



Drupal



Was ist Drupal?

- Drupal ist cool!



Wichtige Begriffe

- Dries Buytaert
- Lullabot
- Drupal.org/ Drupalcenter.de
- Modules,Themes
- CCK, Views, Taxonomy



Praxis

Installation, Konfiguration und
Anwendung von Drupal



Installation

- Download
- Entpacken
- Datenbank anlegen
- Evtl /sites/default/setting.php editieren
- Install.php aufrufen und ab geht die Post



Nach der Installation

- Ersten Benutzer anlegen und direkt danach das Passwort ändern!
- Dann in `administer/logs/status report` schauen und evtl Fehlermeldungen beheben



Theorie

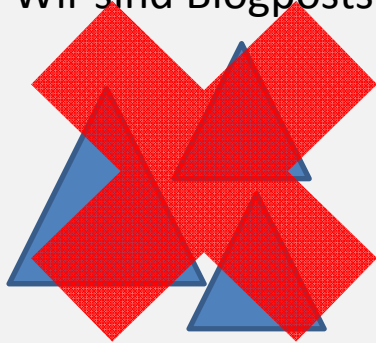


Funktionsweise „normales“ CMS

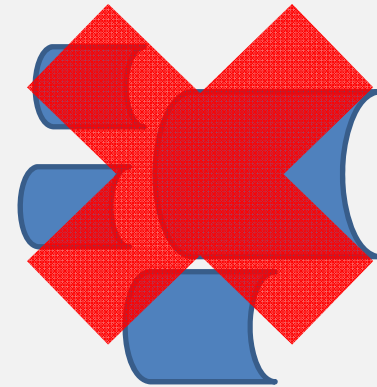
- Hierarchische Struktur
- Workflow: Backend -> Strukturwahl -> Inhalt



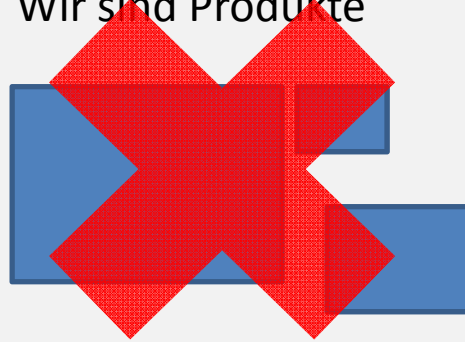
Wir sind Blogposts



Wir sind News



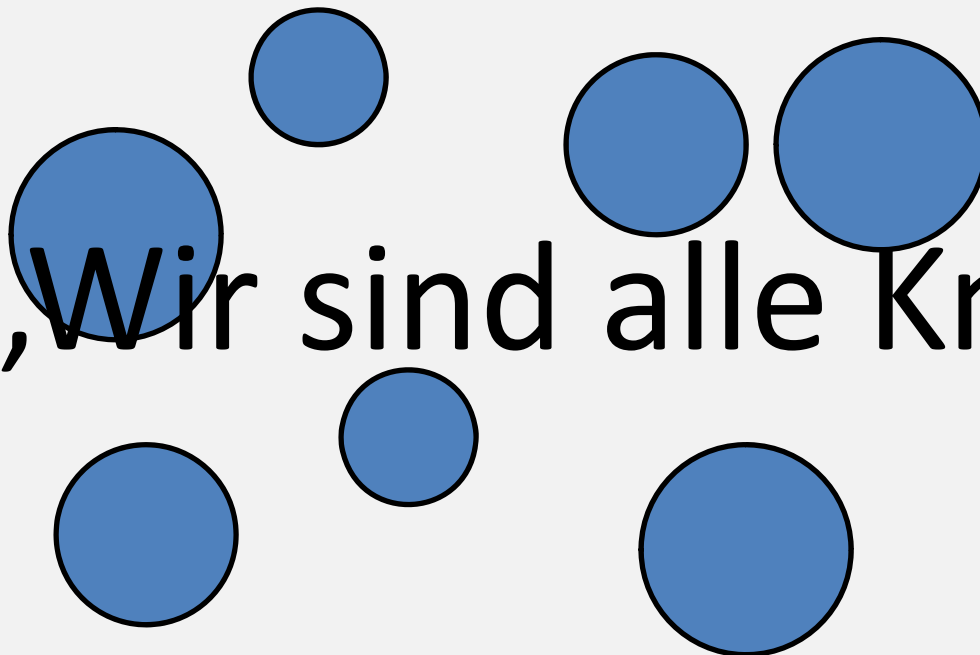
Wir sind Produkte





The Drupal way of life

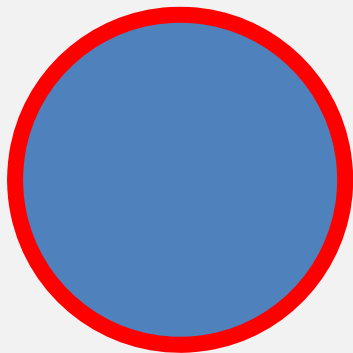
- Freie Knoten

A diagram consisting of seven blue circles of varying sizes scattered across the slide. The circles are arranged in a way that they appear to be nodes in a network, with some larger and some smaller. The text "Wir sind alle Knoten" is overlaid on the circles.

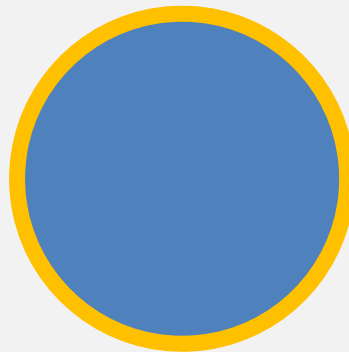
„Wir sind alle Knoten“



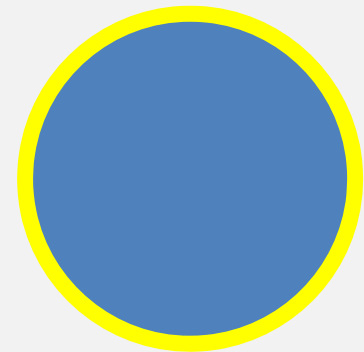
Produktknoten

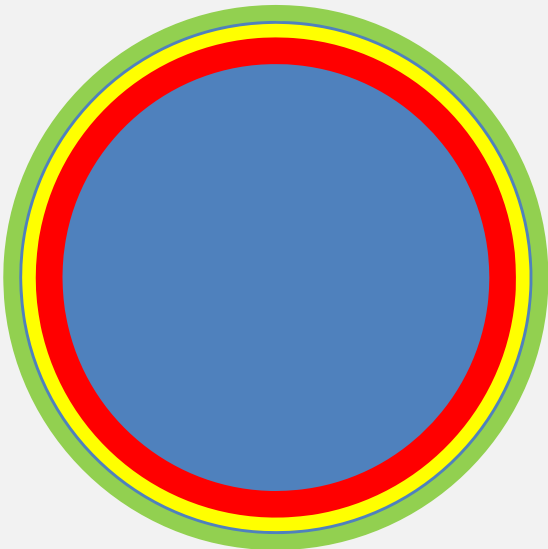
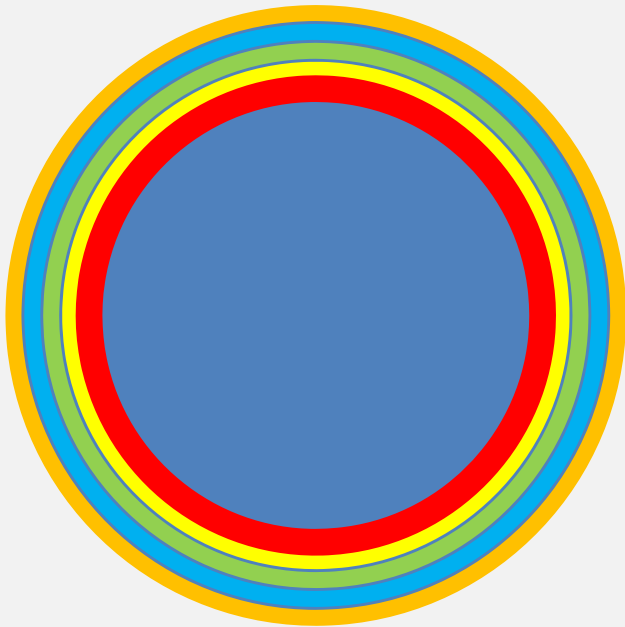


Newsknoten



Blogpostknoten



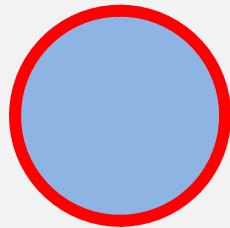




The Drupal way of life

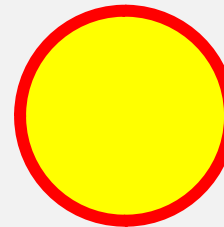
- Freie Knoten
- Zusammengehalten durch Taxonomie

Ich bin Inhalt



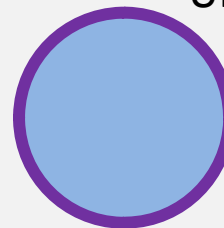
Und habe einen **roten Rahmen** und **blaue Füllung**

Ich auch



Und habe einen **roten Rahmen** und **gelbe Füllung**

Und ich auch



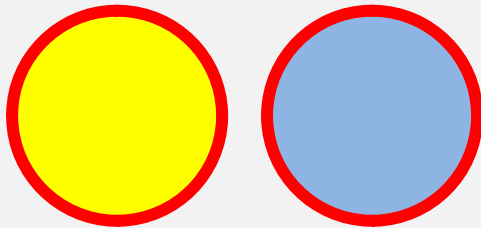
Und habe einen **lila Rahmen** und **blaue Füllung**



The Drupal way of life

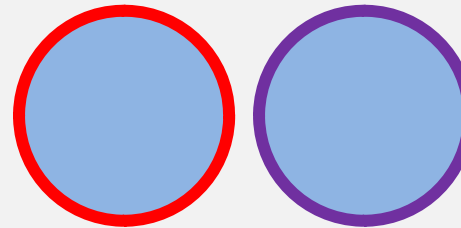
- Freie Knoten
- Zusammengehalten durch Taxonomie
- Ausgabe durch Filter (CCK + Views)

Wir mögen uns



roten Rahmen

Wir auch



blaue Füllung



The Drupal way of life

- Workflow: Inhalt -> Struktur



Modulentwicklung



Für wen dieser Workshop ist

- Für Leute, die in der „Vielleicht ist Joomla doch besser“-Phase sind (siehe Lernkurve)
- Für Leute, die spezifische Probleme haben, für die es kein Modul gibt



Für wen dieser Workshop NICHT ist

- Leute, die keine PHP, HTML oder CSS Kenntnisse haben
- Leute, die schon umfangreiche Module geschrieben haben
- Core-Entwickler :-)



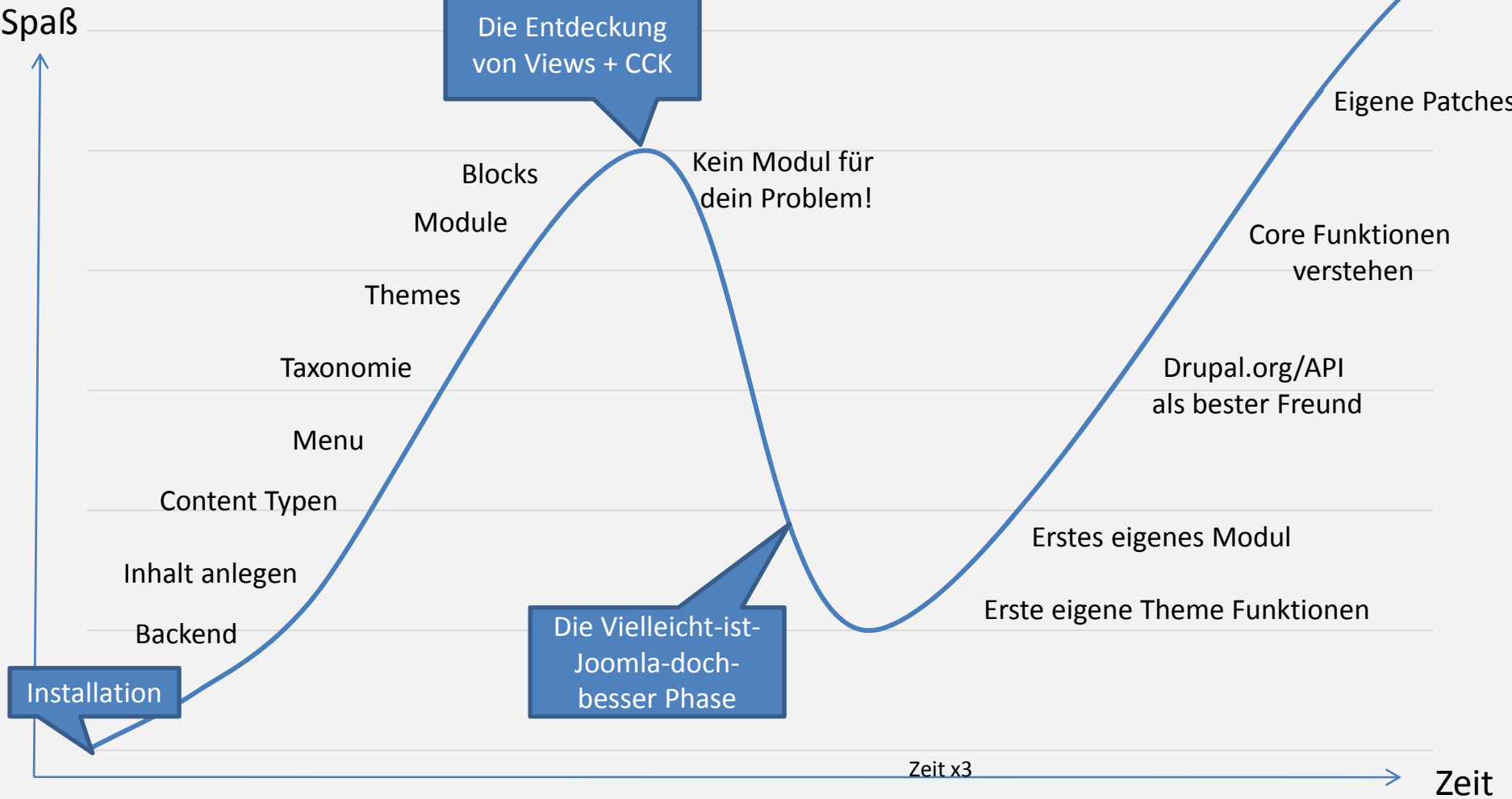
Voraussetzungen

- Man hat Drupal/Themes/Module installiert
- Gute PHP Kenntnisse. Keine Angst, Drupal ist keine OOP!
- Man weiß, was Views, CCK und Taxonomie ist
- Geduld und eine gute Kaffeemaschine



Lernkurve

Drupal
Himmel





Verzeichnisse

- Module in /modules NEIN
- Themes in /themes NEIN

- Alles in /sites



Sites Ordner

- /all
- /default
- (/www.domain1.de)



Drupal Core

- Anschauen JA
- Ändern NEIN



Ressourcen

- Drupal.org/search
- Drupal.org/api
- Groups.drupal.org
- Drupal.org/handbook
- Drupalcenter.de
- [Drupalschool + Drupaldojo](#)
- [Lullabot](#)



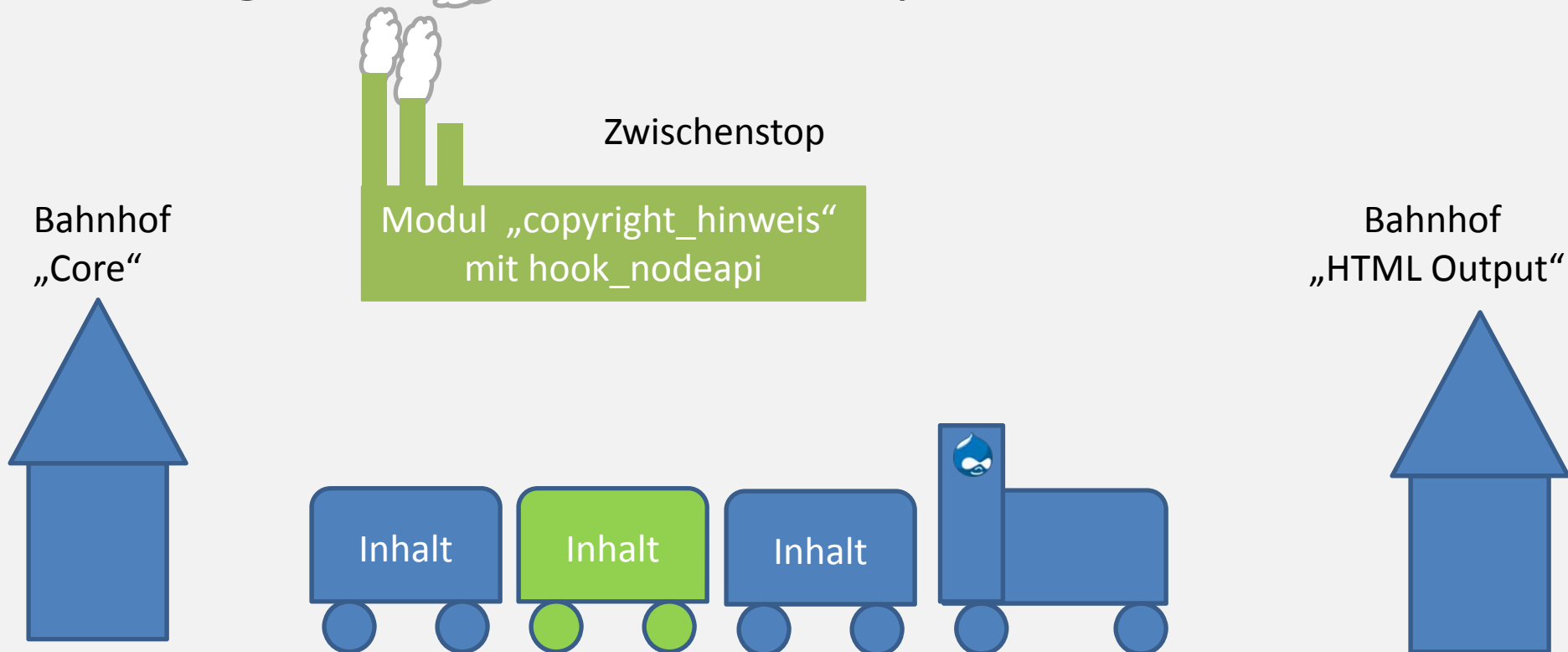
Datenbanken

- `Db_query()`
- `{tabellenname}` wg Prefixen
- `%s, %d, %f, %b, %%` um Injests zu verhindern



Hooks

- Über Hooks kann man in laufende Prozesse eingreifen und diese manipulieren.





Hooks

- Hooks sind Callbacks. Heißen aber nicht so!
- Namenskonvention statt Listener.
- Beim Einloggen wird der Hook „user“ aufgerufen. Daraufhin werden alle Funktionen, die nach `modulename_user()` benannt sind aufgerufen.



Modul anlegen

- Ordner in `/sites/x/modules`
- `Modulname.info`
- `Modulname.install`
- `Modulname.module`



Rechte setzen

- Einsatz von hook_perm

```
function testmodule_perm() {  
  return array ('enter testmodule');  
}
```



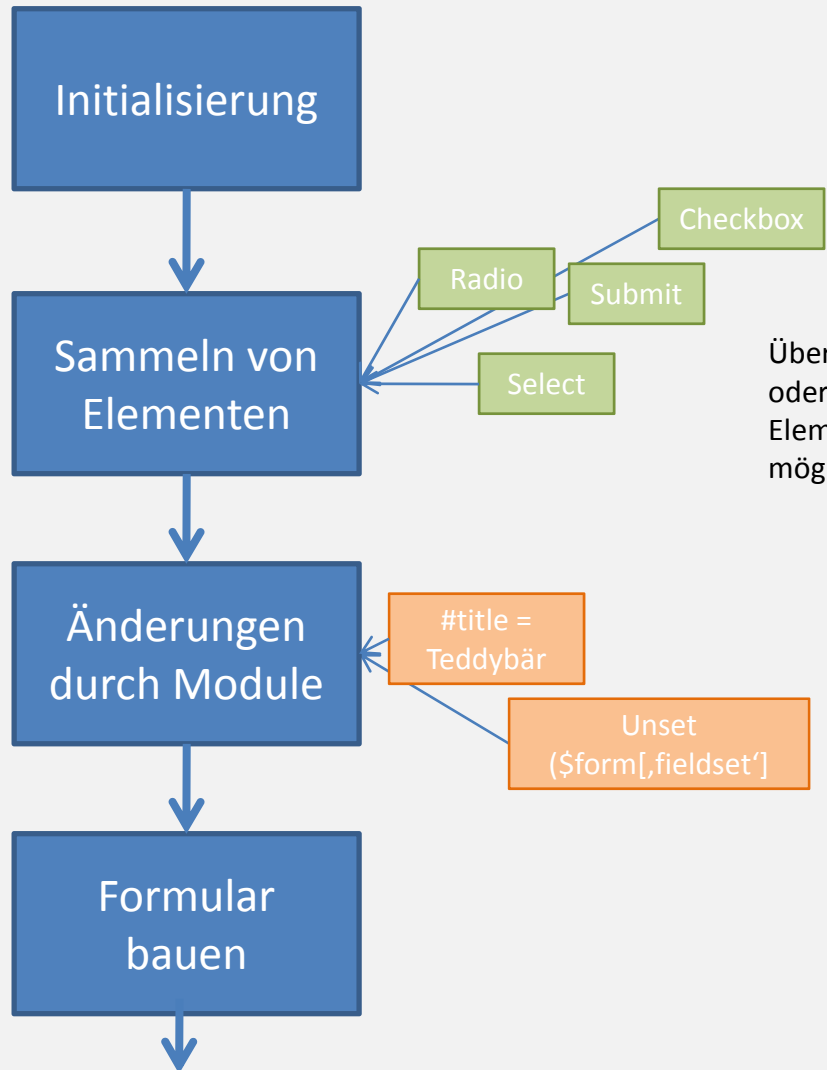
Der Formularprozess (1)

Das Formular wird mit `drupal_get_form()` aufgerufen.
Token wird gesetzt.

Drupal schaut nach, welche Menü-Elemente überhaupt existieren.

Module können durch `hook_form_alter()` Elemente ändern/löschen oder hinzufügen

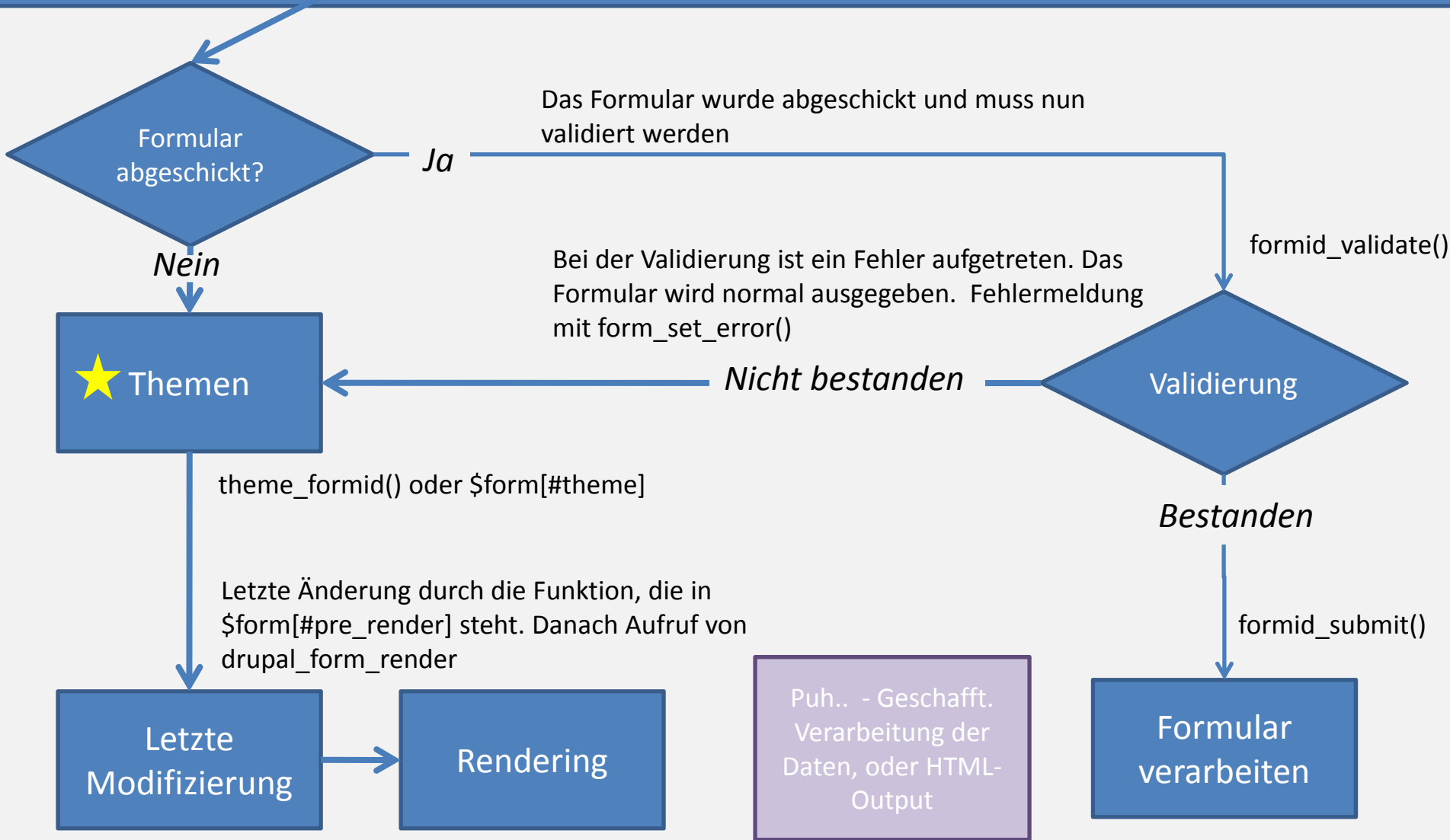
Aufruf von `form_builder()`



Überschreiben von Bestehenden oder Anlegen von eigenen Elementen mit `hook_elements` möglich!



Der Formularprozess (2)





Hook_form_alter

- Nimmt Einfluss auf das Edit-Formular eines Nodes
- Abfrage nach dem Typ des angezeigten Formulars mit `$form[type][#value]`



Formulare themen

- Über normale Theme Funktion
- Theme_form in form.inc
- Theme_form_element in form.inc

Spezifisch für jeden Button:

- theme_submit
- Theme_button



CCK: Standardfunktionen

Node_type

Speichert die Standard-Daten jedes angelegten Content-Typs (Name, Beschreibung, Hilfe)

node

Speichert die eindeutigen Daten eines Knotens (nid, vid, title, date, uid)

Node_revision

Speichert den Standard-Inhalt des Knotens (title, body)



CCK: Speichern zusätzlichen Inhaltes

Content_type_deintyp

Speichert den Inhalt der zusätzlich durch CCK angelegten Felder (außer Titel, Body)

Ausnahme bei
multiple Values:

Content_field_multiplefield

Felder, die mit dem Wert „multiple“ belegt sind, bekommen eine eigene Datenbanktabelle.



CCK: Speichern der CCK-Felder

Feldspezifische Daten



Node_field

Speichert die Einstellungen der Felder, die für irgendeinen Content-Typ bereits angelegt wurden.

Feldname, Feldtyp, Einstellungen, Required, Multiple

Speichert Instanzen.
Feldspezifische Daten
abhängig vom Content Typ



Node_field_instance

Da einmal angelegte Felder wiederverwendet werden können. Werden hier die spezifischen Einstellungen die einen Content-Type betreffen gespeichert.

Feldname, Contenttyp, Label, Widget_Settings, Display_Settings



CCK Hooks Übersicht

- `_widget` definieren i.E. das, was der Benutzer bei der Administration sieht
- `_field` definiert allgemeine Eigenschaften und die Formulare
- `_formatter` ist für die Ausgabe zuständig

Ausgabe der Formulare für
den Benutzer

`_widget_info()`

`_widget_settings()`

`_widget()`

Eigenschaften und
Verhalten von Feldern

`_field_info()`

`_field_settings()`

`_field()`

Verhalten bei Ausgabe

`field_formatter_info()`

`_field_formatter()`



Übersicht der CCK Hooks

- **Fields**

- **hook_field_info()**: Here you declare the label for your field type(s). This will show up when you click "add field" on a CCK content type.
- **hook_field_settings(\$op, \$field)**: Handles displaying, validating, and saving field settings forms. In addition, manages how they're stored in the database and provides Views integration.
- **hook_field(\$op, &\$node, \$field, &\$node_field, \$teaser, \$page)**: Define the behavior of a field type.
- **hook_field_formatter_info()**: Declare information about a formatter.
- **hook_field_formatter(\$field, \$item, \$formatter, \$node)**: Prepare an individual item for viewing in a browser.

- **Widgets**

- **hook_widget_info()**: Here you declare any widgets associated with your field type. These will show up below the fields when you click "add field" on a CCK content type.
- **hook_widget_settings(\$op, \$widget)**: Handle the parameters for a widget.
- **hook_widget**: Define the behavior of a widget.



Hook_widget_info()

- Definition der Widget. Widgets legen fest, wie die Eingabe durch den Benutzer und die Ausgabe auf dem Screen erscheint. Z.B: Textfeld, Select List.

```
function imceimage_widget_info() {
    return array(
        'imceimage' => array(
            'label' => 'IMCE Image',
            'field types' => array('imceimage'),
        )
    );
}
```



hook_widget_settings()

- Definiert das Eingabeformular, auf der Administrationsseite des Widgets/Feldes.
- Hier speichern von **Widgets**spezifischen Einstellungen. Z.B. Höhe einer Textarea oder verwendete Theme-Funktion.
- Feldspezifische Daten werden in `field_settings()` gespeichert.



Hook_widget()

- Das Verhalten des Widgets wenn es auf der Seite ausgegeben wird (im Editbereich eines Nodes)
- **Wichtig:** Dieser Hook ist nicht dafür zuständig den eingegebenen **Inhalt** zu formatieren, sondern nur die Ausgabe des Formulars!



Hook_field_info()

- Definition über das CCK-Feld das angelegt wird.

Unterschied zu `hook_widget_info()`:
 Widgets regeln die Eingabe/Ausgabe, sind quasi die Vorlagen/Masken.
 Das Field ist der Datentyp

field

widgets

Integer

Text Field

Select list

Check boxes/radio buttons

Single on/off checkbox

Decimal

Text Field

Select list

Auszug aus `/admin/content/types/x/add_field`



Hook_field_settings()

- Speichern von feldspezifischen Daten. Z.B. Angabe über Maximallänge.
- Daten werden automatisch in Datenbank gespeichert.
- Festlegen der Datenbankspalten.



Hook_field()

- Definiert das Verhalten eines Feldes bei dessen Ausgabe in einem Node.
- „view“ wird nicht mehr benötigt, da die Standardausgabe komplett über content.module gehandhabt wird.



Hook_field_formatter_info()

- Festlegen des Namens des Formatters.
- Wird im Display Tab für jede Instanz angezeigt
- Wird später in den Views unter „Options“ angezeigt, nachdem ein Feld hinzugefügt wurde.

| Field | Type | Label | Teaser | Full |
|--------------|------------|------------|-----------------|---------------|
| Single Image | IMCE Image | Above ▼ | Default ▼ | Default ▼ |
| IMCE Image | IMCE Image | <Hidden> ▼ | Gallery-Thumb ▼ | Gallery-Big ▼ |



Hook_field_formatter()

- Definiert das Verhalten/Ausgabe eines Feldes, wenn ein bestimmter Formatter ausgewählt wurde.