Administrators Guide

October 5th 2002

Typo3 3.5b3

Table of content

Users and groups

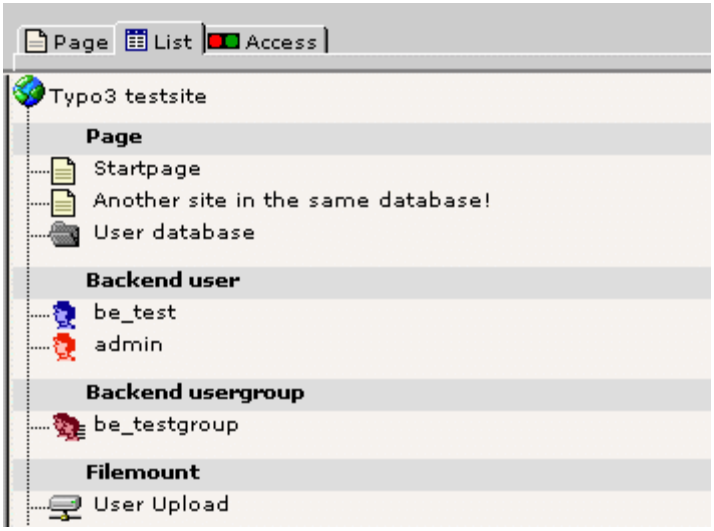
Introduction

For the formal description of permissions in Typo3, see the document "Inside Typo3".

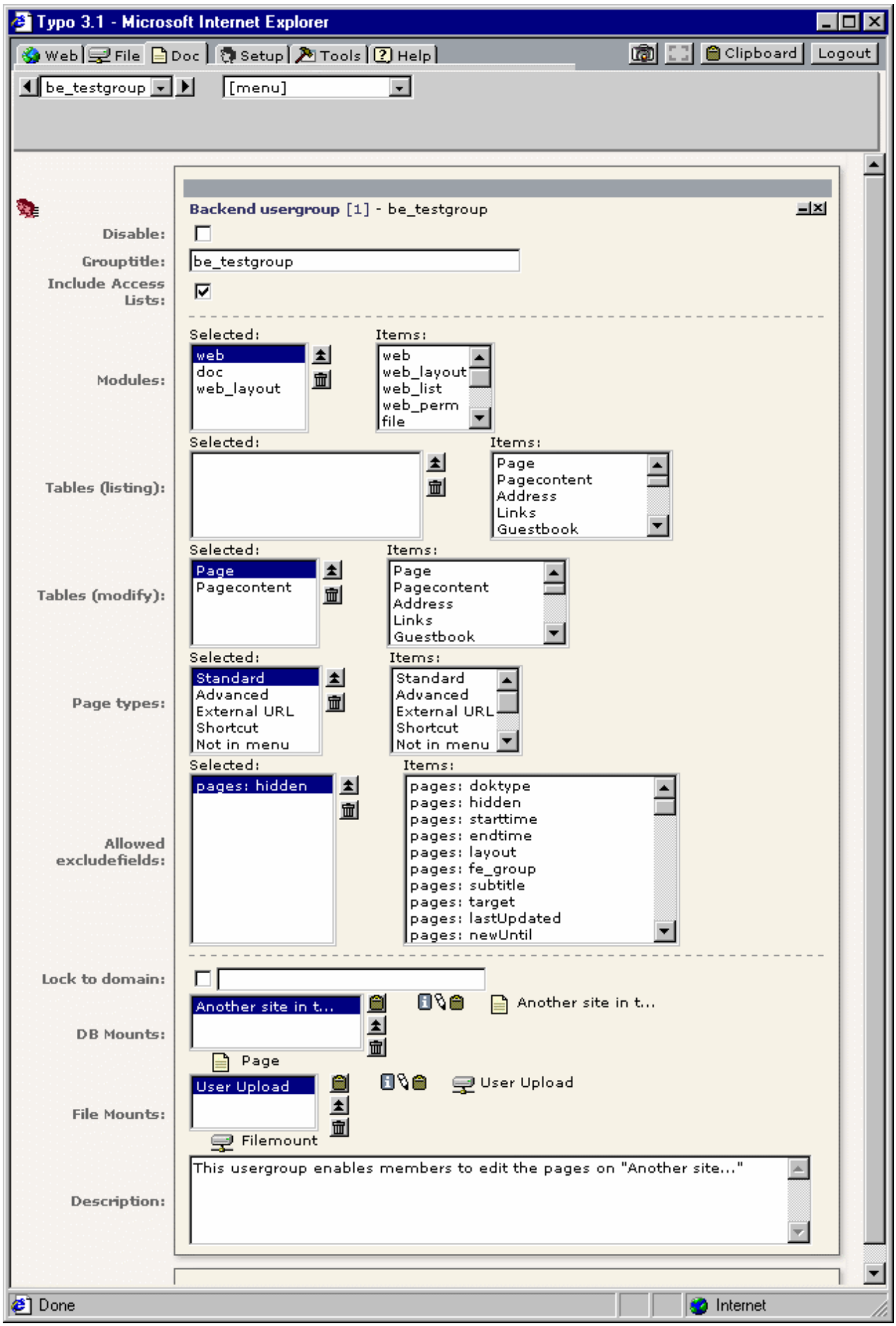
Creating users and groups

Do like this:

- In the root of the pagetree, create a "backend usergroup" (the blue ones...)



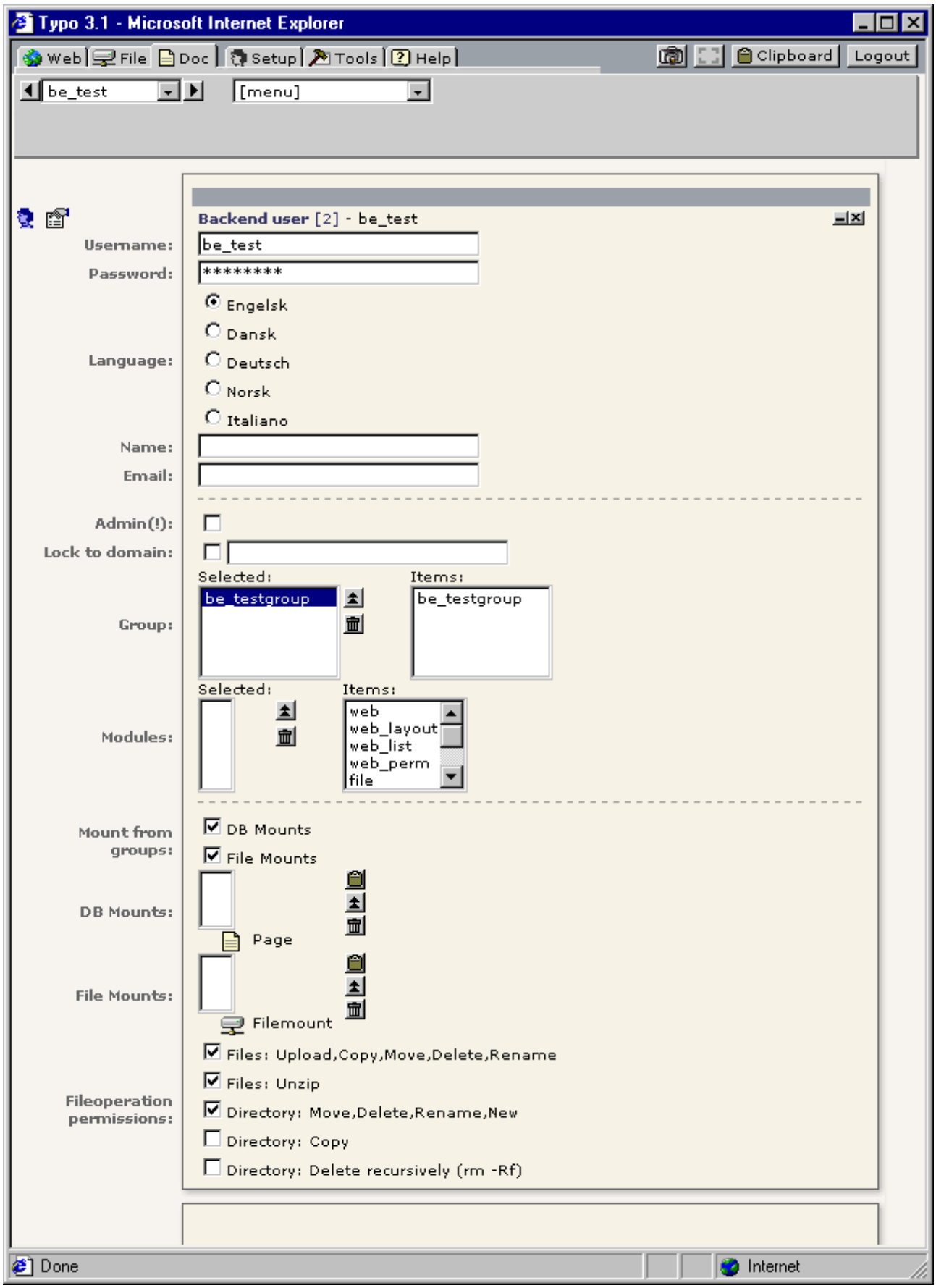
- Modify the record like this:


Typo 3.1 - Microsoft Internet Explorer
Web
File
Doc
Setup
Tools
Help
Clipboard
Logout
be_testgroup
[menu]
Backend usergroup [1] - be_testgroup
Disable:
Grouptitle:
be_testgroup
Include Access Lists:
Modules:
Selected:
web
doc
web_layout
Items:
web
web_layout
web_list
web_perm
file
Tables (listing):
Selected:
Items:
Page
Pagecontent
Address
Links
Guestbook
Tables (modify):
Selected:
Page
Pagecontent
Items:
Page
Pagecontent
Address
Links
Guestbook
Page types:
Selected:
Standard
Advanced
External URL
Shortcut
Not in menu
Items:
Standard
Advanced
External URL
Shortcut
Not in menu
Allowed excludefields:
Selected:
pages: hidden
Items:
pages: doktype
pages: hidden
pages: starttime
pages: endtime
pages: layout
pages: fe_group
pages: subtitle
pages: target
pages: lastUpdated
pages: newUntil
Lock to domain:
DB Mounts:
Another site in t...
Another site in t...
Page
File Mounts:
User Upload
User Upload
Filemount
Description:
This usergroup enables members to edit the pages on "Another site..."
Done
Internet

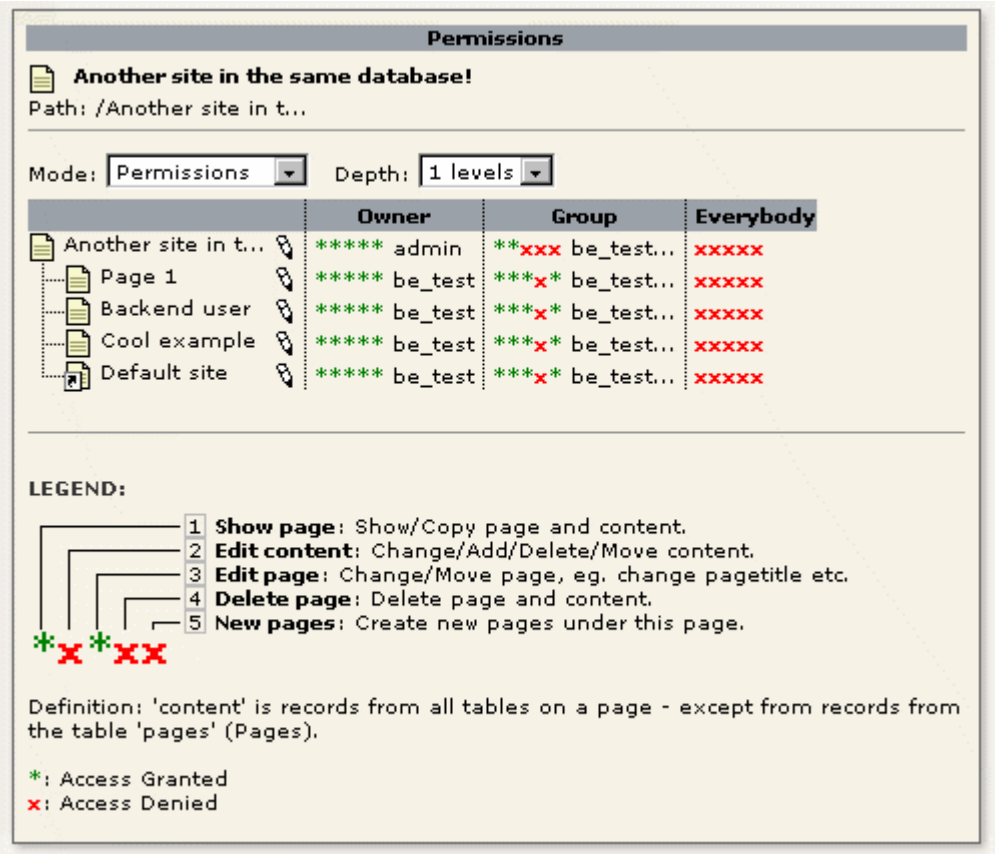
The DBmount is the rootpoint for that user!

The filemount is a filemount-record also created in the root of the pagetree. This is needed if the users needs to upload files. In the filemount-dir a "_temp_" folder must exist as a default upload-location.

Create a back-end userrecord, also in the root of the page-tree.



- Be shure to change the permissions. The new user MUST have at least read-access to the pages!



That should make it work.

TSconfig

Introduction

TypoScript is mainly known in relation to building templates for the frontend of Typo3 made websites. However TypoScript itself is basically a way to define values in a hierarchy. And so TypoScript is also used to make detailed configuration of the backend, both regarding users and groups and the way modules are working. This is done through the 'TSconfig' which is found for both the backend and frontend users/groups in addition to the pages.

This section deals with the options you can set. Notice that because TypoScript is a 'declarational' language (that is, it's merely configurative values) it does not in inself carry out a sequence of actions. The reason why the TypoScript *templates* used for the frontend seems to work a little like this is because those values are interpreted into PHP function calls which constructs a page. In the case where TypoScript is used for configuration like here its clearly more like the Registration Database in windows: The values configure how the application works.

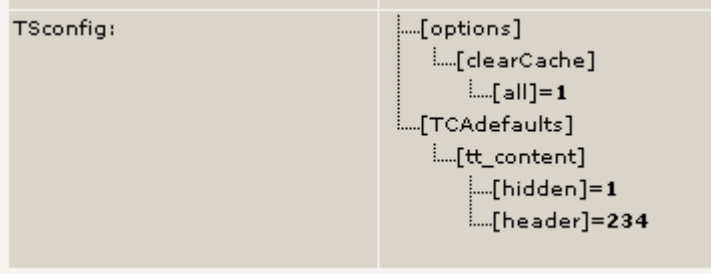
Syntax

The syntax is as usual TypoScript, only you cannot use Constants and Conditions.

User TSconfig

You can enter TSconfig for both backend users and groups. The resulting TSconfig for a certain backend user is the accumulated TSconfig for all the backend groups he's a member of. Thus you can override formerly set options from the usergroups by entering another configuration in the TSconfig field of the user himself.

It's vital to check the resulting configuration of the users. You can do that in the Tools>User Admin module by clicking a username. Then you'll see the TSconfig tree among other information. Here's an example:



This configuration could be set by entering this value in the TSconfig field of the backend user:

```
options.clearCache.all = 1
TCAdefaults.tt_content {
  hidden = 1
  header = 234
}
```

Now, lets say the user is a member of a *usergroup* with this configuration:

```
TCAdefaults.tt_content {
  hidden = 1
  header = Hello!
}
```

Then setting these values in the TSconfig field of the user himself, would override the default value of the header (marked red) and add clear cache option (marked blue). The default value of the hidden field is not changed and simply inherited directly from the group:

```
options.clearCache.all = 1
TCAdefaults.tt_content.header = 234
```

Main objects:

Property:	Data type:	Description:	Default:
admPanel	->ADMPANEL	Options regarding the front-end admin panel	
options	->OPTIONS	Options for the user, various	
mod	*(see ->MOD of Page TSconfig)*	Overriding values for the backend modules	
setup.defaults setup.override	->SETUP	Default values and override values for the user settings known from the setup module.	
TCAdefaults	[tablename].[field] = value	Sets default values for records. The order of default values when creating new records in the backend is this: • Value from tables.php • Value from userTS • Value from submitted global array, defVals (to load_edit.php / transferdata.php) • Value from previous record based on 'useColumnsForDefaultValues' (Loaded with transferdata.php to backend) However the order for default values used by tcemain.php if a certain field is not granted access to for user will be: • Value from tables.php • Value from userTS So these values will be authoritative if the user has no access to the field anyway. **Example:** This sets the default hidden flag for pages to "clear" `TCAdefaults.pages.hidden = 0`	
user	???	This is for custom purposed. You can use if for your own games...	

[beuser]

ADMPANEL:

Property:	Data type:	Description:	Default:
enable	[object]	Used to enable the various parts of the panel for users. All values are 0/1 booleans. **General:** .all: enables all modules **Modules:** .preview .cache .publish .edit .tsdebug .info	(For admin-users, all = 1 is default! Hardcoded in class)
hide	boolean	If set, the panel will not display in the bottom of the page. This has only a visual effect.	

Property:	Data type:	Description:	Default:

module.edit	[object]	**.forceDisplayIcons** (boolean):
		Forces edit-panels to appear regardless of the selectorbox.
		.forceDisplayFieldIcons (boolean):
		Forces edit-icons to appear regardless of the selectorbox.
		.forceNoPopup (boolean):
		Forces edit-forms to open in same window - not pop up window.
		
		Example, that forces the display of the edit icons without displaying the admin-panel itself:
		`admPanel {`
		`  enable.edit = 1`
		`  module.edit.forceDisplayFieldIcons = 1`
		`  hide = 1`
		`}`	

[beuser:admPanel]

Tip:

If you want to link to the login-screen of the backend, but wishes that the user should return to the front-end for editing, you do it by this link, sending the “redirect_url” parameter to the login-screen. In that case the backend interface is not started.

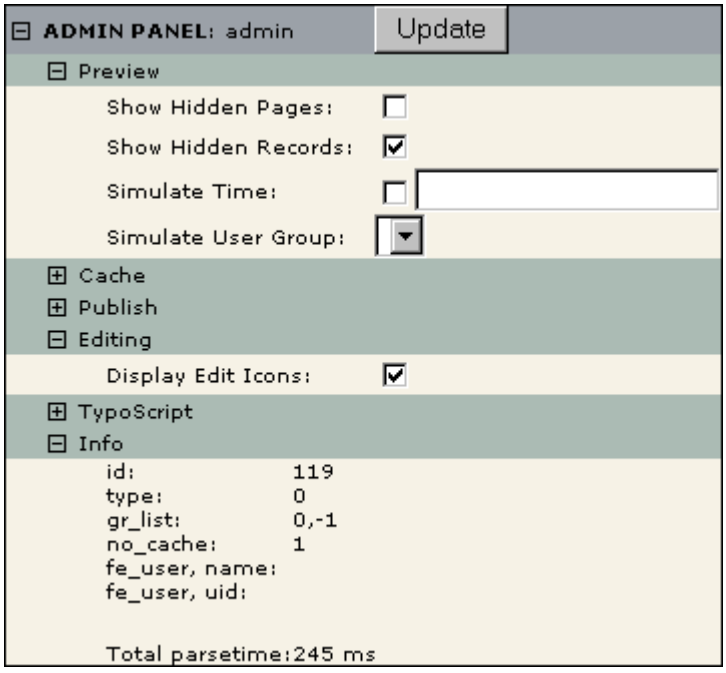
Backend login returning to front-end

This is how the admin panel looks. Notice the fact that the visibility of the admin panel is ultimately dependent on being configured in your TypoScript template for the website! This is however easily done by inserting this string in the template:

```
config.admPanel = 1
```

or if you use frames, this is probably better:

```
page.config.admPanel = 1
```



OPTIONS:

Some user-options

Property:	Data type:	Description:	Default:
RTEkeyList	[list of elements]	This is a list of the Rich Text Editor buttons the user may have displayed. The user will not sett any buttons not listed here. Either enter a list of these elements: cut,copy,paste,formatblock,fontstyle,fontsize,bold,italic,underline,left,center,right,orderedlist,unorderedlist,outdent,indent,line,link,table,image,textcolor or specify all with a wildcard "*" for everything chMode is also an element you can use and that will display an option to look at the source-code instead of WYSIWYG	* (If value is not set at all, *, is default)
clearCache.pages	boolean	This will allow a user to clear the whole page cache.	
clearCache.all	boolean	This will allow a user to clear all cache (that is everything including templates)	
lockToIP	string	List of IP-numbers with wildcards. **Note:** This option is enabled only if the TYPO3_CONF_VARS[BE][enabledBeUserIPLock] configuration is true. **Examples:** 192.168.*.* - will allow all from 192.168-network 192.168.*.*, 212.22.33.44 - will allow all from 192.168-network plus all from REMOTE_ADDR 212.22.33.44 192.168, 212.22.33.44 - the same as the previous. Leaving out parts of the IP address is the same as wild cards...	
saveClipboard	boolean	If set, the clipboard content will be preserved for the next login. Normally the clipboard content lasts only during the session.	
clipboardNumberPads	int (0-20)	This allows you to enter how many pads you want on the clipboard.	3
enableABClipboardInCB	boolean	If set, the Alternative Backend clipboard is enabled in the classic backend as well.	
disableDocModuleInAB	boolean	If set, the doc module is not displayed in the AB even if enabled in general.	
shortcutFrame	boolean	If set, the shortcut frame in the bottom of the window appears.	

shortcutGroups	Array of integers	Setting which shared shortCutGroups are loaded for the user.
		Example:
		`shortcutGroups {`
		`  1=1`
		`  2=1`
		`  3=`
		`}`
		Shared shortcut group 1 and 2 is loaded, but 3 is not loaded	

shortcut_onEditId_dontSetPageTree	boolean	If set, the page tree is NOT opened to the page being edited when an id number is entered in the "Edit Id" box	
shortcut_onEditId_keepExistingExpanded	boolean	If set, the existing expanded pages in the page tree are not collapsed when an id is entered in the "Edit Id" box. (provided .shortcut_onEditId_dontSetPageTree is not set!)	
mayNotCreateEditShortcuts	boolean	If set, then the user cannot create or edit shortcuts. Depends on .shortcutFrame being set.	
createFoldersInEB	boolean	If set, a createFolders option appears in the element browser (for admin-users this is always enabled).	
uploadFieldsInTopOfEB	boolean	If set, the upload-fields in the element browser are put in the top of the window.	
disableJSStop	boolean	If set, detection of JSStop is switched of this user	

Property:	Data type:	Description:	Default:

saveDocNew
saveDocNew.[table]
	boolean / “top”	If set, a button “Save and create new” will appear in TCEFORMs.
		Any value set for a single table will override the default value set to the object “saveDocNew”.
		Example:
		In this example the button is disabled for all tables, except tt_content where it will appear, and in addition create the records in the top of the page (default is after instead of top).
		`options.saveDocNew = 0`
		`options.saveDocNew.tt_content = top`	

disableDelete disableDelete.[table]		Disables the “Delete” button in TCEFORMs. Overriding for single tables works like “saveDocNew” above.	
showHistory showHistory.[table]		Shows link to the history for the record in TCEFORMs. Overriding for single tables works like “saveDocNew” above.	
pageTree.disableIconLinkToContextmenu folderTree.disableIconLinkToContextmenu	boolean / “titlelink”	If set, the page/folder-icons in the page/folder tree will not activate the clickmenu. If the value is set “titlelink” then the icon will instead be wrapped with the same link as the title. (AB only)	
contextMenu.[*key*].disableItems	list of items	List of context menu (“clickmenu”) items to disable. “key” points to which kind of icon that brings up the menu, and possible options are “pageTree”, “pageList”, “folderTree”, “folderList”. “page” and “folder” obviously points to either the Web og File main module. “Tree” and “List” points to whether the menu is activated from the page/folder tree or the listing of records/files Items to disable are (for “page” type - that is database records): view,edit,hide,new,info,copy,cut,paste,delete,move_wizard,history,perms,new_wizard,hide,edit_access,edit_pageheader,db_list Items to disable are (for “folder” type - that is files/folders): edit,upload,rename,new,info,copy,cut,paste,delete (AB only, the small topframe)	
contextMenu.options.leftIcons	boolean	If set, the icons in the clickmenu appears to the left instead of right	
contextMenu.options.clickMenuTimeOut	int, 1-100	Number of seconds the click menu is visible in the top frame before it disappears by it self.	5
contextMenu.options.alwaysShowClickMenuInTopFrame	boolean	If set, then the clickmenu in the top frame is always shown. Default is that it's shown only if the pop-up menus are disabled by user or by browser.	

[beuser:options]

SETUP:

Default values and overriding values for the setup-module.

Default values are set by 'setup.default' while overriding values are set by 'setup.override'. Overriding values will be impossible for the user to change himself and no matter the current value the overriding value will overrule. The default values are used for new users or if the setup is re-initialized.

Please notice that if you have first set a value (by override eg) and then REMOVE that value from being set, the value is NOT restored to the original default but is kept at the current value! Therefore setting a value and later removing that value woudl require the users preferences to be reset OR better, don't remove the value, just change the value of it! (eg. to a blank string if you wish to “reset” the value).

This table shows the keys for both defaults and override values:

Property:	Data type:	Description:	Default:
pane_clip_count	int+	Number of tabs on the clipboard	
pane_web	*list of string*	Page Tree Pane Labels (Classic Backend only)	
pane_file	*list of string*	Directory Tree Pane Labels (Classic Backend only)	
thumbnailsByDefault	boolean	Show Thumbnails by default	
emailMeAtLogin	boolean	Notify me by email, when somebody logs in from my account	
openDirectly	boolean	Open Typo3 Interface at Login	

Property:	Data type:	Description:	Default:
alternativeBackend	boolean	If set, the Alternative Backend will load instead of the classic by default.	
startInTaskCenter	boolean	If set, then the backend will start up in the task center (task center should be enabled for the user)	
localFrameEdit	boolean	If set, all edit-icons (except from clickMenus) will open the record in the local frame instead of loading a document in the doc-module.	
JSwindowParams	string	This string will override the system parameters opening the classic interface in a new window. By setting values here, you can fully customize the way Typo3 loads in the window. **Example:** status=1,menubar=0,resizable=1, screenX=1, screenY=1, left=1, top=1	
saveTreePositions	boolean	Remember position at Logout	
helpText	boolean	Show help text when applicable	
titleLen	int+	Max. Title Length	
edit_wideDocument	boolean	Wide document background	
edit_RTE	boolean	Enable Rich Text Editor	
edit_maxRecords	int+	Max number of content records to edit at a time	
edit_docModuleUpload	boolean	File upload directly in Doc. module	
navFrameWidth	int+	The width in pixels of the navigation frame in the Page and File main modules (Alternative Backend)	245 pixels
winH	int+	Typo3 Window, width (Classic Backend only)	
winW	int+	Typo3 Window, height (Classic Backend only)	
lang	language-key	One of the language-keys. See t3lib/config_default.php for current options. Eg. “dk”, “de”, “es” etc.	
copyLevels	int+	Recursive Copy: Enter the number of page sublevels to include, when a page is copied	
recursiveDelete	boolean	Recursive Delete(!): Allow ALL subpages to be deleted when deleting a page	
deleteCmdInClipboard	boolean	Allow 'Delete' command in the clipboard menu	
allSaveFunctions	boolean	Display all save functions in Doc-module menu	
neverHideAtCopy	boolean	If set, then the hideAtCopy feature for records in TCE will not be used.	
condensedMode	boolean	If set, the Alternative Backend will not load the Web-submodules and File-submodules in a frameset by allow the page and folder trees to load the submodule in it's own frame. This allows for a better display on small screens.	
noMenuMode	boolean / string	If set, the Alternative Backend will not load the left menu frame but rather put a selector-box menu in the topframe. This saves a lot of space on small screens. Also icons will not be displayed in the clickmenu panel in the top. **Value “icons”:** Setting noMenuMode to “icons” will still remove the menu, but instead of the selectorbox menu you will have the whole clickmenu panel as a menu with the icons only as the hidden state of the clickmenu panel. This is extremely nice (in my opinion) for experienced users who know the icons of the modules.	
dontEditInPageModule	boolean	From version 3.3 pages are edited in the Web>Page module and by default the clickmenu in the page tree will open the page there if the Web>Page module is enabled for the user. This can be disabled with this switch.	
classicPageEditMode	boolean	Setting this option will not open the Web>Page module but rather load the content elements (normal column/default language) together with the page header in one big form when a page is edited (clicking a page icon in the page tree). This simulates the old behaviour in Classic Backend	
hideSubmoduleIcons	boolean	(AB) If set then submodule icons will not be shown in the left menu of the Alternative Backend.	
noOnChangeAlertInTypeFields	boolean	When a record field is changed and that field would affect the layout of the form, a message normally pops up, saying that one should save now. If this flag is set, no message pops up.	

Property:	Data type:	Description:	Default:
dontShowPalettesOnFocusInAB	boolean	If set, palettes are not activated in the TCEFORMs (typ. AB) when focus is moved to a field.	

[beuser:setup.default/setup.override]

Notice:

Don't use any keys not listed here and don't add any properties!

Especially these keys are used for other purposes and should under no circumstances be used:

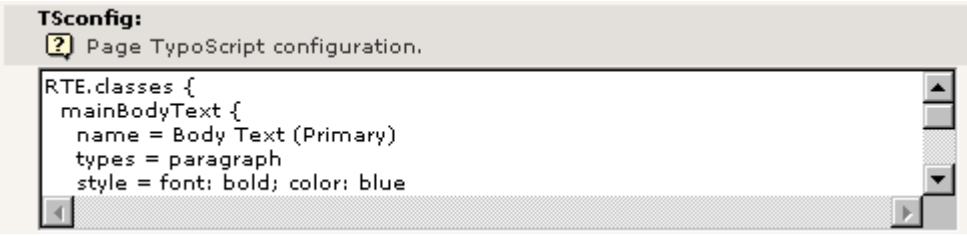
moduleData
interfaceSetup
moduleSessionID

Page TSconfig

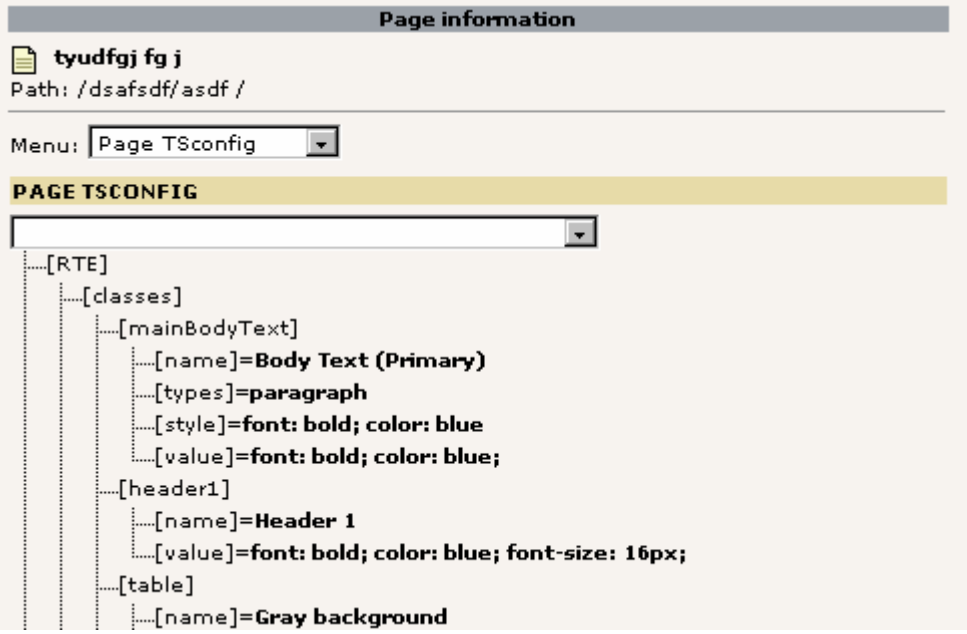
The Page TSconfig primarily concerns configuration of the modules in the Typo3 backend. It may also relate to values in the frontend display occationally.

The Page TSconfig for a page is accumulated from the root and extends to cover the whole branch of subpages as well (unless values are overridden further out).

This is an example of the TSconfig field with a snippet of configuration for the Rich Text Editor. Precisely the Rich Text Editor is quite a good example of the usefullness of 'Page TSconfig'. The reason is that you may need the RTE to work differently in different parts of the website. For instance you might need to offer other style-classes in certain parts of the website. Or some options might need to be removed in other parts. The 'Page TSconfig' is used to configure this.



If you need to check out the actual configuration for a certain branch in the website, use the 'Web>Info' module:



Main objects:

Property:	Data type:	Description:	Default:
mod	->MOD	Options for the modules (overridden by any similar values set for the individual user!)	
RTE	->RTE	This defines configuration for the Rich Text Editor	
TCEMAIN	->TCEMAIN	Defines configuration for the tce_main class (Typo3 Core Engine)	
TCEFORM	->TCEFORM	This defines extra configuration for the form fields rendered by the tceforms-class in general.	
TSFE	->TSFE	Options for the TSFE front end object	
user	???	This is for custom purposed. You can use if for your own games...	
[page]			

TSFE:

Property:	Data type:	Description:	Default:

jumpUrl_transferSession	boolean	If set, the jumpUrl redirection to the URL will be prepended with a parameter that transfers the current fe_users session to that url. This URL should be the Typo3 frontend in the same database, just at another domain (else it makes no sense).
		
		You can implement it in your own links if you like. This is how you do:
		You must send the parameter 'FE_SESSION_KEY' as GET or POST. The parameter looks like this: [fe_user-session-id]-[a hash made to prevent misuse]
		The parameter can be calculated like this:
		`$param = "&FE_SESSION_KEY=".rawurlencode(`
		`  $GLOBALS[“TSFE”]->fe_user->id."-".`
		`  md5(`
		`  $GLOBALS[“TSFE”]->fe_user->id."/".`
		`  $GLOBALS[“TYPO3_CONF_VARS”]["SYS"]["encryptionKey"]`
		`  )`
		`);`	

[page:TSFE]

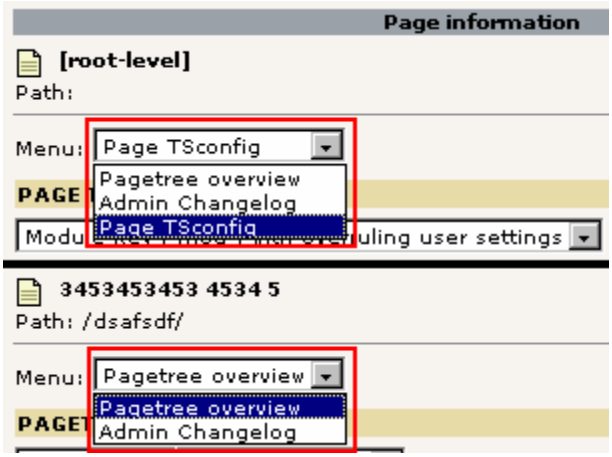
MOD:

Generally the syntax is *[module_name].[property]*. The module name is defined in the conf.php files for the module.

General configuration of the main menu in modules

Most of the modules in Typo3 has a main menu and this menu is usually configurable so you are able to remove menu items in specific sections of the page tree (or by overriding via User TSconfig, you could disable an option totally for a specific user/group).

In this case the main menu of the Web>Info module looks like this:



The latter example shows how the menu looks when configured like this:

```
mod.web_info.menu.function {
  tsconf = 0
}
```

The 'Page TSconfig' is simply disabled by setting this Page TSconfig!

All you need to know in order to disable menu items in the modules are, which modules has this option and what the *key* of the menu item is (in the above example it was 'tsconf'). This list is provided here:

Property:	Data type:	Description:	Default:
web_layout.menu.function		(Module Web>Page): 0 => “QuickEdit” 1 => “Columns” 2 => “Languages”	
web_info.menu.function		(Module Web>Info): page => “Pagetree overview” log => “Admin Changelog” tsconf => “Page TSconfig”	
web_func.menu.function		(Module Web>Functions): 1 => “Import” 2 => “Export” 3 => “Wizards”	
web_ts.menu.function		(Module Web>Template): 0 => "Info/Modify", 1 => "Constant editor", 2 => "TypoScript Object browser", 3 => "Template analyzer"	
user_task.menu.function [page:mod; beuser:mod]		(Module User>Task center): "tasks" => "Tasks", "messages" => "Messages", "note" => "Quick Note", "recent" => "Recent Pages", "mod" => "Modules", "actions" => "Actions"	

Overriding Page TSconfig with User TSconfig

In all standard modules the Page TSconfig values of the “mod.” branch may be overridden by the same branch of values set for the backend user.

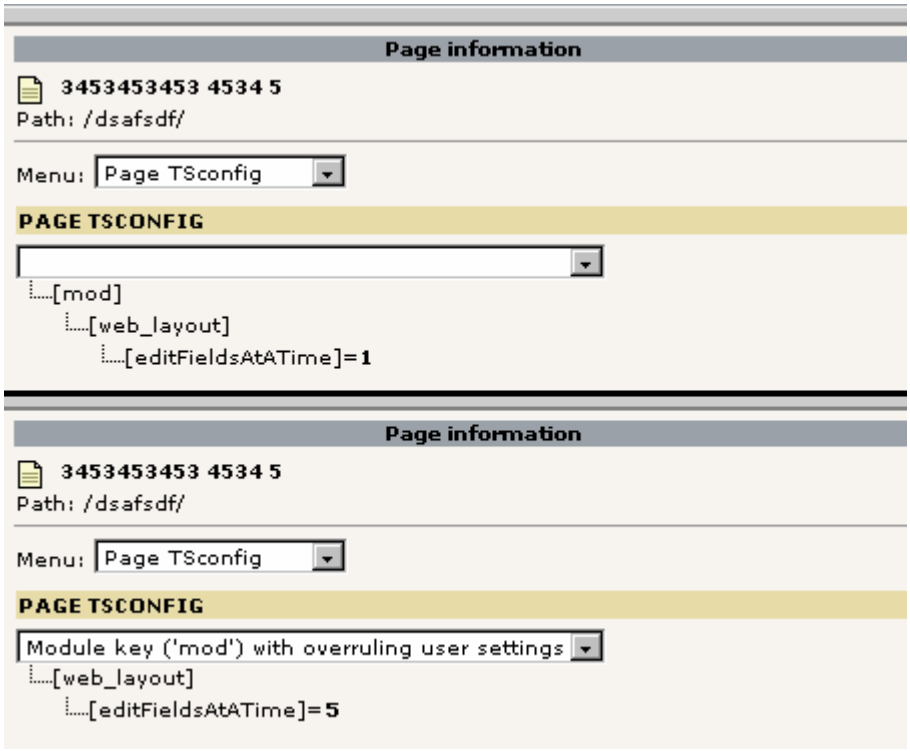
To illustrate this feature lets consider the case from above where a menu item in the Web>Info module was disabled in the Page TSconfig by this value:

```
mod.web_info.menu.function {
  tsconf = 0
}
```

If however we set this configuration in the TSconfig of a certain user (eg. the admin user), that user would still be able to select this menu item because the value of his User TSconfig overrides the same value set in the Page TSconfig:

```
mod.web_info.menu.function {
  tsconf = 1
}
```

Here's another example where the value 'mod.web_layout.editFieldsAATime' has been overridden by User TSconfig to the value '5' instead of '1'. The image shows you how to check the Page TSconfig (the upper image) and also see if User TSconfig of the current user (the logged in user) has altered anything from this configuration (lower image):



Specific module configurations

This is the specific options for certain modules:

Shared options for modules (mod.SHARED):

Property:	Data type:	Description:	Default:
colPos_list [page:mod.SHARED; beuser:mod.SHARED]	(list of integers, blank = don't do anything.)	This option lets you specify which columns of tt_content elements should be displayed in the 'Columns' view of the modules, in particular Web>Page. By default there are four columns, Left, Normal, Right, Border. However most websites use only the Normal column, maybe another also. In that case the remaining columns are not needed. By this option you can specify exactly which of the columns you want to display. Each column has a number which ultimately comes from the configuration of the table tt_content, field 'colPos' found in the tables.php file. This is the values of the four default columns: Left: 1 Normal: 0 Right: 2 Border: 3 **Example:** This results in only the Normal and Border column being displayed: `mod.SHARED.colPos_list = 0,3`	

Web>Template (mod.web_ts):

Property:	Data type:	Description:	Default:

constantEditor.example
 	
[page:mod.web_ts; beuser:mod.web_ts]
		Controlling appearance of the example image in the Constant editor.
		Example:
		This will display the examples in the top of the page (default is bottom):
		
		`mod.web_ts {`
		`  constantEditor.example = top`
		`}`	

Web>Modules (mod.web_modules):

Property:	Data type:	Description:	Default:

dmail
 	
[page:mod.web_modules; beuser:mod.web_modules]
		This is the available properties. They are primarily set as page TSconfig and therefore you should refer to the module itself for documentation:
		
		`mod.web_modules.dmail {`
		`  from_email`
		`  from_name`
		`  replyto_email`
		`  replyto_name`
		`  organisation`
		`  sendOptions (isset)`
		`  HTMLParams (isset)`
		`  plainParams (isset)`
		`  userTable`
		`  enablePlain`
		`  enableHTML`
		`  http_username`
		`  http_password`
		
		`  test_tt_address_uids`
		
		`  categories.[integer=bit, 1-31] = label`
		`}`	

(Module: Web>Modules / Direct Mail)

Web>Page (mod.web_layout):

Property:	Data type:	Description:	Default:
tt_content.colPos_list	(list of integers, blank = don't do anything.)	See mod.SHARED.colPos_list for details. If non-blank, this list will override the one set by mod.SHARED.colPos_list. **Example:** This results in only the Normal and Border column being displayed: `mod.web_layout.tt_content.colPos_list = 0,3`	
tt_content.fieldOrder	list of field names from tt_content table	This allows you to specify (and thereby overrule) the preferred order of the fieldnames of the editing forms of the tt_content table (Content Elements). Just specify the list of fields, separated by comma. Then these fields will be listed first and all remaining fields thereafter in their original order. **Example:** This results in the 'Text' field and thereafter 'Header' field being display as the very first fields instead of the 'Type' field. `mod.web_layout.tt_content.fieldOrder = bodytext, header` (In one line)	
editFieldsAtATime	int+	Specifies the number of subsequent content elements to load in the editform when clicking the editicon of a content element in the 'Columns' view of the module. Default values are '3' in the Classic backend and '1' in the Alternative backend.	
noCreateRecordsLink	boolean	If set, the link in the bottom of the page, Create Records, are hidden.	
QEisDefault	boolean	If set, then the QuickEditor is the default function in the top of the menu in Web>Page	
disableSearchBox	boolean	Disables the search box in the bottom of the page in the module	
disableBigButtons	boolean	Disables the big buttons which shows up in the	
disableAdvanced	boolean	Disables the clear cache advanced function in the bottom of the page in the module	
disableNewContentElementWizard [page:mod.web_layout, beuser:mod.web_layout]	boolean	Disables the fact that the new-content-element icons links to the content element wizard and not directly to a blank “NEW” form.	

Wizard

Web>List (mod.web_list):

Property:	Data type:	Description:	Default:
noCreateRecordsLink	boolean	If set, the link in the bottom of the page, Create Records, are hidden.	
alternateBgColors	boolean	If set, the background colors of elements will alternate	
disableSingleTableView	boolean	If set, then the links which shows details about tables will not be available (including sorting links on columns titles, because these links jumps to the table-only view).	
allowedNewTables	*list of tablenames*	If this list is set, then only tables listed here will have a link to “create new” in the page and subpages. **Note:** Technically records can be created (eg. by copying/moving), so this is pseudo security. The point is to reduce the number of options for new records visually.	
newWizards	boolean	If set, then the new-link over the control panel of the pages and tt_content listings in the List module will link to the wizards and not create a record in the top of the list.	
showClipControlPanelsDespiteOfCMlayers [page:mod.web_list, beuser:mod.web_list]	boolean	If set, then the control- and clipboard panels of the module is shown even if the context-popups (ClickMenu) are available. Normally the control- and clipboard panels are disabled (unless extended mode is set) in order to save bandwidth.	

Property:	Data type:	Description:	Default:
type	int+	Enter the value of the &type parameter passed to the webpage.	

[page:mod.web_view; beuser:mod.web_view]

Tools>Extension Manager (mod.tools_em):

Property:	Data type:	Description:	Default:
allowTVlisting	boolean	If set the Technical and Validation listings are available in the EM. Those will evaluate ALL available extensions and that can take many seconds (up to 30) depending on number of extensions.	

[beuser:mod.tools_em]

New content element wizard (mod.xMOD_db_new_content_el):

Property:	Data type:	Description:	Default:

[page:mod.xMOD_db_new_content_el; beuser:mod.xMOD_db_new_content_el]

Edit document 'module' (mod.xMOD_alt_doc):

Property:	Data type:	Description:	Default:
disableDocSelector	boolean	If set, the document selector is disabled	
disableCacheSelector	boolean	If set, the cache/save/close selector is disabled	

[page:mod.xMOD_alt_doc; beuser:mod.xMOD_alt_doc]

RTE

Please refer to the special RTE section for indepth information on this feature in Typo3.

TCEMAIN

Property:	Data type:	Description:
table.[*tablename*] default	->TCEMAIN_tables	Options for each table.
permissions.userid permissions.groupid	int+	Sets the default owner be_user/be_group uid number of new and copied records. Default of be_user-owner is the creating user himself Default of the group is the main group of the user (the group in the top of his group list)
permissions.user permissions.group permissions.everybody	list of string/int[0-31]	Default permissions set for owner-user, owner-group and everybody. Keylist: show,edit,delete,new,editcontent Alternatively you can specify an integer from 0 to 31 indicating which bits corresponding to the keylist should be set. (Bits in keylist: show=1,edit=2,delete=4,new=8,editcontent=16) Defaults: "user" => "show,edit,delete,new,editcontent", “group" => "show,edit,new,editcontent", "everybody" => ""

[page:TCEMAIN]

TCEMAIN_tables

Property:	Data type:	Description:
history.keepEntries	int+	Max entries in the sys_history for this table. Range 0-200. Zero turns the history of.
history.maxAgeDays	int+	The number of days elements are in the history at most. Takes precedence over keepEntries. Default is 7 days. Range 0-200. Zero turns the magAgeDays of.
disablePrependAtCopy	boolean	Disables the “prependAtCopy” feature (from TCA)
disableHideAtCopy	boolean	Disables the “hideAtCopy” feature (from TCA)

[page:TCEMAIN.default/TCEMAIN.table.(tablename)/->TCEMAIN_tables]

TCEFORM

Most - or all - of these options apply only to the TCEFORM forms in Typo3 (that is the forms in the alternative backend) and you should consider disabling the classic Doc-module if you use these options extensively!

Property:	Description:
[*tablename*].[*field*].[...] [*tablename*].[*field*].types.[*type*].[...]	These objects contain additional configuration of the TCEFORM interface. For the properties available, refer to the table below. This is a description of how you can customize in general and override for specific types. 'TCEFORM.[*tablename*].[*field*].[...]' configures the field in TCEFORM for all types. 'TCEFORM.[*tablename*].[*field*].types.[*type*].[...]' configures the field in TCEFORM in case the 'type'-value of the field matches *type*.

[page:TCEFORM]

Properties of TCEFORM[....]

Property:	Data type:	Description:
disabled	boolean	If set, the field is not rendered.
removeItems	list of values	*(applies to select-types)* This removes the items from the list which has a value found in this comma list of values.
addItems.[itemValue]	string (label) divided by | for system languages	*(applies to select-types)* This will add elements to the list. Notice that the added elements might be removed if the selector represents records. In that case only still existing records will be preserved.
disableNoMatchingValueElement	boolean	*(applies to select-types)* If set, the element CURRENT VALUE IS NOT AVAILABLE will not be added to the list.
noMatchingValue_label	string, divided by | for system languages	*(applies to select-types)* Allows for an alternative label the “noMatchingValue” element. Notice that an empty value will be shown! If you want to unset this option, you must erase the TypoScript node by eg. “.... noMatchingValue_label > ” The string is divided by | according to the system languages.
altLabels.[item_value]	string, divided by | for system languages	*(applies to select-types)* This allows you to enter alternative labels for the items in the list. See also “.noMatchingValue_label” for syntax. The string is divided by | according to the system languages.
PAGE_TSCONFIG_ID	integer	*(applies to select-types with foreign table)* When the select-types are used with foreign-table the where-query has four markers (see description of $TCA in the “Inside Typo3” document.) The value of three of these markers may be set from Page TSconfig.
PAGE_TSCONFIG_IDLIST	comma list of integers	*(applies to select-types with foreign table)* See above
PAGE_TSCONFIG_STR	string	*(applies to select-types with foreign table)* See above
itemsProcFunc.[...]	(custom)	*(applies to select-types with itemsProcFunc)* The properties of this key is passed on to the itemsProcFunc in the parameter array by the key “TSconfig”.
RTEfullScreenWidth	int+/%	*(applies for RTE text fields only with the RTE wizard configured)* The width of the RTE full screen display. If nothing is set, the whole width is used which means “100%”. If you set an integer value, that indicates the pixels width.
linkTitleToSelf	boolean	*(all fields)* If set, then the title of the field in the forms links to alt_doc.php editing ONLY that field. Works for existing records only - not new records. Extra property: .returnUrl = boolean; if set, the nthe return URL is also set.

[page:TCEFORM.(tablename).(field)/TCEFORM.(tablename).(field).types.(type)]

Rich Text Editor (RTE)

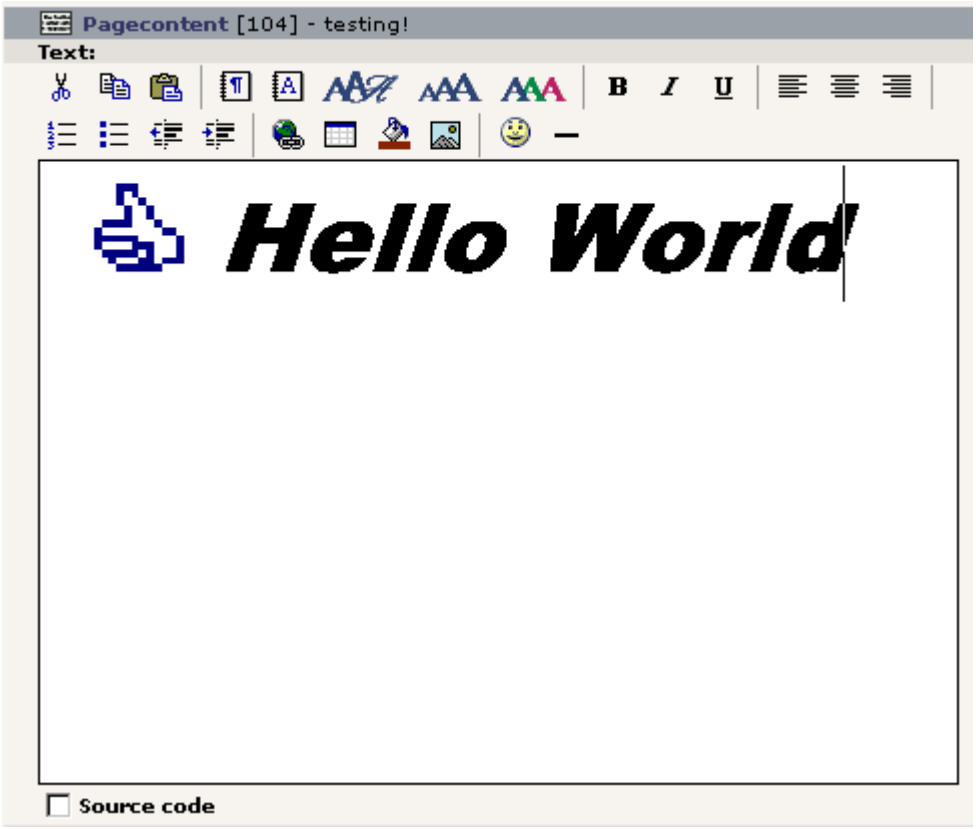
Introduction

Typo3 has got a so called Rich Text Editor built in. This is basically a tool which allows you to edit the content of a database field with HTML formatting options like bold and italics, headlines, links, images etc. The editor is programmed in JavaScript and requires MSIE 5+ in order to work because it depends on special MSIE function calls. With browsers other that MSIE 5+ the field will display as a regular text field and may so still be edited - just without the aid of the RTE interface.

You can configure the Rich Text Editor for any 'text' database field configured in tables.php ($TCA). The most prominent use of the RTE is probably in the 'Text' (bodytext) field of the table tt_content (content elements). This is also the field discussed in details here.

Furthermore the Rich Text Editor may be enabled or disabled on user-basis (there's an option in the User>Setup module), then there's a global switch in the TYPO3_CONF_VARS array set from typo3conf/locallang.php: $TYPO3_CONF_VARS[“BE”][“RTEenabled”], and finally you can disable it through Page TSconfig (see later) and even configure another boolean field in the database table to determine whether it should appear or not...

So the basic requirements for it's use is: 1) It must be configured in the $TCA array for the database field in question and if any 'flag'-field is configured as disable flag, it must not be set, 2) it must be enabled by $TYPO3_CONF_VARS[“BE”][“RTEenabled”] (it is enabled by default), 3) the Page TSconfig must not be set to disable the RTE in the page tree branch of the record, 4) the user must be logged in with a MSIE 5+ browser and 5) the user must have the Rich Text Editor option in the User>Setup enabled (it is enabled by default). There are even more options through Page TSconfig. If you are trying to debug why the RTE does not appear at a field location where you would expect it, try to look in the page HTML-source - in the bottom there should be some HTML-commens with the reason described.



(This is the RTE with the full number of options enabled. This is available from the 'Rich Text' type of content elements, tt_content.)

Configuration of the toolbar

For each field you can configure which options you want in the tool bar. This configuration is done per 'type' of the record (for instance the toolbar is different whether you use the RTE with the content element types 'Text', 'Text

w/Image' or 'Rich Text') and furthermore toolbar buttons may be disabled on userlevel and finally also with Page TSconfig.

The configuration of the RTE is quite comprehensive and therefore discussed separately later in this section.

Properties and 'transformations'

As stated the RTE depends on MSIE specific function calls. This means that the treatment of the field content is dependant on how MSIE handles this. In effect the editor will discard content it doesn't like, for instance fictitious HTML tags and linebreaks.

Also the HTML content created by the editor is not necessarily as 'clean' as you might like. However this is the conditions on which you may use this tool.

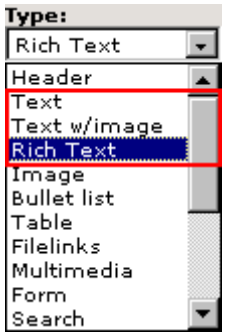
The editor has the ability to paste in formatted content copied/cut from other websites (in which case images are included!) or from text processing applications like MS Word or Star Office. This is a great feature and may solve the issue of transferring formatted content from eg. Word into Typo3.

However these inherent features - good or bad - raises the issue how to handle content in a field which we do not wish to 'pollute' with unnecessary HTML-junk. One perspective is the fact that we might like to edit the content with Netscape later (for which the RTE cannot be used, see above) and therefore would like it to be 'human readable'. Another perspective is if we might like to use only Bold and Italics but not the alignment options. Although you can configure the editor to display *only* the bold and italics buttons, this does *not* prevent users from pasting in HTML-content copied from other websites or from Microsoft Word which *does* contain tables, images, headlines etc.

The answer to this problem is a so called 'transformation' which you can configure in the $TCA (global, authoritative configuration) and which you may further customize through Page TSconfig (localized configuration for specific branches of the website). The issue of transformations is best explained by the following example from the table, tt_content (the content elements).

RTE transformations in Content Elements

The RTE is used in the bodytext field of the content elements, configured for the types 'Text', 'Text w/Image' and 'Rich Text'.



This is how the toolbar looks if the type of the content element is not 'Rich Text' but 'Text'.

The configuration of the two 'Text'-types are the same: The toolbar includes only a subset of the total available buttons. The reason is that the text content of these types, 'Text' and 'Text w/Image' is *traditionally* not meant to be filled up with HTML-codes. But more important is the fact that the content is usually (by the standard TypoScript content rendering used on the vast majority of Typo3 websites!) parsed through a number of routines. In order to understand this, here is an outline of what typically happens with the content of the two Text-types when displayed:

1. All linebreaks are converted to
 codes.

   (Doing this enables us to edit the text in the field rather naturally in the backend because linebreaks in the edit field comes out as linebreaks on the page!)

2. All instances of 'http://...' and 'mailto:....' are converted to links.

   (This is a quick way to insert links to URLs and email address)

3. The text is parsed for special tags, so called 'typotags', configured in TypoScript. The default typotags tags are <LINK> (making links), <TYPOLIST> (making bulletlists), <TYPOHEAD> (making headlines) and <TYPOCODE> (making monospaced formatting).

   (The <LINK> tag is used to create links between pages inside Typo3. Target and additional parameters are automatically added which makes it a very easy way to make sure, links are correct. <TYPOLIST> renders each line between the start and end tag as a line in a bulletlist, formatted like the content element type 'Bulletlist' would be. This would typically result in a bulletlist placed in a table and not using the bullet-list tags from HTML. <TYPOHEAD> would display the tag content as a headline. The type-parameter allows to select between the five default layout types of content element headlines. This might include graphical headers. <TYPOCODE> is not converted).

4. All other 'tags' found in the content are converted to regular text (with htmlspecialchars) unless the tag is

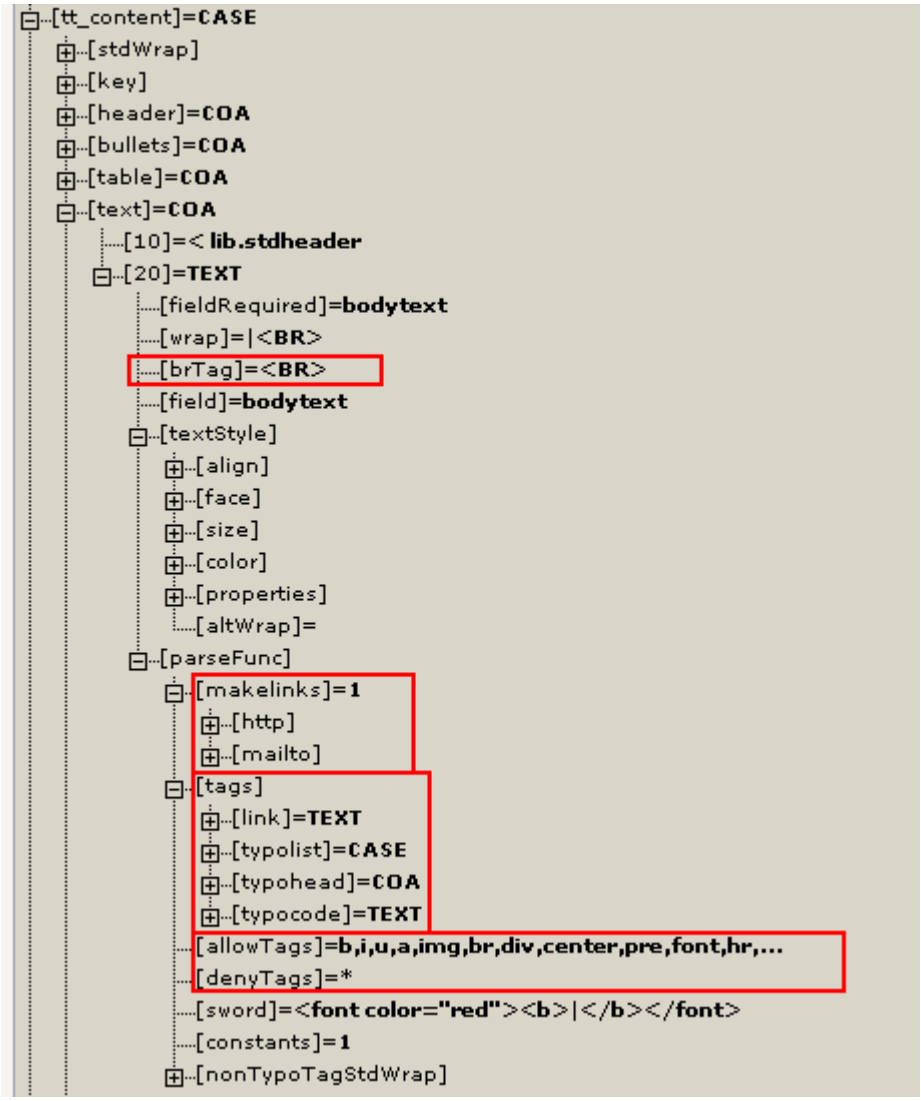
found in the 'allowTags' list.

(This list includes tags like 'b' (bold) and 'i' (italics) and so these tags may be used and will be outputted. However tags like 'table', 'tr' and 'td' is not in this list by default, so table-html code inserted will be outputted as text and not as a table!)

5. Constants and search-words - if set - will be highlighted or inserted.

   (This feature will mark up any found search words on the pages if the page is linked to from a search result page.)

6. And finally the result of this processing may be wrapped in <font>-tags, <p>-tags or whatever is configured. This depends on whether a stylesheet is used or not. If a stylesheet is used the individual sections between the typotags are usually wrapped separately.



This is how the TypoScript configuration of the display of bodytext looks (from the Web>TS module, Object Browser)

Now lets see how this behaviour challenges the use of the RTE. This describes how the situation is handled regarding the two Text-types as mentioned above. (Numbers refer to the previous bulletlist):

1. Linebreaks: The RTE removes all linebreaks and makes linebreaks itself by either inserting a <P>...</P> section or <DIV>...</DIV>. This means we'll have to convert existing lines to <P>...</P> before passing the content to the RTE and further we need to revert the <DIV> and <P> sections in addition to the
-tags to linebreaks when the content is returned to the database from the RTE.

   The greatest challenge here is however what to do if a <DIV> or <P> tag has parameters like 'class' or 'align'. In that case we can't just discard the tag. So the tag is preserved.

2. The substitution of http:// and mailto: does not represent any problems here.

3. Typotags: The typotags are not real HTML tags so they would be removed by the RTE. Therefore those tags must be converted into something else. This is actually an opportunity and the solution to the problem is that all <LINK>-tags are converted into regular <A>-tags, all <TYPOLIST> tags are converted into <OL> or <UL> sections (ordered/unordered lists, type depends on the type set for the <TYPOLIST> tag!), <TYPOHEAD>-tags are converted to <Hx> tags where the number is determined by the type-parameter set for the <TYPOHEAD>-tag. The align/class-parameter - if set - is also preserved. When the HTML-tags are returned to the database they need to be reverted to the specific typotags.

   Other typotags (non-standard) can be preserved by being converted to a <SPAN>-section and back. This must be configured through Page TSconfig.

4. Allowed tags: As not all tags are allowed in the display on the webpage, the RTE should also reflect this situation. The greatest problem is tables which are (currently) not allowed with the Text-types. The reason for this goes back to the philosophy that the field content should be human readable and tables are not very 'readable'.
5. Constants and search words are no problem.
6. Global wrapping does not represent a problem either. But this issue is related more closely to the linebreak-issue in bullet 1.

Finally images inserted are processed very intelligently because the 'magic' type images are automatically post-processed to the correct size and proportions after being changed by the RTE in size.

Also if images are inserted by a copy/paste operation from another website, the image inserted will be automatically transferred to the server when saved.

In addition all URLs for images and links are inserted as absolute URLs and must be converted to relative urls if they are within the current domain.

Conclusion:

These actions are done by so called *transformations* which are configured in the $TCA. Basically these transformations are admittedly very customized to the default behaviour of the Typo3 frontend. And they are by nature “fragile” constructions because the content is transformed forth and back for each interaction between the RTE and the database and may so be erroneously processed. However they serve to keep the content stored in the database 'clean' and human readable so it may continuosly be edited by non-RTE browsers and users. And furthermore it allows us to insert Typo3-bulletlists and headers (especially graphical headers) visually by the editor while still having Typo3 controlling the output.

Now, (most of) these transformations are processed only for the two mentioned Text-types. The alternative 'Rich Text' type is meant as an alternative to which you may seek if these transformations are not sufficient or if you need the full range of toolbar options. The main deal with the 'Rich Text' type is the fact that it is *not* processed in any way when outputted on the page and thus is similar to the 'HTML' type - it's just been added the RTE interface! Only images and links are processed when using the 'Rich Text' type.

Switching between the 'Rich Text' and 'Text'-types may result in loss of information due to their differences in transformation options.

Thoughts behind current strategy of handling content between RTE, database and the website (font/stylesheet rendered) (From email by Kasper Skårhøj to the Typo3-features mailing list 3/5 2002)

BACKGROUND:
OK, there are two scenarios:
Either you are formatting the content by stylesheets in which case you would probably wrap bodytext in <P> or <DIV>-tags. The Rich Text editor will support both options and anyways, my new formatting code is able to convert between the two.
When content is stored in the database, it is stripped from ALL <P> and <DIV> sections which *does not* contain an align or class attribute.

This comes from the RTE:

```
<P>First line</P>
<P align="center">Second line</P>
<DIV class="somestuff">Third line</DIV>
<P>Yet another looooonger line!</P>
```

This is converted to this (below) which is finally stored in the db (done by the ts_transform transformation):

```
First line
<DIV align="center">Second line</DIV>
<DIV class="somestuff">Third line</DIV>
Yet another looooonger line!
```

The reason why the convertion takes place is, that we want the content in the database to be as human readable as possible! There ARE still people NOT using the RTE and from time to time people might disable it temporarily etc. Therefore we are striving for a situation where only the most necessary HTML codes are preserved and the others not.

PROBLEM:
THe problem arises when we are going to render the database content. Because traditionally there was NO problem just wrapping the whole content into <P>-tags or <FONT>-tags and converting all NL's to
. However this is not true anymore with the new

scenario. Consider this:

Database content:

First line
<DIV align="center">Second line</DIV>
<DIV class="somestuff">Third line</DIV>
Yet another looooonger line!

... is converted on display to this (brTag =
, wrap = <FONT face=verdana>|</FONT>)

<FONT face=verdana>
First line
<DIV align="center">Second line</DIV>
<DIV class="somestuff">Third line</DIV>
Yet another looooonger line!
</FONT>

Problems are:
- The
 tags should NOT be put after lines with <DIV> sections.
- The <FONT>-tag wrap will not work for the <DIV>-sections and possibly the <DIV>-sections will remove the effect of the font-wrap of the content after them...

The situation is approximately the same if you do not render with <FONT>-tags, but instead with a stylesheet.

THE SOLUTION:
So what I have done is to add a new stdWrap options which offers an alternative way of rendering lines of content. It explodes the content as usual, but then traverses through each line of the content;
- If the line is wrapped in <P> or <DIV>, then this is observed. For instance the content INSIDE of <P>/<DIV> can be wrapped with the font-tag which used to wrap the WHOLE section. If you are using a style-sheet, you could choose to do nothing, because the <P>/<DIV> might already have a style assigned. This is configurable of course.
- If the line is NOT wrapped in <P> or <DIV> then this is also observed; You might choose to actually wrap the lines in <P>-tags!

This is the new defaults of content rendering (<FONT>-tags) (by Typo3.3b1.9):

Database content:

First line
<DIV align="center">Second line</DIV>
<DIV class="somestuff">Third line</DIV>
Yet another looooonger line!

... is converted on display to this:

<P style="margin:0 0 0;"><FONT face=verdana>First line</FONT></P>
<P style="margin:0 0 0;" align="center"><FONT face=verdana>Second line</FONT></P>
<P style="margin:0 0 0;" class="somestuff"><FONT face=verdana>Third line</FONT></P>
<P style="margin:0 0 0;"><FONT face=verdana>Yet another looooonger line!</FONT></P>

Comments:
- The STYLE-attribute with the value "margin:0 0 0;" ensures that there is no extra space after <P>-sections. In my tests, this will render the line-distances EXACTLY like in the good old times with
-tags! (Goes apparently for browsers including NN4!)
- By wrapping ALL lines with <P>-tags, even though the content is wrapped in font-tags afterwards allows us to align paragraphs. I see no other way to align paragraphs than using <DIV>/<P> tags. Is there any other way?
- Obviously the class-attribute is a mix of concepts here; If one has chosen to render his content with <FONT>-tags it seems strange to allow the class-attribute because the only reason we have for keeping the <P>-tags is alignment.

This is the new defaults when using stylesheets with a definition of <P>-tags:

Database content:

First line
<DIV align="center">Second line</DIV>
<DIV class="somestuff">Third line</DIV>
Yet another looooonger line!

... is converted on display to this:

<P class="default_class">First line</P>
<P align="center">Second line</P>
<P class="somestuff">Third line</P>
<P class="default_class">Yet another looooonger line!</P>

Comments:
- The class "default_class" is what you might choose as a default class for the <P>-tags unless another tag is set.

Of course the options for wrapping are configurable. Please see TSref ->encapsLines for the options and examples.

PERSISTING PROBLEMS (soft linebreaks):

- Rene Fritz has used a concept in Typo3 where he configured through TypoScript that double-line-breaks to be wrapped in <P>-tags while single linebreaks should be wrapped in
. This concept CANNOT be used with RTE anyways and my new concept here does not leave any room for it either. Actually soft linebreaks in general are not supported by the RTE - they are forcibly being converted to real line breaks and thereby <P>/<DIV> sections. You can configure the RTE so that soft linebreaks are preserved, but the price to pay will be that those linebreaks goes into the database content as
-tags (and not NL's). And that will not be handy when pasting content from outside, because any linebreaks there will become
's in the editor.

Technical description of transformations

The transformation of the content can be configured by listing which filters to pass it through. The order of the list is the order in which the transformations are performed when saved to the database. The order is reversed when the content is loaded into the RTE again.

Modes are:

ts_transform: Transforms the content with regard to most of the issues mentioned above related to content elements types 'Text' and 'Text w/Image'.

css_transform: Like “ts_transform”, but headers and bulletlists are preserved (TYPOLIST and TYPOHEAD are still converted to Hx and OL/UL, but not reversely...) and tables are preserved (PROC.preserveTables is not active).

ts_preserve: Converts the list of preserved tags - if any - to <SPAN>-tags with a custom parameter 'specialtag' which holds the value of the original tag. Depreciated.

ts_images: Checks if any images on the page is from external URLs and if so they are fetched and stored in the uploads/ folder. In addition 'magic' images are evaluated to see if their size has changed and if so the image is recalculated on the server. Finally absolute urls are converted to relative urls for all local images.

ts_links: Converts the absolute URLs of links to the TypoScript specific <LINK>-tag. This proces is designed to make links in concordance with the typolink function in the TypoScript frontend.

ts_reglinks: Converts the absolute URLs of links to relative. Keeping the <A>-tag.

ts: Pseudo-mode which is basically a substitute for this list: ts_transform,ts_preserve,ts_images,ts_links. This is the one used specifically for the two 'Text'-types of the content elements.

ts_css: Like “ts”, as pseudo-mode which is a substitute for the list: css_transform,ts_images,ts_links. It is designed to be the new, modern transformation used by most RTE cases, because it converts links between <A> and <LINK> but preserves all other content while still making it as human readable as possible (that means simple <P>-tags are resolved into simple lines.)

Here follows a technical and detailed description of the transformation filters:

ts_transform, css_transform

Direction: To RTE

PHP-Function: function t3lib_parseHTML::TS_transform_rte()

Actions:

- Sections by the tags TABLE,PRE,UL,OL,H1,H2,H3,H4,H5,H6 are not processed and thus just passed on to the RTE.
- The content of <BLOCKQUOTE> sections are sent recursively through the ts_transform filter. The tag remains.
- <TYPOLIST> sections are converted to <OL> or <UL> sections, the latter is the case if the type parameter is set to 1.

  The conversion of TYPOLIST-tags can be disabled by setting the 'proc.typolist' option. See later.
- <TYPOHEAD> sections are converted to <Hx>-tags. The type parameter ranging from 1-5 determines which H-tag will be used. If no type parameter is set, H6 is used.

  The conversion of TYPOHEAD-tags can be disabled by setting the 'proc.typohead' option. See later.
- All content outside the tags already mentioned are now processed as follows: Every line is wrapped in <P>-tags (configurable to DIV), if a line is empty a   is set and if the line happens to be wrapped in DIV/P-tags already, it's not wrapped again (this might be the case if align or class parameters has been set). Then <B> tags are mapped to <STRONG> tags and <I> tags are mapped to <EM> tags (This is how the RTE prefers it).
- All content between the P/DIV tags outside of other allowed HTML-tags are htmlspecialchar()'ed. Thus only allowed HTML code is preserved and other “pseudo tags” are mapped to real text.

Direction: To Database

PHP-Function: function t3lib_parseHTML::TS_transform_db()

Actions:

- Sections by the tag PRE is not processed and thus just passed on to the RTE.

- <TABLE>-sections are dissolved so only the text of the table cells remains. Every cell represents a new line. The reason for this action basically is that tables are not wanted in the 'Text'-types and they may also be nice to get rid of in case you have transferred content from other websites. (This can be disabled.)

  (Does NOT apply to “css_transform”)
- The content of <BLOCKQUOTE> sections are sent recursively through the ts_transform filter. The tag remains.
- <OL> and <UL> sections are converted to <TYPOLIST> sections. If the bulletlist is <OL> (ordered list with numbers) the type parameter of the typolist is set to 1. Bulletlists in multiple levels are not supported.

  The conversion of TYPOLIST-tags can be disabled by setting the 'proc.typolist' option. See later.

  (Does NOT apply to “css_transform”)
- <Hx> sections are converted to <TYPOHEAD>-tags. The number of the Hx-tag ranging from 1-5 is set as the type-number of the TYPOHEAD tag. <H6> is equal to type=0 (default). Also the align parameter is preserved as well as the class parameter if set.

  The conversion of TYPOHEAD-tags can be disabled by setting the 'proc.typohead' option. In that case the tag is preserved with the parameters align and class. See later.

  (Does NOT apply to “css_transform”)
- All content outside these block are now processed as follows:
- All <DIV> and <P> sections are dissolved into lines (unless align and/or class parameters are set).
 tags are as well converted into newlines (configurable). Then <STRONG> and <EM> tags are remapped to <B> and <I> tags. (This is more human readable. Configurable). The list of allowed tags (configurable) is preserved - all other tags discarded (thus junk-tags from pasted content will not survive into the database!).
- The content outside the allowed tags are de-htmlspecialchar()'ed - thus converted back to human-readable text. Furthermore the nesting of tags inside of P/DIV sections is preserved. For instance this: <P>One <U><B>two</B> three</P></U> will be converted to <P>One <B>two</B> three</P>. That is the U-tags being removed, because they were falsely nested with the <P> tags.

ts_preserve (depreciated)

Direction: To RTE

PHP-Function: function t3lib_parseHTML::TS_preserve_rte()

Actions:

- If 'proc.preserveTags' are configured those tags are converted to <SPAN specialtag=”...(the preserved tag rawurlencoded)...”>-sections. Those are supposed to be let alone by the RTE.

Direction: To database

PHP-Function: function t3lib_parseHTML::TS_preserve_db()

Actions:

- If 'proc.preserveTags' are configured <SPAN>-tags with the custom 'specialtag' parameter set are converted back to the tag value contained in the specialtag-parameter.

ts_images

Direction: To RTE

PHP-Function: function t3lib_parseHTML::TS_images_rte()

Actions:

- All <IMG>-tags are processed and if the value of the src-parameter happens *not* to start with 'http' it's expected to be a relative url and the current site URL is prefixed so the reference is absolute in the RTE as the RTE requires.

Direction: To database

PHP-Function: function t3lib_parseHTML::TS_images_db()

Actions:

- All <IMG>-tags are processed and if the first part of the src-parameter is not the same as the current site URL, the image must be a reference to an external image. In that case the image is read from that URL and stored as a 'magic' image in the upload/ folder (can be disabled).

- All magic images (that is images stored in the uploads/ folder (configured by TYPO3_CONF_VARS["BE"]["RTE_imageStorageDir"], filenames prefixed with 'RTEmagicC_' (child=actual image) and 'RTEmagicP_' (parent=original image))) are processed to see if the physical dimensions of the image on the server matches the dimensions set in the img-tag. If this is not the case, the user must have changed the dimensions and the image must be re-scaled accordingly.
- Finally the absolute reference to the image is converted to a proper relative reference if the image url is local.

ts_links

Direction: To RTE

PHP-Function: function t3lib_parseHTML::TS_links_rte()

Actions:

- All <LINK>-tags (TypoScript specific) are converted to proper <A>-tags. The parameters of the <LINK>-tags are separated by space. The first parameter is the link reference (see typolink function in TSref for details on the syntax), second is the target if given (if '-' the target is not set) and the third parameter is the class.

Direction: To database

PHP-Function: function t3lib_parseHTML::TS_links_db()

Actions:

- All <A>-tags are converted to <LINK> tags, however only if they do not contain any parameters other than href, target and class. These are the only three parameters which can be represented by the TypoScript specific <LINK>-tag.

ts_reglinks

Direction: To RTE

PHP-Function: function t3lib_parseHTML::TS_reglinks()

Actions:

- All A-tags have urls converted to absolute urls if they are relative

Direction: To database

PHP-Function: function t3lib_parseHTML::TS_reglinks()

Actions:

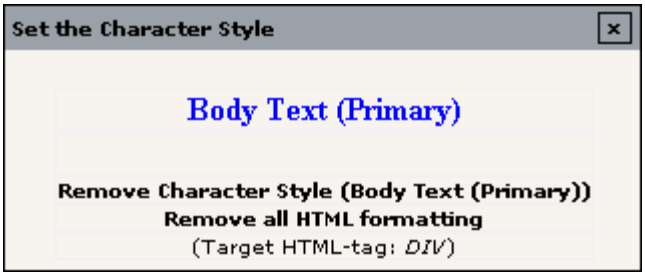
- All A-tags have their absolute urls converted to relative if possible (that is the url is within the current domain).

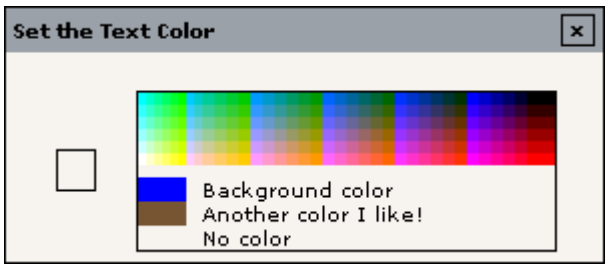
Interface configuration

Interface configuration through Page TSconfig

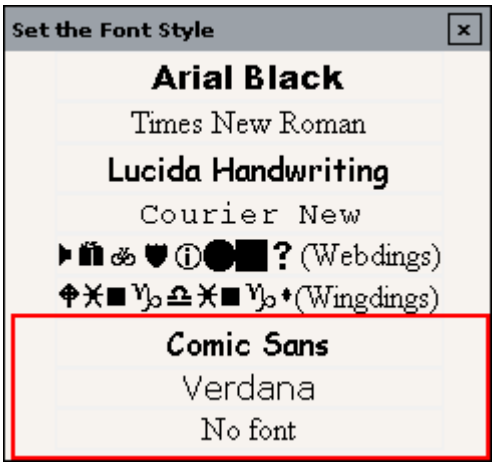
This is the 'Page TSconfig' options of the RTE:

Property:	Description:

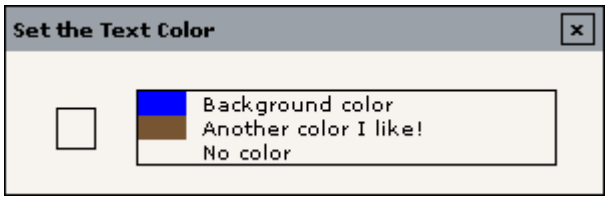
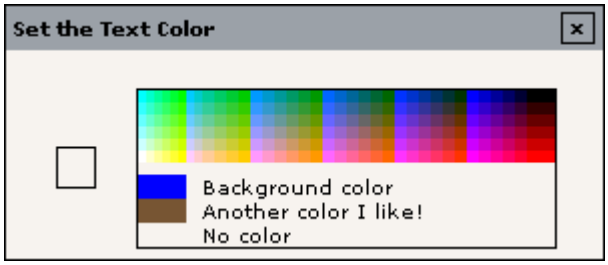
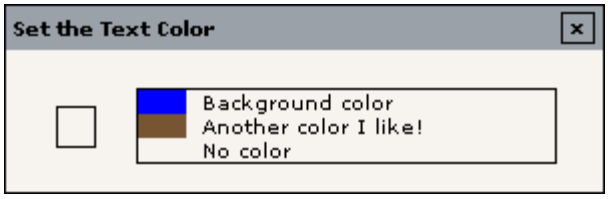
classes.[*classname*]	Defines the classes available in the RTE. *classname* is the actual name of the style-class you're configuring. Notice you must specifically assign the classes to the various facilities also. See later.
	
	Properties:
	
	.name = Label of the class (language-splitted)
	.value = The style for the class
	.noShow = Boolean; if set, the style of the class is not used to render it in the pop-up selector.
	
	Example:
	
	`# General configuration of the available classes:`
	`RTE.classes {`
	`  mainBodyText {`
	`  name = Body Text (Primary)`
	`  value = font:bold; color:blue;`
	`  }`
	`}`
	`# Specific configuration for the Character Style menu:`
	`RTE.default.classesCharacter = mainBodyText`
	
	The class might be configured to show up like this:
	
	 

colors.[*id-string*]	Defines the colors available in the RTE.
	
	Properties:
	
	.name = Label of the color in menu (language-splitted)
	.value = The HTML-color value
	
	Example:
	
	`# General configuration of the available colors:`
	`RTE.colors {`
	`  color1 {`
	`  name = Background color`
	`  value = blue`
	`  }`
	`  color2 {`
	`  name = Another color I like!`
	`  value = #775533`
	`  }`
	`  noColor {`
	`  name = No color`
	`  value =`
	`  }`
	`}`
	`# Specific setting for the font color selector:`
	`RTE.default.colors = color1, color2, noColor`
	
	This results in a color picker like this:
	
	 

Property:	Description:

fonts.[*id-string*]	Defines the fonts available in the RTE.
	
	Properties:
	
	.name = Label of the font in menu (language-splitted)
	.value = The font face value
	
	Example:
	
	`# General configuration of the available fonts:`
	`RTE.fonts {`
	`  face1 {`
	`  name = Verdana`
	`  value = verdana, arial`
	`  }`
	`  face2 {`
	`  name = Comic Sans`
	`  value = Comic Sans MS`
	`  }`
	`  noFace {`
	`  name = No font`
	`  value =`
	`  }`
	`}`
	`# Specific setting for the fontstyle selector:`
	`RTE.default.fontFace = face2 , face1, noFace`
	
	This results in a font menu like this:
	
	 

Property:	Description:

default.[...]
 	
config.[*tablename*].[*field*].[...]
 	
config.[*tablename*].[*field*].types.[*type*].[...]
	These objects contain the actual configuration of the RTE interface. For the properties available, refer to the table below. This is a description of how you can customize in general and override for specific fields/types.
	
	'RTE.default' configures the RTE for all tables/fields/types
	
	'RTE.config.[*tablename*].[*field*]' configures a specific field. The values inherit the values from 'RTE.default' in fact this is overriding values.
	
	'RTE.config.[*tablename*].[*field*].types.[*type*]' configures a specific field in case the 'type'-value of the field matches *type*. Again this overrides the former settings.
	
	Example:
	
	Lets imaging we would like to disable the color picker matrix for the RTE in general. Then we would set this option:
	
	`RTE.default.disableColorPicker = 1`
	
	and this would be the result:
	
	 
	
	However we might like the color picker to appear for all use of the RTE with the bodytext field ('Text'-field) of content elements. Then we would set this value:
	
	`RTE.config.tt_content.bodytext.disableColorPicker = 0`
	
	and this would be the result:
	
	 
	
	Finally we might like to remove the color picker again in case the user has selected the type 'Text' or 'Text w/Image' of the content element:
	
	`RTE.config.tt_content.bodytext.types {`
	`  text.disableColorPicker = 1`
	`  textpic.disableColorPicker = 1`
	`}`
	
	and we're back to this:
	
	 
	
	(Only little thing here is that the color picker is not used anywhere with the 'text' and 'textpic' types, but in principle this is how it works... :-))

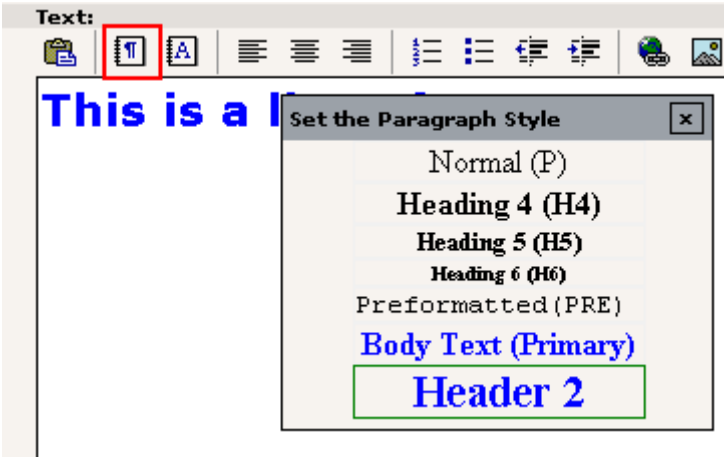
[page:RTE]

Configuration of the RTE editor interface

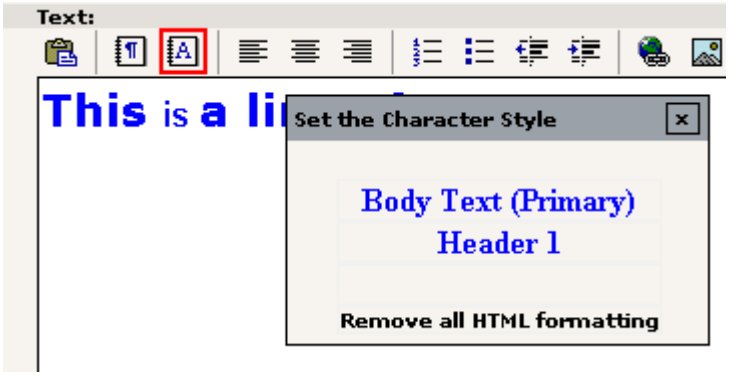
These options are designed to let you customize the RTE interface. The RTE interface does not automatically reflect the styles defined in your frontend template. However you might like to get as close as possible to the actual layout

of your website in the editor. Therefore you must configure the RTE to reflect the design of the frontend template.

Property:	Data type:	Description:
disabled	boolean	If set, the editor is disabled.
disableColorPicker	boolean	Disables the color picker matrix in all color dialogs. The color picker lets you select web-colors. See examples above.

classesParagraph	*list of id-strings*	Classes available in the Paragraph Style selector.
		
		The Paragraph Style selector lets you format a block of content also known typographically as a *paragraph* (the section between two linebreaks). The content of the paragraph is wrapped in block-tags and by default you can select between <Hx>, <P> and <PRE> formatting (you can disable any or all of these default values by the 'hidePStyleItems', see later).
		
		The classes defined in the 'classesParagraph' value will be prepended to these options as you see in this example. They are inserted as <DIV>-tags with a class-parameter.
		
		Example:
		
		This configuration would result in this Paragraph style selector (provided that the classes 'mainBodyText' and 'header2' are found in the RTE.classes definition, see above):
		
		`RTE.default {`
		`  classesParagraph = mainBodyText, header2`
		`}`
		
		 

Property:	Data type:	Description:

classesCharacter	*list of id-strings*	Classes available (by default) in the Character Style selector.
		
		The Character Style selector is used to format any string of text inside a paragraph, tablecell. So this formatting relates to the selected piece of text in opposition to the Paragraph Styles which always formats the current paragraph no matter the selection.
		
		Although the Character Style selector is aimed at text-formatting, it will also apply classes to other elements, for instance tables or images if selected as well as links and table-cells. The allowed classes for these operations may be configured separately by listing the classes with the options, you find below.
		
		Example:
		
		This configuration will display these options in the Character Style dialog:
		
		`RTE.default {`
		`  classesCharacter = mainBodyText, header1`
		`}`
		
		 

classesImage	*list of id-strings*	Classes available for images. See description of 'classesCharacter'.
classesTable	*list of id-strings*	Classes available for tables. See description of 'classesCharacter'.
classesLinks	*list of id-strings*	Classes available for links. See description of 'classesCharacter'.
classesTD	*list of id-strings*	Classes available for tablecells. See description of 'classesCharacter'.

colors	*list of id-strings*	Defines the specific colors generally available in the color selectors. The id-strings must be configured in the RTE.colors array (see description earlier):
		
		Example:
		
		`RTE.default {`
		`  colors = color1, color2,noColor`
		`}`

fontFace	*list of id-strings*	Define
hidePStyleItems	*list of tag names,* ** removes all*	Lets you disable any of the default Paragraph Style options. Possible values are: H1, H2, H3, H4, H5, H6, P, PRE
hideFontFaces	*list of id-numbers,* ** removes all*	Lets you disable any of the default font faces in the Font Style selector. These are the possible values you can set: 1: Arial 2: Arial Black 3: Verdana 4: Times New Roman 5: Garamond 6: Lucida Handwriting 7: Courier 8: Webdings 9: Wingdings
hideFontSizes	*list of size-numbers,* ** removes all*	Lets you disable any of the default font sizes available in the Font Size selector. Values are ranging from 1-7.

Property:	Data type:	Description:

showButtons
hideButtons
	list of id-strings	*showButtons:* Any keys entered here is added to the keylist configured in TCA. Thus you can add buttons to the default set. However any keys entered here are subject to be removed by the .hideButtons or any per-user defined limitations of buttons in the RTE.
		hideButtons: The options in the toolbar are configured through the settings in the $TCA. However certain options may be blinded on user level. This option allows you to further blind toolbar options on a general level.
		This is the keylist: cut,copy,paste,formatblock,class,fontstyle,fontsize, textcolor,bold,italic,underline,left,center,right,orderedlist,unorderedlist,outdent, indent,link,table,bgcolor,image,emoticon,line,user,chMode
		The keylist for the buttons is also displayed later in this documentation.
		Example:
		This will remove the buttons for Bold, Italics and Underline from the toolbar:
		`RTE.default {`
		`  hideButtons = bold, underline, italic`
		`}`

proc	->PROC	Customization of the server processing of the content - also called 'transformations'. See table below. The transformations are only initialized, if they are configured ("rte_transform" must be set for the field in the types-definition in TCA.)

mainStyleOverride	string	By default the editor style section is set with the CSS-code below. However you may override this by this option.
		Default:
		`BODY {`
		`  border: 1px black solid;`
		`  border-top: none;`
		`  margin : 2 2 2 2;`
		`  font-family:Verdana;`
		`  font-size:10px;`
		`  color:black;`
		`  background-color:white;`
		`}`
		`TD {font-family:Verdana; font-size:10px;}`
		`P {margin-top:0px; margin-bottom:5px;}`
		`DIV {margin-top:0px; margin-bottom:5px;}`
		`OL {margin: 5px 10px 5px;}`
		`UL {margin: 5px 10px 5px;}`
		`BLOCKQUOTE {margin-top:0px; margin-bottom:0px;}`

mainStyleOverride_add.[*key*]	string (css-style)	Allows to add style configuration to the values above (for .mainStyleOverride above) Keys are: .P / .DIV / .TD /.BODY / .BLOCKQUOTE /.OL / .UL / .PRE / .Hx will all take values which are prepended to the above settings. **Example:**
mainStyle_font mainStyle_size mainStyle_color mainStyle_bgcolor	string	Setting the default font-family (verdana) , font-size (10px), font-color (black) and background color (white) The default is shown in ".mainStyleOverride" above.
inlineStyle.[any *keysting*]	string	CSS code to be included in the editor style section. This will be included *after* the default code.
defaultLinkTarget	string	This sets the default target for new links in the RTE
blindLinkOptions	list of strings	List of menu items in the link selector to remove. Keylist is page,file,url,mail,spec
blindImageOptions	list of strings	List of menu items in the image selector to remove. Keylist is magic,plain,dragdrop
userElements.[#]	string/->userCategory	Configuration of the categories of user elements The string value sets the name of the category. Value is language-splitted (by |) to allow for multiple languages.
userLinks.[#]	string/->userLinks	Configuration of user defined links. The string value sets the name of the category. Value is language-splitted (by |) to allow for multiple languages.
disableRightClick	boolean	If set, the right click (context) menu of the RTE is disabled.

Property:	Data type:	Description:
disablePCexamples	boolean	If set, the examples of classes in Paragraph and Character selectors are disabled.

[page:RTE.default/RTE.config.(table).(field)/RTE.config.(table).(field).types.(type)]

PROC:

These options does not relate to the RTE interface as the above options. Rather they relate to the processing (transformations) of the content to and from the database. For information about the concept of transformations see the extensive description above.

The main objective of these options are to allow for minor configuration of the transformations. For instance you may disable the mapping between <B>-<STRONG> and <I>-<EM> tags which is done by the 'ts_transform' transformation. Or you could disable the default transfer of images from external URL to the local server. This is all possible through the options.

Property:	Data type:	Description:
overruleMode	List of RTE transformations	This can overrule the RTE transformation set from TCA.
typolist	boolean	*(Applies for "ts_transform" only)* This disables the conversion between <TYPOLIST> and <UL> sections.
typohead	boolean	*(Applies for "ts_transform" only)* This disables the conversion between <TYPOHEAD> and <Hx> sections.

preserveTags	list of tags	**(DEPRECIATED)**
		Here you may specify a list of tags - possibly user-defined pseudo tags - which you wish to preserve from being removed by the RTE. See the information about preservation in the description of transformations.
		Example:
		In the default TypoScript configuration of content rendering the tags typotags <LINK>, <TYPOLIST> and <TYPOHEAD> are the most widely used. However the <TYPOCODE>-tag is also configured to let you define a section being formatted in monospace. Lets also imaging, you have defined a custom tag, <MYTAG>. In order to preserve these tag from removal by the RTE, you should configure like this.
		`RTE.default.proc {`
		`  preserveTags = TYPOCODE, MYTAG`
		`}`
		Relates to the transformation 'ts_preserve'

dontConvBRtoParagraph	boolean	*(Applies for "ts_transform" only (function divideIntoLines))* By default tags in the content are converted to paragraphs. Setting this value will *prevent* the convertion of -tags to new-lines (chr(10))
internalizeFontTags	boolean	*(Applies for "ts_transform" only (function divideIntoLines))* This splits the content into font-tag chunks. If there are any <P>/<DIV> sections inside of them, the font-tag is wrapped AROUND the content INSIDE of the P/DIV sections and the outer font-tag is removed. This functions seems to be a good choice for pre-processing content if it has been pasted into the RTE from eg. star-office. In that case the font-tags is normally on the OUTSIDE of the sections.
allowTagsOutside	commalist of strings	*(Applies for "ts_transform" only (function divideIntoLines))* Enter tags which are allowed outside of <P> and <DIV> sections when converted back to database. Default is "img" **Example:** IMG,HR
allowTagsInTypolists	commalist of strings	*(Applies for "ts_transform" only (function divideIntoLines))* Enter tags which are allowed inside of <typolist> tags when content is sent to the database. Default is "br,font,b,i,u,a,img,span"

Property:	Data type:	Description:
allowTags	commalist of strings	*(Applies for "ts_transform" only (function divideIntoLines))* Tags to allow. Notice, this list is *added* to the default list, which you see here: *b,i,u,a,img,br,div,center,pre,font,hr,sub,sup,p,strong,em,li,ul,ol,blockquote,strike,span* If you wish to deny some tags, see below.
denyTags	commalist of strings	*(Applies for "ts_transform" only (function divideIntoLines))* Tags from above list to disallow.
HTMLparser_rte HTMLparser_db	->HTMLparser	*(Applies for "ts_transform" only (function divideIntoLines))* This is additional options to the HTML-parser calls which strips of tags when the content is prepared for the RTE and DB respectively. You can configure additional rules, like which other tags to preserve, which attributes to preserve, which values are allowed as attributes of a certain tag etc. .nestingGlobal for HTMLparser_db is set by default to "b,i,u,a,center,font,sub,sup,strong,em,strike,span" unless another value is set. Also B/I tags are mapped to STRONG/EM tags in the RTE direction and vise versa. This parsing is done on a per-line basis, so you cannot expect the paragraph tags (P or DIV) to be included. **Notice** the ->HTMLparser options, "keepNonMatchedTags" and "htmlSpecialChars" is NOT observed. They are preset internally
dontRemoveUnknownTags_db	boolean	*(Applies for "ts_transform" only (function divideIntoLines))* Direction: To database Default is to remove all unknown tags in the content going to the database. (See HTMLparser_db above for default tags). Generally this is a very usefull thing, because all kinds of bogus tags from pasted content like that from Word etc. will be removed to have clean content in the database. However this disables that and allows all tags, that are not in the HTMLparser_db-list.
dontUndoHSC_db	boolean	*(Applies for "ts_transform" only (function divideIntoLines))* Direction: To database Default is to re-convert literals to characters (that is < to <) outside of HTML-tags. This is disabled by this boolean. (HSC means HtmlSpecialChars - which is a PHP function)
dontProtectUnknownTags_rte	boolean	*(Applies for "ts_transform" only (function divideIntoLines))* Direction: To RTE Default is that tags unknown to HTMLparser_rte is "protected" when sent to the RTE. This means they are converted from eg <MYTAG> to <MYTAG>. This is normally very fine, because it can be edited plainly by the editor and when returned to the database the tag is (by default, disabled by .dontUndoHSC_db) converted back. Setting this option will prevent unknown tags from becoming protected.
dontHSC_rte	boolean	*(Applies for "ts_transform" only (function divideIntoLines))* Direction: To RTE Default is that all content outside of HTML-tags is passed through htmlspecialchars(). This will disable that. (opposite to .dontUndoHSC_db) This option disables the default htmlspecialchars() conversion.
dontConvAmpInNBSP_rte	boolean	*(Applies for "ts_transform" only (function divideIntoLines))* Direction: To RTE By default all codes are NOT converted to which they naturally word (unless .dontHSC_rte is set). You can disable that by this flag.
allowedFontColors	list of HTMLcolors	*(Applies for "ts_transform" only (function divideIntoLines))* Direction: To DB If set, this is the only colors which will be allowed in font-tags! Case insensitive.

Property:	Data type:	Description:
allowedClasses	list of strings	*(Applies for "ts_transform" only (function divideIntoLines))* Direction: To DB Allowed general classnames when content is stored in database. Could be a list matching the number of defined classes you have. Case-insensitive. This might be a really good idea to do, because when pasting in content from MS word for instance there are a lot of <SPAN> and <P> tags which may have class-names in. So by setting a list of allowed classes, such foreign classnames are removed. If a classname is not found in this list, the default is to remove the class-attribute.
skipAlign skipClass	boolean	*(Applies for "ts_transform" only (function divideIntoLines))* If set, then the align and class attributes of <P>/<DIV> sections (respectively) will be ignored. Normally <P>/<DIV> tags are preserved if one or both of these attributes are present in the tag. Otherwise it's removed.
keepPDIVattribs	list of tag attributes (strings)	*(Applies for "ts_transform" only (function divideIntoLines))* "align" and "class" are the only attributes preserved for <P>/<DIV> tags. Here you can specify a list of other attributes to preserve.
remapParagraphTag	string / boolean	*(Applies for "ts_transform" only (function divideIntoLines))* When <P>/<DIV> sections are converted to be put into the database, the tag - P or DIV - is preserved. However setting this options to either P or DIV will force the section to be converted to the one or the other. If the value is set true (1), then it works as a general disable-flag for the whole section-convertion stuff here and the result will be no tags preserved what so ever. Just removed.
useDIVasParagraphTagForRTE	string	*(Applies for "ts_transform" only (function divideIntoLines))* Use <DIV>-tags for sections when converting lines from database to RTE. Default is <P>. Applies only to lines which has NO tag wrapped around already.
preserveTables	boolean	*(Applies for "ts_transform")* If set, tables are preserved
dontFetchExtPictures	boolean	*(Applies for "ts_images")* If set, images from external urls are not fetched for the page if content is pasted from external sources. Normally this process of copying is done.
exitHTMLparser_rte exitHTMLparser_db entryHTMLparser_rte entryHTMLparser_db	boolean/->HTMLparser	*(Applies for all kinds of processing)* Allows you to enable/disable the HTMLparser for the content before (entry) and after (exit) the content is processed with the predefined processors (eg. ts_images or ts_transform). There is no default values set.
disableUnifyLineBreaks	boolean	*(Applies for all kinds of processing)* When entering the processor all \r\n linebreaks are converted to \n (13-10 to 10). When leaving the processor all \n is reconverted to \r\n (10 to 13-10). This options disables that processing...

[page:->PROC]

userCategory:

Properties of each user element category.

Property:	Data type:	Description:
load	string	If set, the a predefined set of user element is loaded into this category. They are always loaded in the key starting with 100 and then forward in steps of 10. Current options are: "images_from_folder": Loads gif,jpg,jpeg,png images from the specified folder (defined by the .path property)
merge	Boolean	If set, then any manually configured user elements are merged onto the ones loaded by the .load operation.

Property:	Data type:	Description:
path	String	*(Applies for load=images_from_folder only)* Sets the path of the folder from which to fetch the images (gif,jpg,jpeg,png) **Example:** .path = fileadmin/istate/

[#]	string/->userElements	Configuration of the user elements.
		The string value is the name of the user element. Language-splitted.
		Example:
		RTE.default.userElements {
		# Category with various elements
		10 = Various elements | Diverse elements
		10 {
		# An image is inserted
		1 = Logo 1 | Bomærke 1
		1.description = This is the logo number 1. | Dette er logo nummer 1
		1.content = <img src="###_URL###fileadmin/istate/curro.png">
		# The text-selection is wrapped with <sup> tags.
		2 = Subscript
		2.description = Selected text is wrapped in <sup>-tags.
		2.mode = wrap
		2.content = ^{|}
		# This submits the selected text content to the script, rte_cleaner.php
		5 = Strip all tags
		5.description = All HTML-codes are removed from selection.
		5.mode = processor
		5.submitToScript = typo3/rte_cleaner.php
		}
		# Category with images from the fileadmin/istate/ folder
		2.load = images_from_folder
		2.merge = 1
		2.path = fileadmin/istate/
		# here the logo from “Various elements” is included as well
		2.1 < .10.1
		}
		# Show the user-button, if not existing
		RTE.default.showButtons = user

[page:->userCategory]

userElements:

Properties of each user elements setup.

Property:	Data type:	Description:
mode	string	Which kind of object it is. Options: “wrap”: If a wrap, then the content is exploded by “|” and wrapped around the current text selection. “processor”: The content is submitted to the php-script defined by .submitToScript. GPvar(“processContent”) carries the selection content of the RTE and GPvar(“returnUrl”) contains the return url. (The “content” property is not used here!) default: The content is just inserted (pasted into) at the cursor or substituting any current selection.
description	string	A short description shown beneath the user element title (which is in bold)
content	string	The content inserted/wrapped into the RTE
submitToScript	string	*(Applies only to mode=processor)* PHP to which the current text selection of the RTE is submitted. The script must be relative to the site-url or a full url starting with http://... **Example:** .submitToScript = typo3/rte_cleaner.php or .submitToScript = http://www.domain.org/some_extenal_script.php

Property:	Data type:	Description:
dontInsertSiteUrl	boolean	If set, the marker ###_URL### in the content property's content IS NOT substituted by the current site url. Normally you wish to do this for all image-references which must be prepended with the absolute url in order to display correctly in the RTE!

[page:->userElements]

userLinks:

Properties of user links

Property:	Data type:	Description:
url	string	The url. If set, the marker ###_URL### in the content property's content is substituted by the current site url.
description	string	A short description shown beneath the link title (which is in bold)
target	string	Default target (if isset())

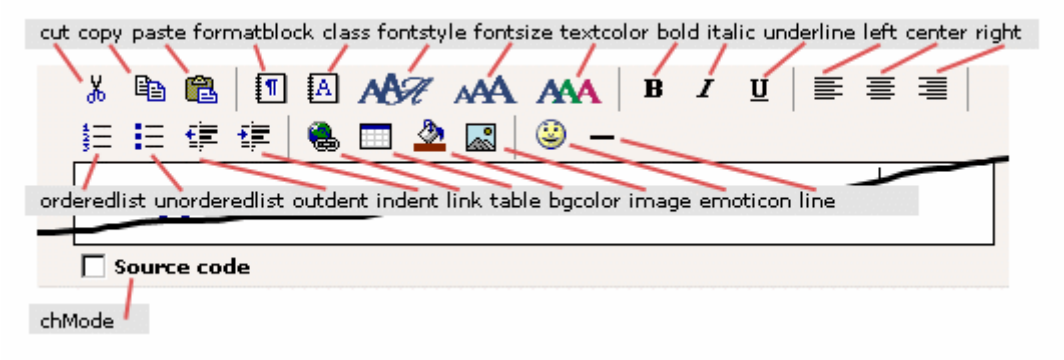
[page:->userLinks]

Toolbar keylist

This is the complete list of keys for the toolbar:

cut copy paste formatblock class fontstyle fontsize textcolor bold italic underline left center right orderedlist unorderedlist outdent indent link table bgcolor image emoticon line user chMode

And this is how the keys relate to the buttons:



HTMLparser:

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

Property:	Data type:	Description:
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	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 &...;) “-1” does the opposite of “1” - converts < to <, > to >, " to “ etc.

[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].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.
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]