforeImgTagParams	<img>-params		
beforeImgLink	boolean	If set, this image is linked with the same <A> tag as the text	
beforeROImg	imgResource	If set, ".beforeImg" and ".beforeROImg" is expected to create a rollOver-pair.	
beforeWrap	wrap	wrap around the ".before"-code	
linkWrap	wrap		
stdWrap	stdWrap-options	stdWrap to the link-text!	
ATagBeforeWrap	boolean		
ATagParams	<A>-data		
ATagTitle	string / stdWrap	Allows you to specify the "title" attribute of the <a> tag around the menu item. **Example:** ATagTitle.field = abstract // description This would use the abstract or description field for the <a title=""> attribute.	
doNotLinkIt	boolean	if set, the linktext are not linked at all!	
doNotShowLink	boolean	if set, the text will not be shown at all (smart with spacers)	
stdWrap2	wrap /+stdWrap	stdWrap to the total link-text and ATag. (Notice that the plain default value passed to stdWrap function is "|".)	|
RO	boolean	if set, rollOver is enabled for this link	
after...	[mixed]	The series of "before..." properties are duplicated to "after..." properties as well. The only difference is that the output generated by the .after.... properties are placed after the link and not before.	
altTarget	target	Alternative target overriding the target property of the TMENU if set.	
allStdWrap	/->stdWrap	stdWrap of the whole item	

[tsref:(cObject).HMENU.(mObj).TMENUITEM]

IMGMENU:
Background:

Imagemaps are made by creating one large GIFBUILDER-object based on the GIFBUILDER-object ".main" and adding the properties of the GIFBUILDER-objects for each item (NO, ACT, SPC... and so on).

Property:	Data type:	Description:	Default:
target	target	Target of the menulinks	self
forceTypeValue	int	If set, the &type parameter of the link is forced to this value regardless of target.	
noBlur	Boolean	Normally graphical links are "blurred" if the browser is MSIE. Blurring removes the ugly box around a clicked link. If this property is set, the link is NOT blurred (browser-default) with "onFocus".	
wrap	wrap		
params	<img>-params		
main	->GIFBUILDER	Main configuration of the image-map! This defines the "underlay"!	
dWorkArea	offset + calc	Main offset of the GIFBUILDER-items (also called the "distribution")	

Property:	Data type:	Description:	Default:
[Common Item States, see above]	->IMGMENUITEM + .distrib	This is the TMENUITEM-options for each category of menuitem that can be generated. **SPECIAL:** The ->OptionSplit function is run on the whole GIFBUILDER-configuration before the items are generated. **.distrib** is (x,y,v,h +calc) of the distribution of the menuitems. This provides a way to space each item from the other. The codes "textX" and "textY" can be used for the width (X) and height (Y) dimension of each link. This works by adding a WORKAREA-GifBuilderObj between each of the IMGMENUITEM ("subset" of a GIFBUILDER-object) and this workarea defines where the text should be printed. As such the "x,y" defines the offset *the next item will have* (this should be the width of the previous in many cases!) and "v,h" defines the *dimensions of the current item*. Consider this example taken from the static_template "template: MM": NO.distrib = textX+10, 0, textX+10, textY+5 In the future TypoScript may provide better ways to position GIFBUILDER-objects on the image-maps! **ImgMap** is automatically used on the links! (that is the ".imgMap" property of the text-objects in the GIFBUILDER-objects is set automatically, unless is allready set.)	
imgMapExtras	<area...>-tags	Extra <area...>tags for the image-map	
debugRenumberedObject	boolean	if set, the final GIFBUILDER object configuration is output in order for you to debug your configuration	

[tsref:(cObject).HMENU.(mObj).IMGMENU]

IMGMENUITEM:

Property:	Data type:	Description:	Default:

1,2,3,4...	->GifBuilderObj	**NOTE:**
		The way a imagemap is made is this; All IMGMENUITEMS are included in one big Gifbuilderobj (and renumbered!!). Because of this, Gifbuilderobjects on the next level will not be able to access the data of each menuitem.
		Also the feature of using [##.w] and [##.h] with +calc is currently not supported by IMGMENUITEMs.
		Therefore all IMAGE-objects on the first level is checked; if "file" or "mask" for any IMAGE-objects are set to "GIFBUILDER", the Gifbuilder-object is parsed to see if any TEXT-objects are present and if so, the TEXT-object is "checked" - which means, that the stdWrap-function is called at a time where the $cObj->data-array is set to the actual menuitem.
		In the example below, the text of each menuitem is rendered by letting the title be rendered on a mask instead of directly on the image. Please observe that the "NO.10"-object is present in order for the image-map coordinates to be generated!!
		
		`NO.6 = IMAGE`
		`NO.6.file = masked_pencolor*.gif`
		`NO.6.mask = GIFBUILDER`
		`NO.6.mask {`
		`  XY = 500, 200`
		`  backColor = black`
		`  10 = TEXT`
		`  10 {`
		`    text.field = title`
		`    fontFile = fileadmin/fonts/caflisch.ttf`
		`    fontSize = 34`
		`    fontColor = white`
		`    angle = 15`
		`    offset = 48,110`
		`  }`
		`  20 = EFFECT`
		`  20.value = blur=80`
		`}`
		`NO.10 = TEXT`
		`NO.10 {`
		`  text.field = title`
		`  fontFile = fileadmin/fonts/caflisch.ttf`
		`  fontSize = 34`
		`  angle = 15`
		`  offset = 48,110`
		`  hideButCreateMap = 1`
		`}`	

[tsref:(cObject).HMENU.(mObj).IMGMENUITEM]

JSMENU:

Property:	Data type:	Description:	Default:
levels	int, 1-5	How many levels there are	1
menuName	string	JavaScript menu name. If you have more than one JSMENU on the page, you should set this value for each one.	
target	target	Decides target of the menu-links	
forceTypeValue	int	If set, the &type parameter of the link is forced to this value regardless of target.	
1,2,3,4...	JSMENUITEM	levels-config	
wrap	wrap	wrap around the selector-boxes	
wrapAfterTags	wrap	wrap around the selector-boxes with wrap and form-tags og JS-code.	
firstLabelGeneral	string	General firstlabel. May be overridden by the one set in each JSMENUITEM	
SPC	boolean	If set, spacer can go into the menu, else not.	

[tsref:(cObject).HMENU.(mObj).JSMENU]

JSMENUITEM:

Property:	Data type:	Description:	Default:
noLink	boolean	Normally the selection of a menu item in the selector box will update the selector on the next level (if there is a next level) and if there are no items for that selector (because there were no subpages), then the link jumps to the page of itself. If this flag is set, however, no menuitems in the selector box will ever link to anything. Only update the content of the next selector box on next level.	
alwaysLink	boolean	If set an item in the menu selector will always link. This takes precedence over "noLink".	
showFi rst	boolean	if set, the first link will be shown when the menu is updated.	
showActive	boolean	if set, the active level will be selected, if present	
wrap	wrap	wraps the selectorbox	
width	int+	Initial width of the boxes set by a number of _ (underscores)	14
elements	int+	Initial number of elements in the menu. This is of course overruled by the actual menu item texts.	5
additionalParams	string	Additional parameters to the <select> box. Eg, you could set the width with a style-parameter like this: style="width: 200px;"	
firstLabel	string	Firt label in top of the menu (default is blank)	

[tsref:(cObject).HMENU.(mObj).JSMENUITEM]

Example:

```
# The menu:
temp.jsmenu = HMENU
temp.jsmenu.1 = JSMENU
temp.jsmenu.1 {
  levels = 2
  1.wrap = |<BR>
  2.wrap = |<HR>
}

# Insert on page.
page = PAGE
page.typeNum =0
page.5 = TEXT
page.5.field = title
page.10 < temp.jsmenu
```

This draws a menu with two selector boxes

media/scripts/ Plugins

media/scripts/ in general

This directory primarily contains php-scripts which are meant as 'external modules' as opposed to features included in the tslib/ libraries. Although they are distributed with Typo3 just like tslib/ they form a basis for externally developed front-end functionality. So for most of these scripts, be inspired by them to write your own code. Notice the word 'most'; because some are written long time ago and does not represent the state-of-the-day to do it.

About 'example templates'

For each plugin script there is one or more example templates. These templates are a part of the documentation of the features in the plugin because they describe the features of the markers and subparts and present an example to learn from. Therefore the example templates may be change when new features come along or of other reasons.

You should therfore *not* rely on using the default templates unless you'll except the fact that they may change in the future! So make a copy, modify it for your own purpose if need and set up the TypoScript of the plugin to use your own template file!

fe_adminLib.inc

Files:

File:	Description:
fe_adminLib.inc	Main class used to display the fe administration forms Call it from a USER_INT cObject with 'userFunc = user_feAdmin->init'. See the static_templates for examples. **Note:** Using the USER_INT cObject allows the script to work regardless of the page-cache which is necessary!!
fe_admin_dmailsubscrip.tmpl	Example template file for subscription to newsletters of users to the tt_address table. This template is used by the static_template 'plugin.feadmin.dmailsubscription'
fe_admin_fe_users.tmpl	Example template file for creating new front end users (fe_users). This template is used by the static_template 'plugin.feadmin.fe_users'

Description

This class is used to create forms for database-administration in the front-end *independently of the backend (TBE).* Thus you may want to use this, if you would like front-end users to edit database content.

Authentication goes through either fe_user login in which case you can stamp the records with the fe_user_uid so a record belongs to a certain fe_user. The other authentication option is email authentication. In this case you have access to the record if your email is found in a certain field. By fe_user authentication you can get a menu of items to edit when you're logged in. With email-authentication, you can request an email to be sent to your email address. This email contains a list of the available records.

It's all based on HTML-template files which you have to design by your self, so there's some design work to do. On the other hand you get total freedom to design your forms.

Example:
See static_templates 'plugin.feadmin.*' for various examples. Test them configured on the Typo3 testsite.

Static template
plugin.feadmin.*

Incoming GET or POST vars:

Name:	Description:
cmd	Command;
preview	Preview flag.
backURL	Back URL.
rU	Record UID.
aC	Authentication Code.
fD	fixed Data (array of fields)
FE	Frontend Edit data array, syntax, FE[*tablename*][*fieldname*] = value

fe_adminLib.inc properties

Property:	Data type:	Description:	Default:
templateFile	resource	The template file, see examples in media/scripts/fe_user_admin.tmpl	
templateContent	string	Alternatively you can set this property directly to the value of the template.	
table	tablename	The table to edit. Notice: The ultimate lsit of fields allowed to be edited for the table is defined in TCA with the key ["feInterface"]["fe_admin_fieldList"] for each table in question. For an example, see the table definition for fe_users which is a good example.	
defaultCmd	string	Defines which action should be default (if &cmd= is not set when calling the page)	
clearCacheOfPages	*[list of integers]*	This is a list of page-ids for which to clear the cache on any successfull operation be it EDIT, CREATE og DELETE.	
debug	boolean	If set, debug information will be output from fe_adminLib which helps to track errors.	
Actions:			

Property:	Data type:	Description:	Default:
edit	boolean / actionObject	If set, editing is basically allowed. But you need to specify: **.fields** (list of fieldnames) which determines the fields allowed for editing. Every field in this list must be found as well in the ["feInterface"]["fe_admin_fieldList"] found in the TCA array which ultimately determines which fields can be edited by the fe_adminLib. **.overrideValues.[fieldname]** (value string) defines values for specific fields which will override ANY input from the form. Overriding values happens after the outside values has been parsed by the .parseValues-property of fe_adminLib but before the evaluation by .required and .evalValues below. For example this may be useful if you wish to hide a record which is being edited, because you want to preview it first. **.required** (list of fieldnames, subset of .fields) which determines which fields are required to return a true value. The valid fields entered here will have the subpart ###SUB_REQUIRED_FIELD_*[fieldname]###* removed from the templates if they evaluates to being true and thereby OK. See below for information about this subpart. **.evalValues.[fieldname]** (list of eval-codes) defines specific evaluation forms for the individual fiels of the form. See below. **.preview** (boolean) will enable the form submitted to be previewed first. This requires a template for preview to be found in the template file. See below for subpart marker names. **.menuLockPid** (boolean will force the menu of editable items to be locked to the .pid (edit only) **.userFunc_afterSave** (function-name) is called after the record is saved. The content passed is an array with the current (and previous) record in.	
create	boolean / actionObject	The same as .edit above except where otherwise stated. Plus there is these additional properties: **.noSpecialLoginForm** (boolean) - if set, fe_adminLib does NOT look for the subpart marker TEMPLATE_CREATE_LOGIN but always for TEMPLATE_CREATE **.defaultValues.[fieldname]** (value string); Like .overrideValues but this sets the default values the first time the form is displayed.	
delete	boolean	Whether or not records may be deleted. Still regular authentication (ownership or email authCode) is required. Setting the var "preview" lets you make a delete-preview before actually deleting the record.	
infomail	boolean	Infomails are plaintext mails based on templates found in the template file. They may be used for such as sending a forgotten password to a user, but what goes into the infomail is totally up to your design of the template. Normally you may have only a default infomail (infomail.default) for instance for sending the password. But you can use other keys also. See below.	
infomail.[key]	(configuration of infomail properties)	In order to make fe_adminLib send an infomail, you must specify these vars in your GET vars or HTML-form. **fetch** - if integer, it searches for the uid being the value of 'fetch'. If not, it searches for the email-field (defined by a property of fe_adminLib, see below). **key** - points to the infomail.[key] configuration to use **Properties:** **.dontLockPid** (boolean) - selects only records from the .pid of fe_adminLib. **.label** (string) - The suffix for the markers, see 'Email Markers' beneath.	

Property:	Data type:	Description:	Default:

setfixed	boolean / properties	Allows set-fixed input, probably coming from a link in an infomail or notification mail.
		Syntax:
		
		.[fix*key*].[*fieldname*] = *fieldvalue* - is used to setup a setfixed-link insertable in the infomail by the SYS_SETFIXED_*-markers. See above (setfixed-property of fe_adminLib).
		Special fixkey 'DELETE' is just a boolean.
		
		Concept:
		The 'setfixed' concept is best explained by describing a typical scenario - in fact the most common situation of its use:
		Imagine you have some users submitting information on your website. But before that information enters the database, you would like to moderate it - simply preview it and then either delete it or approve it. In the 'create' configuration of fe_adminLib, you set up the hidden field of the record to be overridden to 1. Thus the record is hidden by default. Then you configure a setfixed-fixkey to set the hidden field to 0. This set up generates a list of parameters for use in an URL and those parameters are finally inserted by a corresponding marker in the email template. The link includes all necessary authentication to perform the change of values and thus a single click on that link is enough to change the field values. So this will - by a single click of a link in a notification mail sent to an admin - enable the record! Or of course a similar link with a cmd=delete link will delete it...
		There is a special "fieldname" you can use, which is '_FIELDLIST" and that lets you specify a list of fields in the record to base the auth-code on. If nothing is specifyed the md5-hash is based on the whole record which means that any changes will disable the setfixed link. If on the other hand, you set _FIELDLIST = uid,pid then that record will be editable as long as the uid and pid values are intact.
		Example:
		This is a common configuration of the email-properties with a simple setfixed setting:
		<pre>email.from = kasper@typo3.com
		email.fromName = Kasper Skårhøj
		email.admin = kasper@typo3.com
		setfixed.approve {
		  hidden = 0
		  _FIELDLIST = uid,pid
		}
		setfixed.DELETE = 1
		setfixed.DELETE._FIELDLIST = uid</pre>
		Now, if you insert this marker in your email template
		
		`###SYS_SETFIXED_approve###`
		
		if will get substituted with something like these parameters:
		
		`&cmd=setfixed&rU=9&fD[hidden]=0&aC=5c403d90`
		
		Now, all you need is to point that to the correct url (where fe_adminLib is invoked!), eg:
		
		`###THIS_URL######FORM_URL######SYS_SETFIXED_approve###`
		
		and for deletion:
		
		`...###SYS_SETFIXED_DELETE###`	

Others			

TYPO3
get.content.right

Property:	Data type:	Description:	Default:
authcodeFields	*[list of fields]*	Comma separated list of fields to base the authCode generation on. Basically this list would include "uid" only in most cases. If the list includes more fields, you should be aware that the authCode will change when the value of that field changes. And then the user will have to re-send an email to himself with a new code. **.addKey** (string) adds the string to the md5-hash of the authCode. Just enter any random string here. Point is that people from outside doesn't know this code and therefore are not able to reconstruct the md5-hash solely based on the uid **.addDate** (date-config) You can use this to make the code time-disabled. Say if you enter "d-m-Y" here as value, the code will work until midnight and then a new code will be valid. **.codeLength** (int) Defines how long the authentication code should be. Default is 8 characters. In any case TYPO3_CONF_VARS[SYS][encryptionKey] is prepended. **Advice:** If you want to generate authCodes compatible with the standard authCodes (used by the direct mailer by t3lib_div::stdAuthCode()), please set TYPO3_CONF_VARS[SYS][encryptionKey] to a unique and secret key (like you should in any case) and add "uid" as authcodeField ONLY. This is secure enough.	
email		**.from** (string, email) Defines the sender email address of mails sent out **.fromName** (string) Defines the name of the sender. If set, this will be used on the form NAME <EMAIL> **.admin** Email address of the administrator which is notified of changes. **.field** (string/integer) Defines the fieldname of the record where the email address to send to is found. If the field content happens to be an integer, this is assumed to be the uid of the fe_user owning the record and the email address of that user is fetched for the purpose instead.	
pid	int+	The pid in which to store/get the records.	Current page
fe_userOwnSelf	boolean	If set, fe_users created by this module has their fe_cruser_id-field set to their own uid which means they 'own' their own record and can thus edit their own data. All other tables which has a fe_cruser_id field configured in the 'ctrl' section of their $TCA-configuration will automatically get this field set to the current fe_user id.	
fe_userEditSelf	boolean	If set, fe_users - regardless of whether they own themselves or not - will be allowed to edit himself.	
allowedGroups	[list of integers]	List of fe_groups uid numbers which are allowed to edit the records through this form. Normally only the owner fe_user is allowed to do that.	
evalFunc	function-name	Function by which you can manipulate the dataArray before it's saved. The dataArray is passed to the function as $content and MUST be returned again from the function. The property "parentObj" is a hardcoded reference to the fe_adminLib object.	
no_cache	boolean	If set, &no_cache=1 is sent in the links of the fe_adminLib	

Property:	Data type:	Description:	Default:
parseValues.[field]	[list of parseCodes]	**ParseCodes:** **int** - returns the integer value of the input **lower** - returns lowercase version of the input **upper** - returns uppercase version of the input **nospace** - strips all space **alpha, num, alphanum, alphanum_x** - only alphabetic (a-z) and/or numeric chars. alphanum_x also allows _ and - **trim** - trims whitespace in the ends of the string **setEmptyIfAbsent** - will make sure the field is set to empty if the value is not submitted. This ensures a field to be updated an is handy with checkboxes **random[x]** - Returns a random number between 0 and x **files[semicolon-list(!) of extensions, none=all][maxsize in kb, none=no limit]** - Defining the field to hold files. See below for details! **multiple** - Set this, if the input comes from a multiple-selector box (remember to add ...[] to the fieldname so the values come in an array!) **checkArray** - Set this, if you want several checkboxes to set bits in a single field. In that case you must prepend every checkbox with [x] where x is the bitnumber to set starting with zero. The default values of the checkbox form elements must be false. **uniqueHashInt[semicolon-list(!) of other fields]** - This makes a unique hash (32 bit integer) of the content in the specified fields. The values of those fields are first converted to lowercase and only alphanum chars are preserved.	
userFunc_updateArray	function-name	Points to a user function which will have the value-array passed to it before the value array is used to construct the update-JavaScript statements.	
evalErrors.[field].[evalCode]		This lets you specify the error messages inserted in the ###EVAL_ERROR_FIELD_[fieldname]### markers upon an evaluation error. See description of evaluation below.	

cObjects.[marker_name]	cObject	This is cObjects you can insert by markers in the template.
		
		Example:
		Say, you set up a cObject like this:
		
		`cObject.myHeader = TEXT`
		`cObject.myHeader.value = This is my header`
		
		then you can include this cObject in most of the templates through a marker named ###CE_myHeader### or ###PCE_myHeader### (see below for details on the difference).	

wrap1	-> stdWrap	Global Wrap 1. This will be splitted into the markers ###GW1B### and ###GW1E###. Don't change the input value by the settings, only wrap it in something. **Example:** `wrap1.wrap = <b> |</B>`	
wrap2	-> stdWrap	Global Wrap 2 (see above)	
color1	string /stdWrap	Value for ###GC1### marker (Global color 1)	
color2	string /stdWrap	Value for ###GC2### marker (Global color 2)	
color3	string /stdWrap	Value for ###GC3### marker (Global color 3)	

[tsref:(script).fe_adminLib]

Main subparts

There are a certain system in the naming of the main subparts of the template file. These markers below is used when an action results in "saving". The *[action]* code may be DELETE, EDIT or CREATE depending on the cmd value.

Subpart marker:	Description:
###TEMPLATE_*[action]*_SAVED###	Used for HTML output
###TEMPLATE_SETFIXED_OK### (general) ###TEMPLATE_SETFIXED_OK_[*fixkey*]###	Used for a successfull setfixed-link.
###TEMPLATE_SETFIXED_FAILED###	Used for an unsuccessfull setfixed-link. Notice that if you click a setfixed link twice, the second time it will fail. This is because the setfixed link is bound to the original record and if that changes in any way the authentication code will be invalid!
###EMAIL_TEMPLATE_*[action]*_SAVED###	Used for an email message sent to the website user
###EMAIL_TEMPLATE_*[action]*_SAVED-ADMIN###	Used for an email message sent to the admin

Subpart marker:	Description:
###EMAIL_TEMPLATE_SETFIXED_*[fixkey]*###	Used for notification&messages in the event of successfull setfixed operations.
###EMAIL_TEMPLATE_SETFIXED_*[fixkey]*-ADMIN###	Ditto, for admin email

Likewise there are a system in the subpart markers used for the EDIT and CREATE actions to display the initial forms:

###TEMPLATE_*[action]*###   or if a fe_user is logged in (only CREATE): ###TEMPLATE_*[action]*_LOGIN###

... and if the &preview-flag is sent as well (including DELETE)

###TEMPLATE_*[action]*_PREVIEW###

Must-have subparts:

These are subparts that should exist in any template.

Subpart marker:	Description:
###TEMPLATE_AUTH###	Displayed if the authentication - either of fe_user or email authentication code - failed. You must design the error display to correctly reflect the problem!
###TEMPLATE_NO_PERMISSIONS###	This error message is displayed if you were authenticated but of other reasons (like wrong fe_user/group ownership) did not posses the right to edit or delete a record.

'infomail' Email subparts
All email subparts can be sent as HTML. This is done if the first and last word of the templates is <html> and </html> respectively. In addition the t3lib_htmlmail class must be loaded.

Subpart:	Description:
###EMAIL_TEMPLATE_NORECORD###	
###EMAIL_TEMPLATE_[*infomail_key*]###	
###SUB_RECORD###	

'infomail' Email markers

Marker:	Description:
###SYS_AUTHCODE###	
###SYS_SETFIXED_[*fixkey*]###	

FORM conventions
The forms used with fe_adminLib should be named after the table their are supposed to edit. For instance if you are going to edit records in the table 'fe_users' you must use a FORM-tag like this:

```
<FORM name="fe_users_form" method="POST" action="....">
```

The fields used to submit data for the records has this syntax, FE[*tablename*][*fieldname*]. This means, if you want to edit the 'city' field of a tt_address record, you could use a form element like this:

```
<INPUT name="FE[tt_address][city]">
```

Submit buttons can be named as you like except using the name "doNotSave" of a submit button will prevent saving. If you need Cancel button, please resort to JavaScript in an onClick even to change document.location.

Common markers
```
###GW1B### / ###GW1E###: Global wrap 1, begin and end (headers)
###GW2B### / ###GW2E###: Global wrap 2, begin and end (bodytext)
###GC1### / ###GC2### / ###GC3###: Global color 1 through 3

###FORM_URL###: The url used in the forms: index.php?id=page-id&type=page-type
###FORM_URL_ENC###: As above, but rawurlencoded.
###BACK_URL###: The backUrl value. Set to the value of incoming "backURL" var
###BACK_URL_ENC###: As above, but rawurlencoded.
```

```
###REC_UID###: The UID of the record edited. Set to the value of incoming "rU" var
###AUTH_CODE###: The "aC" incoming var
###THE_PID###: The "thePid" value - where the records are stored.
###THIS_ID###: Set to the current page id
###THIS_URL###: Set to the current script url as obtained by t3lib_div::getThisUrl()
###HIDDENFIELDS###: A bunch of hiddenfields which are required to be inserted in the forms. These
includes by default 'cmd', 'aC' and 'backURL'
```

In addition you can in most cases use markers like this

```
###FIELD_[fieldname]###
```

where [fieldname] is the name of a field from the record. All fields in the record are used.

Finally you can insert cObjects defined in TypoScript with this series of markers (see .cObject property in table above):

```
###CE_[cObjectName]###
###PCE_[cObjectName]###
```

(###PCE_* is difference from ###CE_* cObjects by the fact they are rendered with a newly created cObj (as opposed to the parant cObj of fe_adminLib) where the data-array is loaded with the value of ->dataArr which is the array submitted into the script. This makes then useful for presenting preview data. Finally both PCE_ and CE_ types cObject markers may be used with each single element in a edit menu (list of available records) by prefixing the marker with 'ITEM_', eg. ###ITEM_PCE_[cObjectName]###

Evaluation of the form fields
Printing error messages for REQUIRED fields

When a form template is displayed all subparts with the markers

```
    ###SUB_REQUIRED_FIELDS_WARNING###
```
and
```
    ###SUB_REQUIRED_FIELD_[fieldname]###
```

are removed. If there is a simple "required"-error (a field is not filled in) then the SUB_REQUIRED_FIELDS_WARNING is not removed and thus the error message contained herein is shown.

Lets say that more specifically it's the 'email' field in a form which is not filled in. Then you can put in a subpart named

```
    ###SUB_REQUIRED_FIELD_email###
```
This is normally removed, but it'll *not* be removed if the email field fails and thus you are able to give a special warning for that specific field.

Printing other error messages

However you may use other forms of evaluation than simple "required" check. This is specified for "create" and "edit" modes by the properties ".evalValues.*[fieldname]* = *[list of codes]*". In order to tell your website user *which* of the possible evaluations went wrong, you can specify error messages by the property .evalErrors which will be inserted as the marker named ###EVAL_ERROR_FIELD_*[fieldname]*###.

Lets say that you have put the code 'uniqueLocal' in the list of evaluation code for the email field. You would do that if you want to make sure that no email address is put into the database twice. Then you may specify that as:

```
  create.evalValues {
    email = uniqueLocal, email
  }
```

Then you set the evaluation error messages like this:

```
  evalErrors.email {
    uniqueLocal = Apparently you're already registered with this email address!
    email = This is not a proper email address!
  }
```

If the error happens to be that the email address already exists the field ###EVAL_ERROR_FIELD_email### will be substituted with the error message "Apparently you're already registered with this email address!".

Passing default values to a form
You can pass default values to a form by the same syntax as you use in the forms. For instance this would set the name and email address by default:

```
...?FE[tt_address][name]=Mike%20Tyson&FE[tt_address][email]=mike@trex.us&doNotSave=1&noWarnings=1
```

Notice the blue value names are the field values (must be rawurlencoded. In javascript this function is called escape()) and the red values are necessary if you want to NOT save the record by this action and NOT to display error messages if some fields which are required is not passed any value.

List of eval-codes

Eval-code:	Description:
uniqueGlobal	This requires the value of the field to be globally unique, which means it must not exist in the same field of any other record in the current table.
uniqueLocal	This is like uniqueGlobal, but the value is required to be unique *only* in the PID of the record. Thus if two records has different pid values, they may have the same value of this field.
twice	This requires the value of the field to match the value of a secondary field name [fieldname]_again sent in the incoming formdata. THis is useful for entering password. Then if your password field is name "user_pass" then you simple add a second field name "user_pass_again" and then set the 'twice' eval code.
email	Requires the field value to be an email address at least on the form [name]@*[domain].[tld]
required	Just simple required (trimmed value). 0 (zero) will evaluate to false!
atLeast atMost	Specifies a minimum / maximum of characters to enter in the fields. **Example**, that requires at least 5 characters: atleast [5]
inBranch	inBranch requires the value (typically of a pid-field) to be among a list of page-id's (pid's) specified with the inBranch parameters. The parameters are given like *[root_pid; depth; beginAt]* **Example**, which will return a list of pids one level deep from page 4 (included): inBranch [4;1]
unsetEmpty	This evaluation does not result in any error code. Only it simply unsets the field if the value of the field is empty. Thus it'll not override any current value if the field value is not set.

[tsref:(script).fe_adminLib.evalErrors.(field).(evalCode)]

Uploading files

fe_adminLib is able to receive files in the forms. However there currently are heavy restrictions on how that is handled. Ideally the proces would be handled by the t3lib_tcemain class used in the backend. In fact this could have been deployed but is not at this stage. The good thing about tcemain.php is that it perfectly handles the copying/deletion of files which goes into a certain field and even handles it independent of the storing method be it a list of filenames or use MM-relations to records (see tables.php section in 'Inside Typo3').

This is how files are handled by fe_adminLib and the restrictions that apply currently:

• You can upload files ONLY using "create" mode of a record. In any case you cannot edit currently attached files (this may be improved in the future). You can however use 'delete' mode.

• However you can use PREVIEW mode with 'create'. Works like this: if the mode is preview the temporary uploaded file is copied to a unique filename (prepended with the tablename) in typo3temp/ folder. Then the field value is set to the filenames in a list. When the user approves the content of the preview those temporary files are finally copied to the uploads/* folder (or whereever specified in TCA). Limitations are that the temporary files in typo3temp/ are NOT deleted when copied to the real upload-folder (this may be improved) and certainly not if the user aborts (can't be improved because the user may go anywhere). If the user cancels the preview in order to change values, the files will need to be uploaded again (this may be improved).

• The TCA extensions allowed for the field is ignored! However you can specify a list of extensions of allowed for the files in the .parseValues property of fe_adminLib

• The TCA filesize limitation for the field is ignored! However you can specify a max file size in kb in the .parseValues property of fe_adminLib

• Works only on fields configured for comma-list representation of the filenames (non-MM, see "Inside Typo3" document on MM relations for files).

It's recommended to use a dedicated folder for files administered by the fe_adminLib. The Typo3 testsite does that by using the uploads/photomarathon/ folder for images. This makes it much easier to clean up the mess if files and their relations to the records are broken.

Fieldnames for files

Lets say you have a field named "picture" of a table name "user_cars", the form-element should look like this:

```
<input type="file" name="FE[user_cars][picture][]">
```

If you wish to upload multiple files to that field, the form-elements should look like:

```
<input type="file" name="FE[user_cars][picture][]">
<input type="file" name="FE[user_cars][picture][]">
<input type="file" name="FE[user_cars][picture][]">
```

Use blob-types for the file-fields and reserve a minimum of 32 characters pr. filename.

NOTE: Make sure to always add the last square brackets ('...[]') to the fieldname! Otherwise it will not work!

tipafriendLib.inc

Files:

File:	Description:
tipafriendLib.inc	Main class used to display the Tip-a-Friend form Call it from a USER cObject with 'userFunc = user_tipafriend->main_tipafriend'
tipafriend_template.tmpl	Example template file.

Description

Example:
(See static_template 'plugin.tipafriend' for a working configuration)

Static template
plugin.tipafriend

tipafriendLib.inc properties

Property:	Data type:	Description:	Default:
templateFile	resource	The template-file. See example in 'media/scripts/tipafriend_template.tmpl'	
code	string / stdWrap	Code to define, what the script does. Case sensitive.	
defaultCode	string	The default code (see above) if the value is empty. By default it's not set and a help screen will appear	
wrap1	-> stdWrap	Global Wrap 1. This will be splitted into the markers ###GW1B### and ###GW1E###. Don't change the input value by the settings, only wrap it in something. **Example:** wrap1.wrap = <B> |</B>	
wrap2	-> stdWrap	Global Wrap 2 (see above)	
color1	string /stdWrap	Value for ###GC1### marker (Global color 1)	
color2	string /stdWrap	Value for ###GC2### marker (Global color 2)	

Property:	Data type:	Description:	Default:
color3	string /stdWrap	Value for ###GC3### marker (Global color 3)	
typolink	->typolink	TypoLink configuration for the TIPLINK to the TIPFORM page. . additionalParams is added the parameter "&tipUrl="	
htmlmail	boolean	If set, the page is fetched as HTML and send in HTML (a plain text version is sent as well).	

[tsref:(script).tipafriend]

plaintextLib.inc

Files:

File:	Description:
plaintextLib.inc	Main class used to display plain text content Call it from a USER cObject with 'userFunc = user_plaintext->main_plaintext'
plaintext_content.tmpl	Example template file.

Description

Example:
(See static_template 'plugin.alt.plaintext' for a working configuration)

Static template
plugin.alt.plaintext

plaintextLib.inc properties

Property:	Data type:	Description:	Default:
siteUrl	url	Url of the site.	
defaultOutput	untrimmed string	Default output if CType is not rendered.	
uploads.header	untrimmed string	Header for uploads	
images.header	untrimmed string	Header for images	
images.captionHeader	untrimmed string	Header for imagecaptions	
images.linkPrefix	untrimmed string	Prefix for image-links	
.header			
defaultType	int	Which type to use as default	
date	date-config	For header date	
datePrefix	untrimmed string	Prefix for header date	
linkPrefix	untrimmed string	Prefix for header links	
[1-5].preLineLen	int	Lenght of line before header	
[1-5].postLineLen	int	Lenght of line after header	
[1-5].preBlanks	int	Number of blank lines before header	
[1-5].postBlanks	int	Number of blank lines after header	
[1-5].stdWrap	->stdWrap	for header text	
[1-5].preLineChar	string	Character to pre-line	
[1-5].postLineChar	string	Character to post-line	
[1-5].preLineBlanks	int	Number of blank lines between header and pre-line	
[1-5].postLineBlanks	int	Number of blank lines between header and post-line	
[1-5].autonumber	boolean	If set, a number is prepended every header. The number corresponds to the content element number in the select.	
[1-5].prefix	untrimmed string	Header string prefix	
bulletlist.[0-3].bullet	untrimmed string	Bullet for bullet list, layout [0-3]	
bulletlist.[0-3].secondRow	untrimmed string	If set, this is used for lines on the second row of bullet-lists.	
menu	cObject	cObject to render menu. The output is stripped for tags and the links is extracted. Further all chars are converted to chr(10)	

Property:	Data type:	Description:	Default:
shortcut	cObject	cObject to render other elements. See config below which simply uses this object to render more tt_content elements as plaintext.	
bodytext.stdWrap	->stdWrap	stdWrap for body-text. See config example below.	
userProc	function-name	Lets you proces the output of each content element before it finally is returned. Property "parentObj" of the conf-array holds a references to the plainText object calling the function.	

[tsref:(script).plaintextLib]

Datatype 'untrimmed string' means that you can enter a string as usual, but if you enter a value between two vertical lines, that value will be used and NOT trimmed. Normally values are trimmed.

Example:

```
lib.renderObj = USER
lib.renderObj.userFunc = user_plaintext->main_plaintext
lib.renderObj {
  header.defaultType = 1
  header.date = D-m-Y
  header.datePrefix = |Date: |
  header.linkPrefix = | - Headerlink: |
  header.1.preLineLen = 76
  header.1.postLineLen = 76
  header.1.preBlanks=1
  header.1.stdWrap.case = upper

  header.2 < .header.1
  header.2.preLineChar=*
  header.2.postLineChar=*

  header.3.preBlanks=2
  header.3.postBlanks=1
  header.3.stdWrap.case = upper

  header.4 < .header.1
  header.4.preLineChar= =
  header.4.postLineChar= =
  header.4.preLineBlanks= 1
  header.4.postLineBlanks= 1

  header.5.preBlanks=1
  header.5.autonumber=1
  header.5.prefix = |: >> |


  siteUrl = {$plugin.alt.plaintext.siteUrl}
  defaultOutput (
|
[Unrendered Content Element; ###CType### ]
|
  )

  uploads.header = |DOWNLOADS:|

  images.header = |IMAGES:|
  images.linkPrefix = | - Imagelink: |
  images.captionHeader = |CAPTION:|

  bulletlist.0.bullet = |*  |

  bulletlist.1.bullet = |#  |

  bulletlist.2.bullet = | - |

  bulletlist.3.bullet = |>  |
  bulletlist.3.secondRow = |.  |
  bulletlist.3.blanks = 1

  menu = <tt_content.menu.20
  shortcut = <tt_content.shortcut.20
  shortcut.0.conf.tt_content = <lib.renderObj
  shortcut.0.tables = tt_content

  bodytext.stdWrap.parseFunc.tags {
    link < styles.content.parseFunc.tags.link
    typolist = USER
    typolist.userFunc = user_plaintext->typolist
    typolist.siteUrl = {$plugin.alt.plaintext.siteUrl}
    typolist.bulletlist < temp.renderObj.bulletlist
    typohead = USER
    typohead.userFunc = user_plaintext->typohead
    typohead.siteUrl = {$plugin.alt.plaintext.siteUrl}
    typohead.header < temp.renderObj.header
```

```
    typocode = USER
    typocode.userFunc = user_plaintext->typocode
    typocode.siteUrl = {$plugin.alt.plaintext.siteUrl}
  }
}
```

Standard Templates

static_template

This section of the TypoScript reference is used to introduce the standard templates that follows with Typo3 in the static table "static_template". You should not alter this table yourself but rather submit suggestions via the www.typo3.com-website if you want to correct errors or add templates or other pieces of TypoScript.

The "static_template" is published in new versions. The old records in the static_template are NOT changed from version to version (when finally released) unless they are under development and explicitly tagged with a note saying they are still not fixed! Still changes may appear though as long as The Typo3 project is not finally released!

Media

The standard templates uses some standard media-files, likes gif-images and fonts. These are situated in the folder "media/" relative to the root of the Typo3-website.

PHP include scripts

Introduction

Although you can do very much with TypoScript itself, it can sometimes be a much more flexible solution to include a PHP-script you write on your own. But you must understand and respect som circumstances. For example the caching system: When a page is shown with Typo3 it's normally cached afterwards in the SQL-database. This is done to ensure a high performance when delivering the same page the next time. But this also means that you can only make custom code from your include files if you differ your output based on the same conditions that the template may include! Fx. you cannot just return browser-specific code to TypoScript if not the template also distinguish between the actual browsers. If you do, the cache will cache the page with the browser-specific HTML-code and the next hit by another browser will trigger the cache to return a wrong page. If the condition is correctly setup "another browser"-hit will instead render another page (which will also be cached but tagged with the other browser!) an the two browsers will receive different pages but still the pages will be cached.

Including your script

Your script is included by a function, PHP_SCRIPT, inside the class "tslib_cObj" in the "tslib_content.php" script. Thereby your file is a part of this object (tslib_cObj) and function. This is why you must return all content in the variable "$content" and any TypoScript-configuration is available from the array "$conf" (it may not be set at all though so check it with is_array()!)

$conf

The array $conf contains the configuration for the PHP_SCRIPT cObject. Try debug($conf) to see the content printed out for debugging!

$content

Return all content in this variable.

Remember, don't output anything (but debug code) in your script!

Whitespace

Because nothing is sent off to the browser before everything is rendered and returned to index_ts.php which originally set of the rendering process, you must ensure that there's no whitespace before and after your <?...?> tags in your include- or library-scripts!

$GLOBALS["TSFE"]->set_no_cache()

Call the function $GLOBALS["TSFE"]->set_no_cache(), if you want to disable caching of the page. Call this during development! And call it, if the content you create may not be cached.

NOTE: If you make a syntax error in your script that keeps PHP from executing it, then the $GLOBALS["TSFE"]->set_no_cache() function is not executed and the page *is* cached! So in such situations, correct the error, clear the page-cache and try again. This is true only for PHP_SCRIPT and not PHP_SCRIPT_INT and PHP_SCRIPT_EXT which are rendered *after* the cached page!

Example:
```
$GLOBALS["TSFE"]->set_no_cache();
```

$this->cObjGetSingle(value , properties)

Gets a content-object from the $conf-array. (See the casestory on how to use this!)

Example:
```
$content=$this->cObjGetSingle($conf["image"], $conf["image."]);
```

This would return any IMAGE-cObject at the property "image" of the conf-array for the include-script!!

$this->stdWrap(value, properties)

stdWrap's the content "value" due to the configuration of the array "properties".

Example:
```
$content = $this->stdWrap($content, $conf["stdWrap."]);
```

This will stdWrap the content with the properties of ".stdWrap" of the $conf-array!

Internal Vars in the main frontend object, TSFE (TypoScript Front End)

There are some vars in the global object, TSFE, you might need to know about. These ARE ALL READ-ONLY!! (Read: Don't change them!). See the class tslib_fe for the full descriptions.

You access them like this example with "id": $GLOBALS["TSFE"]->id

Var:	PHP-Type:	Description:	Default:
id	int	The page id	
type	int	The type	
page	array	The pagerecord	
fe_user	object	The current front-end user. Userrecord in $GLOBALS["TSFE"]->fe_user->user, if any login.	
loginUser	boolean	Flag indicating that a front-end user is logged in.	0
rootLine	array	The rootLine (all the way to tree root, not only the current site!). Current site root line is in $GLOBALS["TSFE"]->tmpl->rootLine	
sys_page	object	The object with pagefunctions (object) See t3lib/page.php	
gr_list	string (list)	The group list, sorted numerically. Group -1 = no login	
beUserLogin	boolean	Flag that indicates if a Backend user is logged in!	0

Global vars

Var:	PHP-Type:	Description:	Default:
BE_USER	object	The back-end user object (if any)	not set
TYPO3_CONF_VARS	array	Typo3 Configuration	
TSFE	object	main frontend object.	

Casestory:

This is a casestory of how to use include-scripts.

In this situation we would like to use some libraries of our very own, not part of Typo3. Therefore we use the feature of including a library at the very beginning of the page-parsing.

First we put this TypoScript line in the "Setup"-field of the template:

```
config.includeLibrary = fileadmin/scripts/include.inc
```

The file **include.inc** is now included (in tslib/pagegen.php). In this case it looks like this:

file: fileadmin/scripts/include.inc

```
<?
    ...
    include("fileadmin/scripts/hello_world.inc");
    include("fileadmin/scripts/other_library.inc");
    ...
?>
```

As you can see, this file includes our library "hello_world" and some other libraries too!

The file **hello_world.inc** looks like this:

file: fileadmin/scripts/hello_world.inc

```
<?
class hello_world {
    function theMessage ()          {
        return "Hello World";
    }
}
?>
```

So far nothing has happend, except our libraries are included, ready for use.

Now we need to use the outcome of the hello_world class somewhere on a page. So in the TypoScript code we setup a content-object that includes the third script:

```
page.100 = PHP_SCRIPT
page.100.file = fileadmin/scripts/surprise.inc
```

surprise.inc looks like this:

file: fileadmin/scripts/surprise.inc

```
<?
    $hello_world_object = new hello_world;                    // New instance is created
    $contentBefore = $this->cObjGetSingle($conf["cObj"],$conf["cObj."]);
    $content = $contentBefore.$hello_world_object->theMessage();
    $content = $this->stdWrap($content,$conf["stdWrap."]);
?>
```

Line 1: The PHP-object $hello_world_object is created.

Line 2: This fetches the content of a cObject, "cObj", we defined

Line 3: The result of line 2 is concatenated with the result of the "theMessage"-function of the $hello_world_object object

Line 4: Finally the content is stdWrap'ed with the properties of ".stdWrap" of the $conf-array.

The output:

With this configuration -

```
page.100 = PHP_SCRIPT
page.100.file = fileadmin/scripts/surprise.inc
```

- the output will look like this:

Hello World

With this configuration -

```
page.100 = PHP_SCRIPT
page.100 {
      file = fileadmin/scripts/surprise.inc
      cObj = TEXT
      cObj.value = Joe says: 
}
```

- the output will look like this:

Joe says: Hello World

With this configuration -

```
page.100 = PHP_SCRIPT
page.100 {
      file = fileadmin/scripts/surprise.inc
      cObj = TEXT
      cObj.value = Joe says: 
      stdWrap.wrap = <font color="red"> | </font>
      stdWrap.case = upper
}
```

- the output will look like this:

JOE SAYS: HELLO WORLD

End of lesson.

Storing user-data or session-data

Doing so is quite simple with Typo3.

Userdata is data, that follows login users. As soon as a login user is logged out, these data are no more accessible and cannot be altered.

Session data is data, that follows the user currently browsing the site. This user may be a login-user, but his session-data is bound to the "browsing-session" and not to the user-id of his. This means, that the very same person will carry this data still, even if he logs out. As soon as he closes his browser, his data will be gone though.

Also you should know, that session-data has a default expire-time of 24 hours.

Retrieving and storing user-/session-data is done by these functions:

$GLOBALS["TSFE"]->fe_user->getKey(type, key)

"type" is either "user" or "ses", which defines the data-space, user-data or session-data

"key" is the "name" under which your data is stored. This may be arrays or normal scalars.

Note that the key "recs" is reserved for the built-in "shopping-basket". As is "sys" (for Typo3 standard modules and code)

Example:

```
if ($GLOBALS["TSFE"]->loginUser)        {
```

```
      $myData = $GLOBALS["TSFE"]->fe_user->getKey("user","myData");
} else {
      $myData = $GLOBALS["TSFE"]->fe_user->getKey("ses","myData");
}
```

This gets the stored data with the key "myData" from the user-data, but if no user is logged in, it's fetched from the session data instead.

$GLOBALS["TSFE"]->fe_user->setKey(type, key, data)

"type" is either "user" or "ses", which defines the data-space, user-data or session-data

"key" is the "name" under which your data is stored.

Note that the key "recs" is reserved for the built-in "shopping-basket". As is "sys" (for Typo3 standard modules and code)

"data" is the variable, you want to store. This may be arrays or normal scalars.

Example:

```
$myConfig["name"] = "paul";
$myConfig["address"] = "Main street";
$GLOBALS["TSFE"]->fe_user->setKey("ses","myData", $myConfig);
```

This stores the array $myConfig under the key "myData" in the session-data. This lasts as long as "paul" is surfing the site!

Using the built in "shopping basket"

Typo3 features a shopping basket for the session-data.

When you submit data from forms (or by querystring) (post/get-method) in the array "recs" it's stored in the session-data under the key recs.

The syntax is like this:

```
      recs[table_name][uid_of_record]
```

Example:

This form-element will change the registered value of record with uid=345 from the "tt_products" table in typo3. Please note, that the record itself is NOT in any way modified, only the "counter" in the session-data indicating the "number of items" from the table is modified.

```
      <input name="recs[tt_products][345]">
```

NOTE on checkboxes:

When you are creating forms with checkboxes, the value of the checkbox is sent by MSIE/Netscape ONLY if the checkbox is checked! If you want a value sent in case of a disabled checkbox, include a hidden formfield of the same name just before the checkbox!

Example:

```
 <INPUT type="hidden" name="recs[tt_content][345]" value="0">
 <INPUT type="checkbox" name="recs[tt_content][345]" value="1">
```

Clearing the "basket":

This will clear the basket:

```
 <INPUT type="hidden" name="recs[clear_all]" value="1">
```

index.php

Introduction

index.php is the main script for showing pages with Typo3 / TypoScript. This page show some information about this script and how to use it.

Normally you request pages by setting a value for "id" and possibly "type".

"id" refers to a page. This is an integer. If a string is supplied, it's regarded as an alias and the corresponding page is found.

"type" defines which "type" the page is. Always an integer (0-255). If "type" is not set it's regarded to be zero. "type" is used to build framesets. Fx. the frameset would have "type=0" (or nothing) and the pages in the various frames would have "type=1" and "type=2" and "type=3". In TypoScript you define a PAGE-object for each type so Typo3 renders different pages depending on the type-value. Normally the PAGE-object displaying the page content is named "page" and has the "type=1" value.

Submitting data to index.php

You can submit data to index.php for several reasons. These are the standard features included in the script

Login/Logout:

Detected by class "t3lib_userauth" looking for the var "logintype". If this is set, authentication is done.

Input may be of both GET and POST method.

Login:

logintype = "login"

pass = the password

user = the username

pid = the id of the page where the user-archive is found. You don't need this value if the TYPO3_CONF_VARS[FE][checkFeUserPid] is set.

(redirect = No use)

Logout:

logintype = "logout"

See the cObject FORMS for an in-depth description

Search:

Detected by the cObject SEARCHRESULT, which proceeds with a search if "sword" && "scols" are set. The search MUST submit to a page with such a content-object on!

Input may be of both GET and POST method.

Search:

sword = the searchwords

stype = the search type

scols = the tables/columns to search

locationData = Reference to the record carrying the form. Used to look up the original startingpoint of the search (ONLY POST-method)

(redirect = No use)

scount = Used by the searchresult to indicate the number of results

spointer = Used by the searchresult to indicate the startingpoint for the next number of results.

See the cObject SEARCHRESULT for a complete description

Emailforms:

Detected by the mainscript "index.php" looking for the var "formtype_mail" to be set. (could be the submit-button)

Input MUST be POST method. And the REFERER and HTTP_HOST must match. Also the locationData var must be sent and at least point to the uid of a readable page.

Database-submit

Detected by the mainscript "index.php" looking for the var "formtype_db" to be set. (could be the submit-button)

Input MUST be POST method. And the REFERER and HTTP_HOST must match. To setup a script to handle the input, refer to the FE_DATA object.

See examples from the tslib/media/scripts/ folder, eg. "guest_submit.inc"

Tips

Preparing pages for Direct Mail sending

When you're designing HTML-newsletter templates, there are some good things to keep in mind. First is to enable 'config.insertDmailerBoundaries' so you can categorize the content. In addition using media in your mail is perfectly OK as long as the url set by the href-parameter is encapsulated in double-quotes or not encapsulated. Using single-quotes, the capture function will not find your image.

If you place forms in the direct mail, you should use GET method. POST method may not transfer data to the page. In addition form-urls are not directed through the jumpurl feature but rather directly to the target page.

A nice trick is to place a little clear-gif image (use a filename other than 'clear.gif' though!) and put the parameter dmailerping=1 in the tag. This will force the capture function to set the url of this image absolute through the jumpurl registration. This means in other words that when this mail is *opened* it will be registered (response_code=2). This is an additional feature to the regular feature which registeres all links clicked in the mail (response_code=1).

Example:
```
<img src=dmailerping.gif width=1 height=1 dmailerping=1>
```

Preparing pages for indexing

The Typo3 Indexing 'robot' will detect if any HTML contains the comments <!--TYPO3SEARCH_begin--> and <!--TYPO3SEARCH_end-->.

If '_begin' is found first, then the preceeding code is excluded and everything until the next _end is included etc.

If '_end' is found first, then the preceeding code is included, everything after is excluded until the next _begin from which code is again included etc.

If neither '_begin' or '_end' is found everything is included in the indexed content.