

# *Content Construction Kit (CCK)*



# *Agenda*

- Why CCK?
- Creating fields and widgets
- Adding content
- Displaying fields
- Other CCK modules
- CCK version 3
- CCK for Drupal 7

# *Drupal's Swiss Army Knife*



# *No programming needed*



# *Development setup*

- CCK
- Views
- Token
- Devel & Devel Generate
- Advanced Help
- Backup and Migrate or Demo
- Admin Menu or Admin
- Admin Role
- Admin theme:
  - Root Candy
  - Seven
  - Cleanr

# *Basic fields*

- What is a field?
- Text
- Number
- Optionwidgets
- Nodereference
- Userreference
- Fieldgroup

# *Basic widgets*

- What is a widget?
- Textfield
- Textarea
- Drop-down select list
- Checkboxes
- Radios
- Auto-complete

# Advanced help



## CCK Demo

### admin

- My account
- ▷ Create content
- ▽ Administer
  - ▷ Content management
  - ▷ Site building
  - ▷ Site configuration
  - ▷ Generate items
  - ▷ Panels
  - ▷ User management
  - ▷ Reports
  - Advanced help
  - Help
- Log out

[Home](#) » [Administer](#) » [Help](#)

## CCK help index

---

Enable the search module to search help.

- Fields and Widgets
  - Nodereference field
  - Number field
  - Optionwidgets
  - Text field
  - Userreference field
- 'Manage fields' tab
  - Add fields and groups
    - Add a new field
    - Add an existing field : share a field across content types
    - Add a new group
  - Rearrange fields and groups
  - Remove fields and groups
- Theming CCK data in nodes
  - Node templates
  - Field templates
  - Formatter theme functions

# Create fields

Home > Administer > Content management > Example

Example

Edit

Manage fields

Display fields

? Add fields and groups to the content type, and arrange them on content display and input forms. You can add a field to a group by dragging it below and to the right of the group.

Label	Name	Type	Operations
+ Title	Node module form.		
+ Menu settings	Menu module form.		
+ Body	Node module form.		

## Add

### + ? New field

Label

field\_

Field name (a-z, 0-9, \_)

- Select a field type -

? Type of data to store.

- Select a widget -

Form element to edit the data.

### + ? New group

Label

group\_

Group name (a-z, 0-9, \_)

Save

# Drag into place

Home > Administer > Content management > Example

Example

Edit

Manage fields

Display fields

🔗 Add fields and groups to the content type, and arrange them on content display and input forms. You can add a field to a group by dragging it below and to the right of the group.

Label	Name	Type	Operations
⊕ Title	Node module form.		
⊕ Menu settings	Menu module form.		
⊕ <b>🔗 New field</b>			
<input type="text" value="My Text field"/> Label	field_ <input type="text" value="text"/> Field name (a-z, 0-9, _)	<input type="text" value="Text"/> ▼ 🔗 Type of data to store.	<input type="text" value="Text field"/> ▼ Form element to edit the data.
⊕ Body	Node module form.		

Add

⊕ **🔗 New group**

Label

group\_   
Group name (a-z, 0-9, \_)

\* Changes made in this table will not be saved until the form is submitted.

Save

# Adjust field & widget settings

Allowed HTML tags: <a> <b> <big> <code> <del> <em> <i> <ins> <pre> <q> <small> <span> <strong> <sub> <sup> <tt> <ol> <ul> <li> <p> <br> <img>

▼ Default value

**My Number Field:**

▶ PHP code

Global settings

These settings apply to the *My Number Field* field in every content type in which it appears.

Required

**Number of values:**

 ▼

Maximum number of values users can enter for this field.

'Unlimited' will provide an 'Add more' button so the users can add as many values as they like.

**Warning! Changing this setting after data has been created could result in the loss of data!**

**Minimum:**

**Maximum:**

# Drag 'n drop fields and groups

🔗 Add fields and groups to the content type, and arrange them on content display and input forms. You can add a field to a group by dragging it below and to the right of the group.

Label	Name	Type	Operations
⊕ Title	Node module form.		
⊕ Menu settings	Menu module form.		
⊕ <b>🔗 New group</b>			
<input type="text" value="Admin"/> Label	group_ <input type="text" value="admin"/> Group name (a-z, 0-9, _)		
⊕ My Text field *	field_text	Text	<a href="#">Configure</a> <a href="#">Remove</a>
⊕ <b>🔗 New field</b>			
<input type="text" value="My Number Field"/> Label	field_ <input type="text" value="number"/> Field name (a-z, 0-9, _)	<input type="text" value="Integer"/> ▼ 🔗 Type of data to store.	<input type="text" value="Text field"/> ▼ Form element to edit the data.
⊕ Body	Node module form.		

Add

\* Changes made in this table will not be saved until the form is submitted.

Save

# Optionwidgets

## Example basic information

### Label:

### Widget type:

## Allowed values

Create a list of options as a list in **Allowed values list** or as an array in PHP code. These values will be the same for *My Text field* in all content types.

The 'checkboxes/radio buttons' widget will display checkboxes if the multiple values option is selected for this field, otherwise radios will be displayed.

### Allowed values list:

  
  

The possible values this field can contain. Enter one value per line, in the format key|label. The key is the value that will be stored in the database, and it must match the field storage type (*text*). The label is optional, and the key will be used as the label if no label is specified.

Allowed HTML tags: `<a>` `<b>` `<big>` `<code>` `<del>` `<em>` `<i>` `<ins>` `<pre>` `<q>` `<small>` `<span>` `<strong>` `<sub>` `<sup>` `<tt>` `<ol>` `<ul>` `<li>` `<p>` `<br>` `<img>`

—▷ [PHP code](#)

# Optionwidgets output

Home > Create content

## Create Example

Title: \*

—▶ [Menu settings](#)

Related nodes:

 ▼

Admin

**My Text field:**

- Red things
- Yellow things
- Blue things

**My Number Field:**

# *Nodereference - basic*

## Content types that can be referenced:

- Example
- Page
- Panel

## Related nodes:

- None -
- None -
- 101] Sino Sino
- 102] Turpis Praemitto
- 103] Eum Luptatum
- 104] Imputo
- 105] Uxor Iriure Letalis Valde Oppeto Plaga Ullamcorper
- 106] Elit
- 107] Sit Gemino
- 108] Si Mos Damnum Ad Vulputate Et Vero Zelus
- 109] Antehabeo In Capto Euismod Caecus Vero Iriure
- 110] Vel Adipiscing Roto Eum Quia Cui
- 111] Premo Gravis Velit Humo Nunc
- 112] Roto Autem
- 113] Veniam Iriure Usitas Wisi Pecus Elit
- 114] Premo Vulpes
- 115] Roto Autem Adipiscing Abdo Loquor
- 116] Quis Eum Quia Cui
- 117] Erat
- 118] Odio
- 119] Exerci Meus Quibus Rusticus Defui Molior Vulpes

# Nodereference - w/Views

## Content types that can be referenced:

- Example
- Page
- Panel
- Story

## Advanced - Nodes that can be referenced (View)

### View used to select the nodes:

latest\_stories

Choose the "Views module" view that selects the nodes that can be referenced.

Note:

- Only views that have fields will work for this purpose.
- This will discard the "Content types" settings above. Use the view's "filters" section instead.
- Use the view's "fields" section to display additional informations about candidate nodes on node creation/edition form.

None -

itle: [1] Damnum Imputo Obruo Immitto - Name: Anonymous - Updated date: 12/09/2008 - 09:34

itle: [2] Ibidem Ea Lobortis Lobortis Nibh Vel Secundum - Name: Anonymous - Updated date: 12/09/2008 - 09:34

itle: [3] Humo Eum Laoreet - Name: admin - Updated date: 12/09/2008 - 09:34

itle: [4] Natu Populus Pneum Praesent Quae - Name: Anonymous - Updated date: 12/09/2008 - 09:34

itle: [5] Sagaciter Facilisi Ullamcorper Gemino Gemino Cui - Name: admin - Updated date: 12/09/2008 - 09:34

itle: [6] Pagus Distineo Pala Pala Molior Comis Utinam - Name: Anonymous - Updated date: 12/09/2008 - 09:34

itle: [7] Vero Dolus Adipiscing Abdo - Name: admin - Updated date: 12/09/2008 - 09:34

# Nodereference - w/Views

View *latest\_stories*, displaying items of type **Node**.

Export

Clone

View "Page"

Defaults

**Page** *Display the view as a page, with a URL and menu links.* [Remove display](#)

**Page**

Page

**Basic settings**

Name: **Page**  
Title: *None*  
Style: *Unformatted*  
Row style: *Fields*  
Use AJAX: *No*  
Use pager: *No*  
Items to display: *10*  
Distinct: *No*  
Access: *Unrestricted*  
Exposed form in block: *No*  
Header: *None*

**Relationships**    
*None defined*

**Arguments**    
*None defined*

**Fields**    
Node: *Title* *Title*  
User: *Name* *Name*  
Node: *Updated date* *Updated date*

**Sort criteria**    
*None defined*

**Filters**    
*None defined*

**Page: Row style options**

**Inline fields:**

Node: Title

User: Name

Node: Updated date

Inline fields will be displayed next to each other rather than one after another.

**Separator:**

The separator may be placed between inline fields to keep them from squishing up next to each other. You can use HTML in this field.

# Choose display options

Home > Administer > Content management > Example

Example

Edit

Manage fields

Display fields

Basic

RSS

Token

Configure how this content type's fields and field labels should be displayed when it's viewed in teaser and full-page mode. Use the 'Exclude' checkbox to exclude an item from the \$content value passed to the node template.

Field	Label	Teaser	Exclude	Full node	Exclude
<b>Admin</b>	Above ▼	simple ▼	<input type="checkbox"/>	fieldset ▼	<input type="checkbox"/>
My Text field	Inline ▼	Trimmed ▼	<input type="checkbox"/>	Default ▼	<input type="checkbox"/>
My Number Field	<Hidden> ▼	9,999 ▼	<input type="checkbox"/>	9,999 ▼	<input checked="" type="checkbox"/>

Save

# Customize display

## Template variables

CCK makes the following variables available in your theme's node templates:

### **`<FIELD_NAME>_rendered`**

Contains the rendered html for the field, including the label and all the field's values, with the settings defined on the **Display fields** tab.

### **`<GROUP_NAME>_rendered`**

Contains the rendered html for the fieldgroup (if any), including the label and all the group's fields, with the settings defined on the **Display fields** tab.

This variable therefore includes the html contained in all the `<FIELD_NAME>_rendered` variables for the group's fields.

### **`$FIELD_NAME`**

Contains the raw values of the fields, in the usual array-format used internally by CCK. What you find in there depends on the field type.

Each value also contains a `'view'` element, that holds the ready-to-display value as rendered by the formatter.

For instance:

```
array(
  0 => array(
    'nid' => 5,
    'view' => '<a href="node/5">Title of node 5</a>',
  ),
);
```

**Raw data are not sanitized for output, it is therefore not advised to use them directly.** Use the `'view'` value, or run the values through `content_format()`.

## *Other fields*

- Imagefield
- Filefield
- Date
- Computed field
- Link
- Emfield
- and many many more...

# *Built-in integration*

- Views
- Panels
- Token
- Diff / Revision control
- Devel Generate
- Rules

# *Extended functionality*

- Content Profile
- Auto Nodetitle
- Feed API
- Prepopulate