

11.12.2014



NEOS

TYPO3 Neos 1.2.0 Das Kompendium

[+] pluswerk
Digitale Leidenschaft

Patrick Lobacher
CEO +Pluswerk GmbH

Feedback erwünscht

- Lieber Neos-Enthusiast!

Ich versuche das TYPO3 Neos Kompendium stets aktuell zu halten und ständig zu ergänzen. Aber dafür brauche ich Deinen Input! Wenn Du Ideen hast, Code-Beispiele, FAQ-Themen oder schlicht Lob (oder Kritik), dann schreibe mir bitte an die folgende Adresse:

patrick [AT] lobacher.de

Viel Spaß mit dem Kompendium!
Patrick Lobacher

Changelog

Datum	Veränderungen
08.08.2013	Initiale Version / Danke an Christian Schwerdt für den domainFACTORY Input
09.08.2013	Korrekturen von Roland Schenke und Michael Oehlhof eingearbeitet - Danke!
10.08.2013	Nginx-Config eingearbeitet. Danke an Christian Kuhn, Christian Müller und Anja Leichsenring
10.08.2013	Troubleshoot-Sektion eingefügt
18.08.2013	Korrekturen von Roland Schenke eingearbeitet - Danke!
18.08.2013	Übersetzung des Kompendiums (zur Version alpha5) von Roland Schenke. Danke!!
12.12.2013	Update auf TYPO3 Neos 1.0 final
15.12.2013	Update für TYPO3 Neos 1.0.1
07.01.2014	Link für Installation unter Shared Hosting und Korrekturen (Danke an Christian Glass!)
03.03.2014	CI auf „LOBACHER.“ umgestellt
05.03.2014	Update auf TYPO3 Neos 1.0.2

Changelog

Datum	Veränderungen
07.03.2014	Installation auf einem <u>all-inkl.com</u> Server integriert. Danke an Mario Janetzko!
16.04.2014	Bugfixing im Bereich „Erweiterung der Site - Plugin 1“ . Danke an Rico Schüppel!
27.04.2014	Korrekturen von Roland Schenke eingearbeitet - Danke!
19.06.2014	Update auf TYPO3 Neos 1.1 final, Uberspace Anleitung von Kerstin Huppenbauer - Danke!
28.06.2014	TYPO3 Surf Deployment Anleitung auf Uberspace Servern zugefügt. Danke an Benjamin Albrecht!
27.08.2014	Bugfix-Release 1.1.1
03.09.2014	Bugfix-Release 1.1.2
18.09.2014	Ergänzung bei Anleitung für Installation auf Domain Factory Servern (Danke an Timo Nußbaum!)
21.11.2014	Update für TYPO3 Neos 1.2.0 beta 1 / Integration „Vor-Konfigurierte Neos Server bei Amazon Web Services (AWS)“ Danke an Michael Schams!!
26.11.2014	Update für TYPO3 Neos 1.2.0 beta 2
11.12.2014	Update für TYPO3 Neos 1.2.0

Was ist **TYPO3 Neos?**



TYPO3 Neos 1.2

Ubiquitous Content just arrived. TYPO3 Neos 1.2 has been released.

[Download TYPO3 Neos 1.2](#)



Take a peek on the next generation of content management.





Content Management Simplified

The screenshot shows the TYPO3 Neos CMS interface. On the left, there's a navigation bar with 'Navigate' and 'Edit & Preview' tabs, and a preview switch between 'In-Place' and 'Raw Content'. The main area displays a large orange box with the text 'Hello.'. On the right, the 'Inspector' panel is open, showing settings for the selected element 'Page'. It includes fields for 'Title' (set to 'Home'), 'Visibility' (with options for 'Hide before' and 'Hide after' both set to 'No date set', and 'Hide in menus' checked), and other properties like 'Admin Guy' and 'Publish (2)'.

Neos makes editing fly. It provides an interface so intuitive, that it requires no training and offers power where needed.

<http://neos.typo3.org/download/marketing-material.html>





Neos (griechisch νέος) heißt übersetzt schlicht "neu"

Die Geschichte von TYPO3 Neos beginnt bei TYPO3 CMS

- TYPO3 CMS ist ein „Enterprise Open Source Content Management Framework“
- TYPO3 CMS existiert seit 1998 / Erfunden vom Dänen Kaspar Skårhøj
- ca. 500.000 Installationen weltweit / > 5 Mio Downloads
- Einsatz in DE z.B. bei > 50% aller DAX 500 Unternehmen, > 50% aller Bundesliga-Vereinen, Discounter, Autovermieter, Öffentliche Träger
- > 6.000 Extensions
- > 100.000 Entwickler weltweit
- > 1500 Agenturen weltweit
- Finanziert und unterstützt von der TYPO3 Association



Die Geschichte von TYPO3 Neos: TYPO3 Phoenix

- Auf den ersten T3DD (TYPO3 Developer Days) im Jahr **2006** wurde der Entschluss gefasst, TYPO3 von Grund auf neu zu schreiben
- Codename: TYPO3 Phoenix (bzw. TYPO3 5.0)
- Einige benötigte Features gab es damals in PHP noch nicht und mußten komplett neu implementiert werden: Content Repository, Aspect Oriented Programming, OP, Dependency Injection, ...
- Mit dem „Berlin Manifesto“ wurde **2008** der Rahmen und die Abgrenzung zum TYPO3 CMS festgelegt
<http://typo3.org/roadmap/berlin-manifesto/>
(Das Manifest ist mittlerweile in Teilen nicht mehr gültig)

Die Geschichte von TYPO3 Neos: TYPO3 Flow und Neos

- Viele Grundfunktionen eines CMS sind nicht CMS-spezifisch (Session-Handling, Datenbank-Handling, Templating, ...) => daher Trennung dieser Funktionalitäten in ein eigenes Framework TYPO3 Flow
- Durch die Einführung von Extbase im Jahr **2009** wurde es möglich, bereits in TYPO3 CMS Extensions zu schreiben, die in TYPO3 Flow mit geringen Änderungen lauffähig sind (sofern sie keine Internas verwenden)
- Am 20. Oktober **2011** wurde das Application Framework **TYPO3 Flow** (ehemals FLOW3) als Final veröffentlicht
- **TYPO3 Neos** ist eine Applikation die auf TYPO3 Flow basiert
- TYPO3 Neos Alpha 7 im Oktober **2013** / Beta 1 am **12.11.2013**
- Erste finale Version 1.0.0 am **10.12.2013**



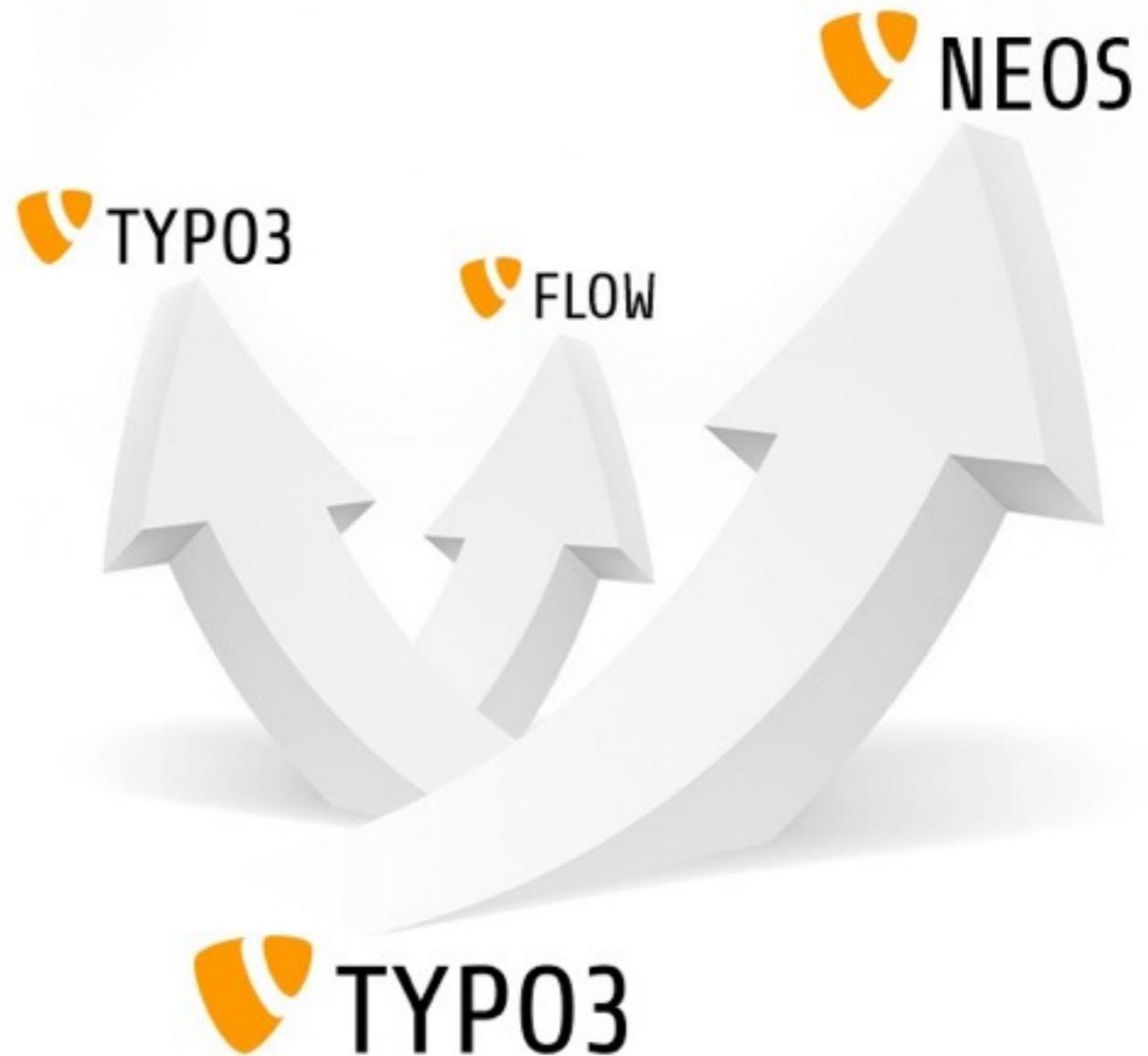
Versionshistorie TYPO3 Neos



Version	Datum
TYPO3 Phoenix Sprint Release 1	31.05.2010
...	...
TYPO3 Phoenix Sprint Release 9	02.08.2012
TYPO3 Neos alpha 1	05.10.2012
TYPO3 Neos alpha 2	19.12.2012
TYPO3 Neos alpha 3	16.02.2013
TYPO3 Neos alpha 4	07.07.2013
TYPO3 Neos alpha 5	07.08.2013
TYPO3 Neos alpha 6	15.10.2013
TYPO3 Neos alpha 7	30.10.2013
TYPO3 Neos beta 1	12.11.2013
TYPO3 Neos beta 2	03.12.2013
TYPO3 Neos 1.0 final	10.12.2013
TYPO3 Neos 1.0.1	13.12.2013
TYPO3 Neos 1.0.2	15.03.2014
TYPO3 Neos 1.1 final	19.06.2014
TYPO3 Neos 1.1.1	27.08.2014
TYPO3 Neos 1.1.2	03.09.2014
TYPO3 Neos 1.2.0 beta 1	14.11.2014
TYPO3 Neos 1.2.0 beta 2	26.11.2014

Die TYPO3 Welt - seit Oktober 2012

- Dachmarke TYPO3
 - TYPO3 CMS
(hat kein eigenes Logo)
 - TYPO3 Flow
 - TYPO3 Neos

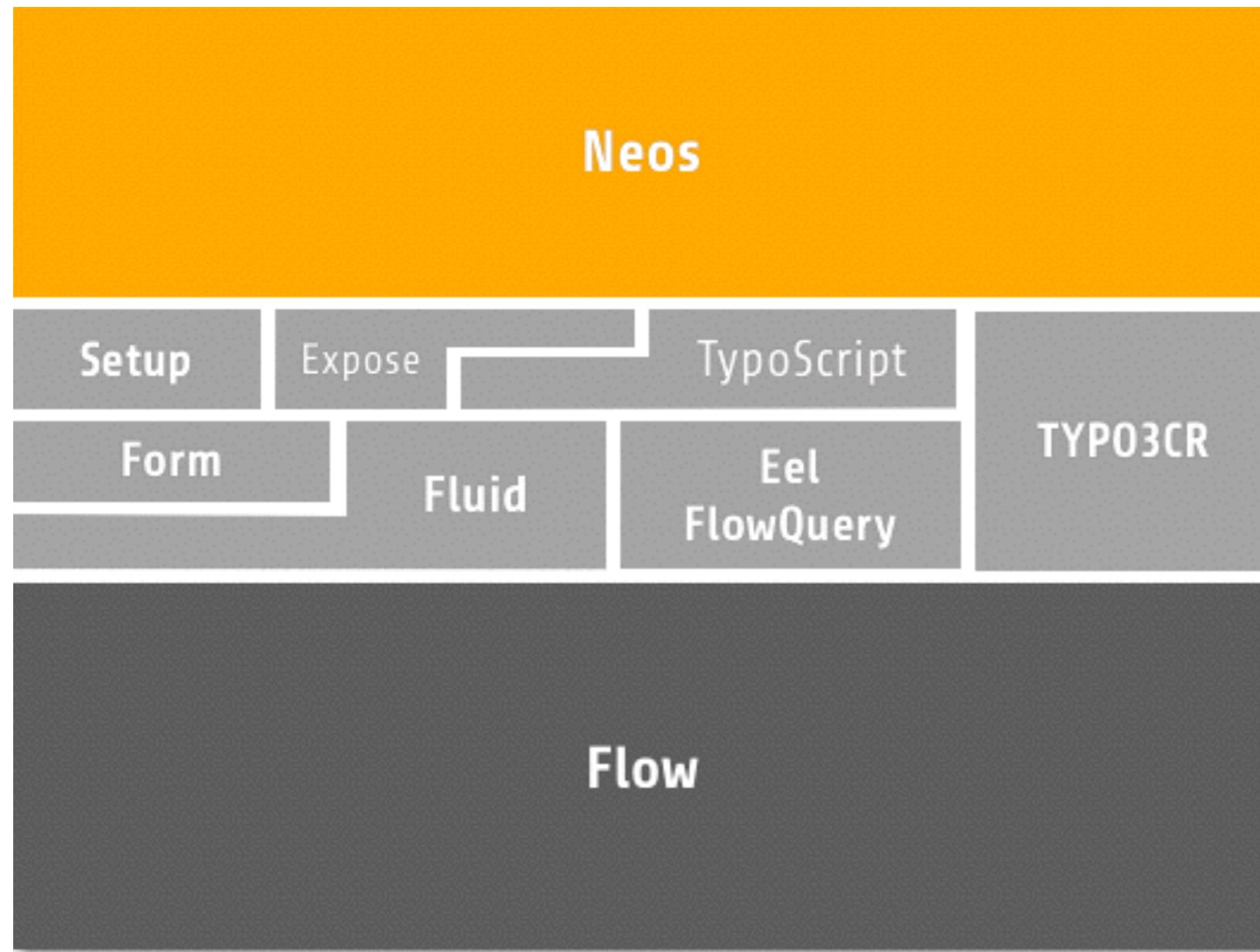


TYPO3 Neos: Positionierung

- Es gibt ab Dezember 2013 zwei unabhängige CMS aus dem Hause TYPO3:
 - **TYPO3 CMS 6.x (4.7, 6.0, 6.1, 6.2 LTS Beta)**
 - **TYPO3 Neos 1.x (1.0, 1.1, 1.2)**
- Technologisch haben beide **NICHTS** miteinander zu tun
- Es gibt **KEINE** Migrationsmöglichkeiten von einem der beiden Systeme zum anderen
- TYPO3 Neos ist **NICHT** der Nachfolger von TYPO3 CMS, sondern ein eigenständiges CMS mit anderem Fokus

Die Architektur von TYPO3 Neos

Die Architektur von TYPO3 Neos - Backend



Fluid

Modern Templating Engine

TYPO3CR

Content Repository (JCR / Sling)

TypoScript

TypoScript 2.0 - next Generation

Forms

Form API & Form Builder

Expose

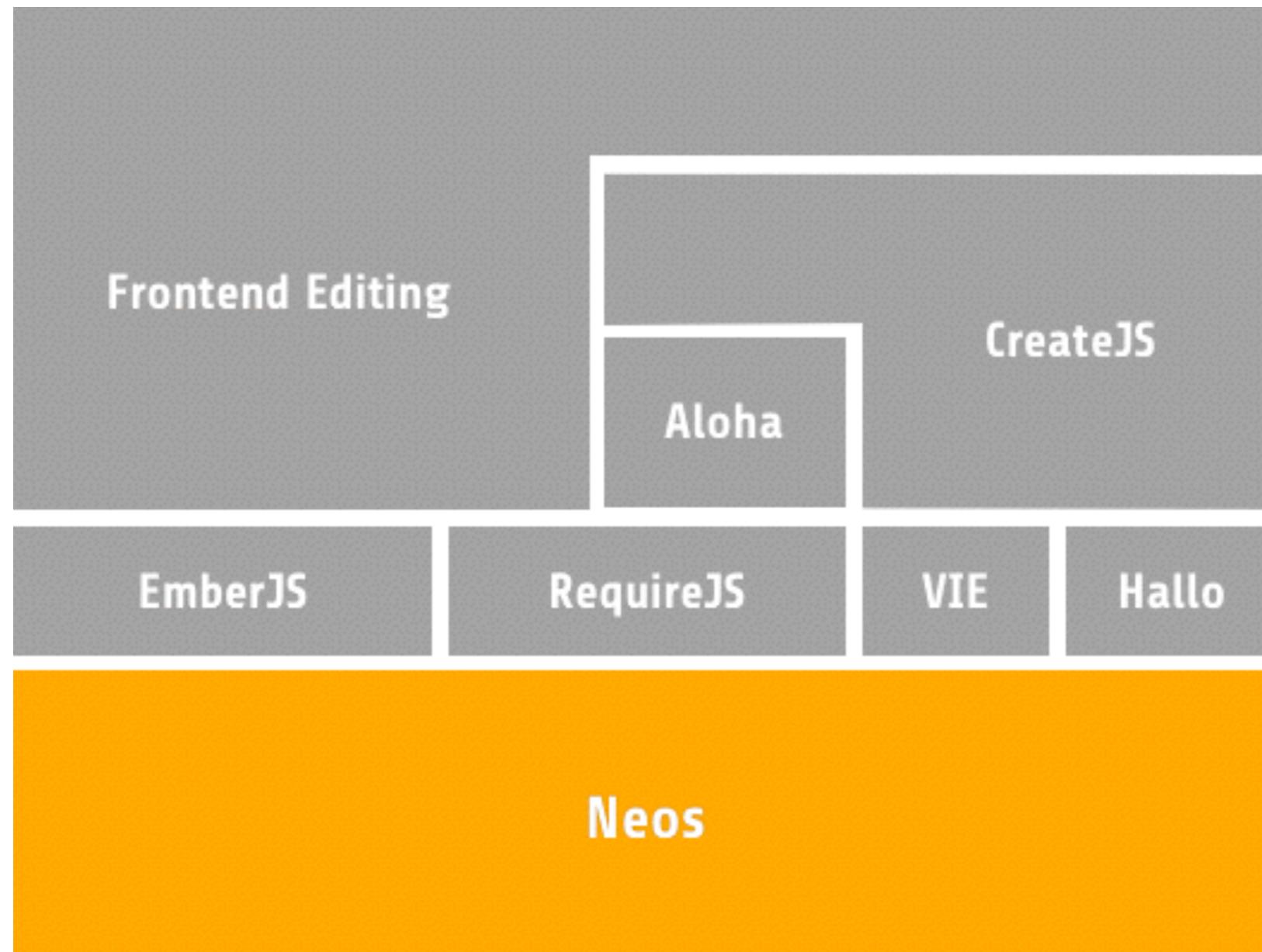
Extensible admin interface

Eel

Embedded Expression Language

FlowQuery

Die Architektur von TYPO3 Neos - Frontend



EmberJS
JavaScript Web Application Framework

Create.js
Web Editing Interface

Aloha / Hallo
HTML5 WYSIWYG Editor

VIE = viejs.org
Semantic Interaction Framework

RequireJS
JavaScript file and module loader

Installation von TYPO3 Neos

Systemvoraussetzungen von TYPO3 Neos

- Webserver (empfohlen ist Apache 2.x mit aktiviertem mod_rewrite Modul)
- PHP 5.3.7 - 5.4.x (minimal wäre PHP 5.3.2 - dort kann es zu Problemen kommen)
- Folgende Funktionen müssen in PHP aktiviert sein: **system**, **shell_exec**,
escapeshellcmd, **escapeshellarg**, **proc_open** und **exec()**
- php.ini: memory_limit = 512M oder höher (empfohlen 1024M)
- php.ini: xdebug.max_nesting_level = 500 (sofern xdebug verwendet wird)
- php.ini: Fügen sie die folgende Optionen ans Ende hinzu: detect_unicode = Off
- php.ini: Zudem muss Magic_Quotes ausgeschaltet werden: magic_quotes_gpc = Off
- php.ini: Die Kommandozeile von Flow benötigt ferner noch eine Zeitzoneneinstellung: date.timezone= "Europe/Berlin"

Systemvoraussetzungen von TYPO3 Neos

- Wichtig ist auch, dass das PHP auf der Kommandozeile ebenfalls mindestens Version 5.3.7 ist (und über die angegebenen php.ini Einstellungen verfügt) - dies kann mit dem folgenden Befehl überprüft werden

php --version

- MySQL 5.1.50 - 5.x.x (zum Beispiel - grundsätzlich kann jede zum Doctrine DBAL kompatible Datenbank verwendet werden)
- Zugang zur Konsole (root User!), wenn die Zugangsrechte vom Hoster nicht entsprechend gesetzt worden sind. Bei den meisten Hostern ist dies aber der Fall - dann reicht ein normaler User.

Verzeichnisstruktur

- Die Installation soll letztlich in folgende Verzeichnisstruktur erfolgen

/pfad/zum/Webserver

```
| -TYPO3-Neos
| ---Build
| ---...
| ---Web (Document Root)
```

- Alle Neos-Dateien liegen in dem Verzeichnis TYPO3-Neos (dieses wird von „Composer“ in den nächsten Schritten angelegt)
- Es gibt keine Trennung zwischen Core und UserSpace (wie bei TYPO3 CMS)
- Das Document Root wird auf **/pfad/zum/Webserver/TYPO3-Neos/Web**

Installation von TYPO3 Neos - Composer

- Die Installation erfolgt über „Composer“
(Dependency Manager für PHP) - Dafür ist Zugang zur Konsole nötig

```
cd /pfad/zum/webserver/
```

```
curl -sS https://getcomposer.org/installer | php
```

- Dies legt die Datei **composer.phar** im aktuellen Verzeichnis an
- Will man den Composer zentral verwenden, kann man ihn auch verschieben/kopieren

```
mv composer.phar /usr/local/bin/composer
```

Installation von TYPO3 Neos - Composer

- Laden von TYPO3 Neos via Composer (in einer Zeile notieren!):

```
php /path/to/composer.phar create-project --no-dev
typo3/neos-base-distribution TYPO3-Neos
```

- Dies sorgt für die Installation von TYPO3 Flow, Neos und den benötigten Modulen (inkl. 3rd Party wie Doctrine 2, Aloha, ...)
- Anschließend erhält man ein Verzeichnis **TYPO3-Neos**, welches die letzte Version von Neos enthält
- Zur Installation von Composer unter Windows gibt es hier Infos:
<http://getcomposer.org/doc/00-intro.md#installation-windows>

[Alternative:] Archive verwenden

- Auf Sourceforge steht die aktuelle Neos Version zum Download als zip, tar.gz und tar.bz2 bereit:

<http://sourceforge.net/projects/typo3/files/TYPO3%20Neos/1.2.0/>

[Alternative:] GIT-Version von TYPO3 Neos verwenden

- Laden der aktuellsten GIT-Version von TYPO3 Neos:

```
git clone git://git.typo3.org/Neos/Distributions/  
Base.git TYPO3-Neos && cd TYPO3-Neos
```

Anschließend müssen dann noch die Abhängigkeiten geladen werden:

```
composer install -s beta
```

Installation von TYPO3 Neos - Rechte setzen

- Anschließend werden die Datei-Rechte in der Konsole gesetzt (falls nötig):

```
cd TYPO3-Neos
sudo ./flow flow:core:setfilepermissions shelluser wwwuser wwwgroup
```

(Weitere Infos: <http://docs.typo3.org/flow/TYPO3FlowDocumentation/TheDefinitiveGuide/PartII/Installation.html#file-permissions>)

- **shelluser**
Dies ist der User, mit dem man in der Konsole eingeloggt ist - kann mittels `whoami` herausgefunden werden
- **wwwuser**
Der User, unter dem der Webserver läuft (steht in der Datei `httpd.conf`) - unter Mac OS X z.B. `_www`
- **wwwgroup**
Die Gruppe, unter dem der Webserver läuft (steht in der Datei `httpd.conf`) - unter Mac OS X z.B. `www`

Installation von TYPO3 Neos - VirtualHost

- Virtual Host Eintrag (z.B. Apache)

```
NameVirtualHost *:80 # sofern benötigt

<VirtualHost *:80>

    DocumentRoot "/pfad/zum/webserver/TYPO3-Neos/Web/"

    # Während der Entwicklung sollte die folgende Zeile
    # auskommentiert werden, denn dies stellt den Context auf
    # „Production“ - dies bedeutet: kein Logging, mit Caching, ...
    # Default ist „Development“
    SetEnv FLOW_CONTEXT Production
    ServerName neos.demo

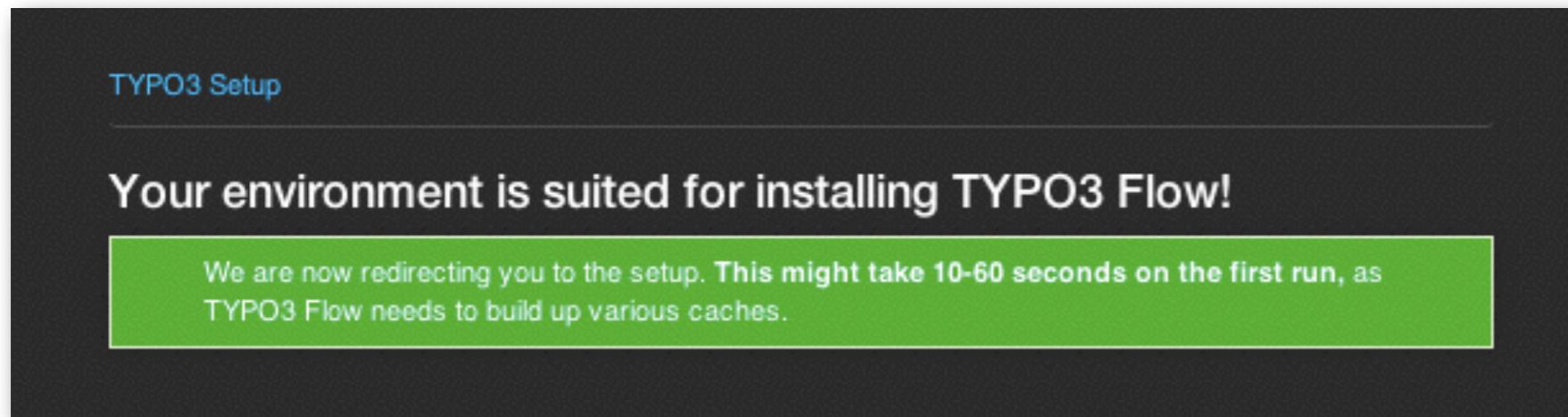
</VirtualHost>
```

- Eintrag in /etc/hosts (C:\windows\system32\drivers\etc\hosts)
`127.0.0.1 neos.demo`

Installation von TYPO3 Neos - Setup

- Aufruf der Setup-Routine durch

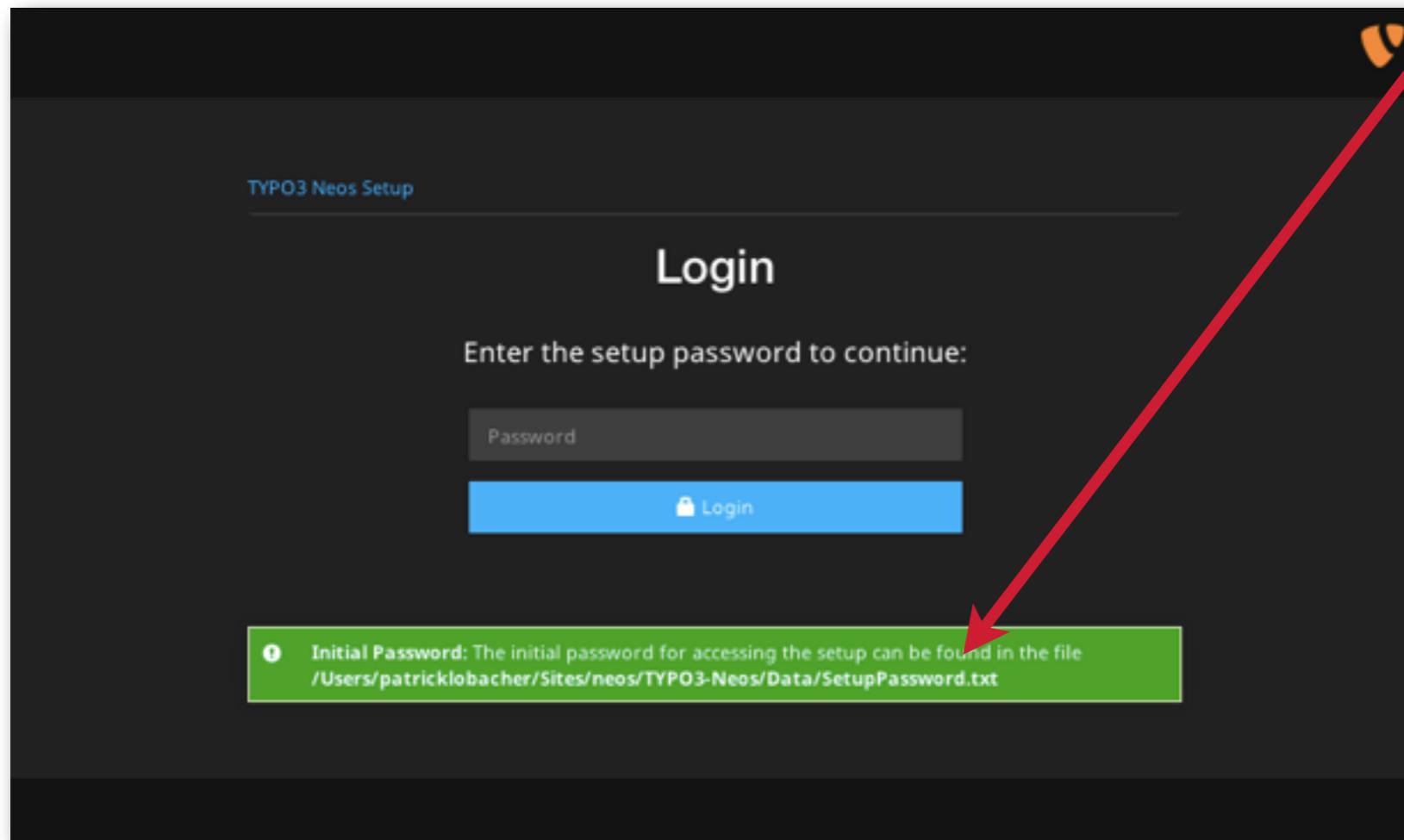
`http://neos.demo/setup/`



Installation von TYPO3 Neos - Setup

- Das Passwort befindet in der Datei im angezeigten Pfad (merken!)

`/pfad/zum/webserver/TYPO3-Neos/Data/SetupPassword.txt`



Die Datei mit dem Passwort wird anschließend wieder gelöscht. Hat man das Passwort vergessen, muss man die Datei

`/pfad/zum/webserver/TYPO3-Neos/Data/Persistent/FileBasedSimpleKeyService/SetupKey`

löschen und das Setup erneut aufrufen.

Installation von TYPO3 Neos - Setup

- Datenbank-Setup
- Voreinstellung ist MySQL
- Änderung des Treibers durch Editieren der Datei:
Configuration/Settings.yaml
- Sollte als DB-Host **127.0.0.1** nicht funktionieren, so kann man probieren stattdessen **localhost** dort einzutragen
- Man kann entweder eine bestehende DB auswählen oder eine neue anlegen

TYPO3 Neos Setup

Logout

Configure database

Please enter database details below:

Connection

DB Driver*

MySQL/MariaDB via PDO

DB Username*

root

DB Password

.....

DB Host*

127.0.0.1

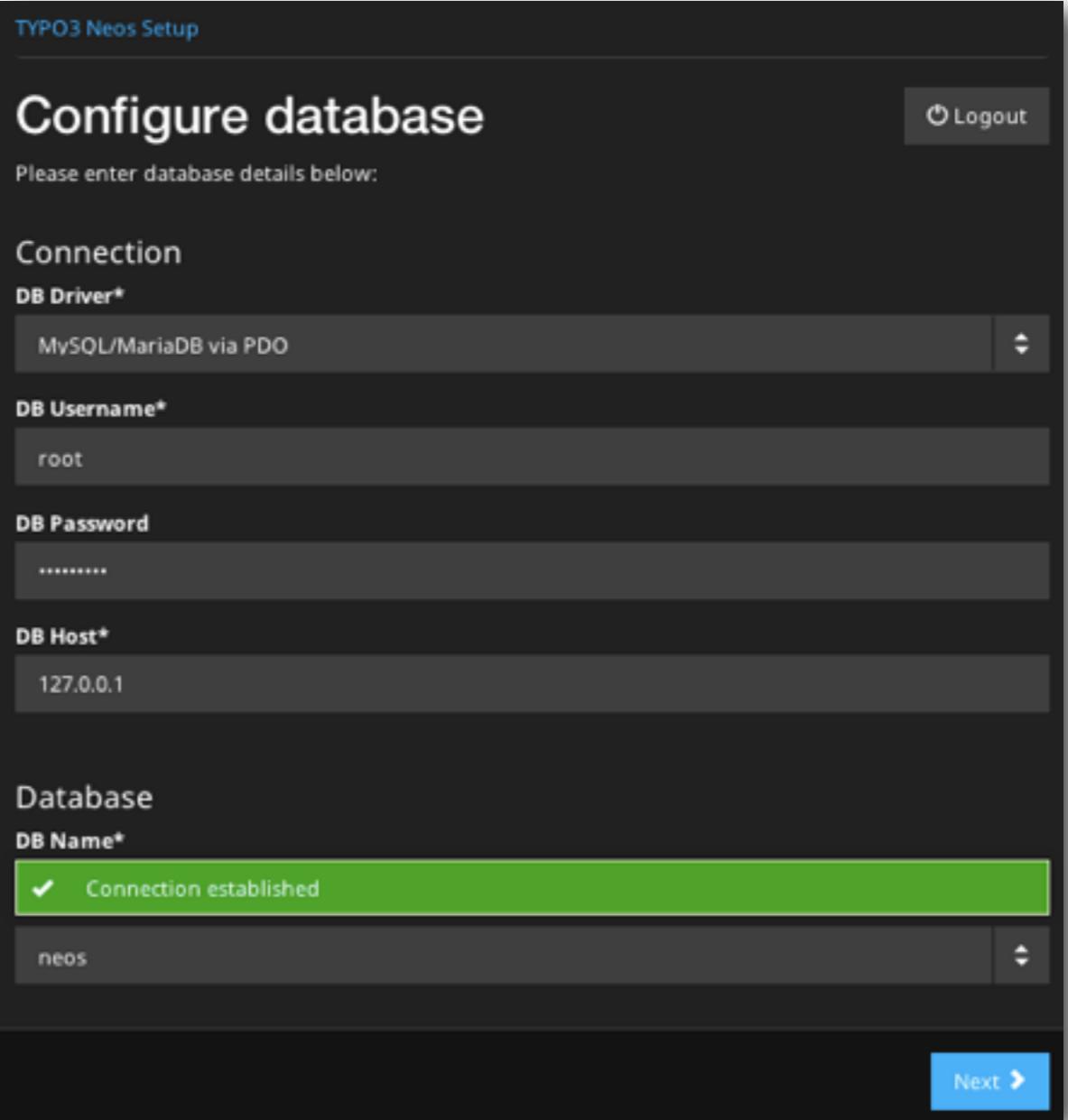
Database

DB Name*

✓ Connection established

neos

Next >



Installation von TYPO3 Neos - Setup

- Anlegen eines Administrators
- Weitere Benutzer können später in der Benutzerverwaltung angelegt werden
- Zusätzliche Benutzer-Daten können ebenfalls später in der Benutzerverwaltung zugefügt werden
- Manuell kann man einen Benutzer ebenfalls anlegen - in der Konsole:

TYPO3 Neos Setup

Create administrator account

Enter the personal data and credentials for your backend account:

Personal Data

First name*

Last name*

Credentials

Username*

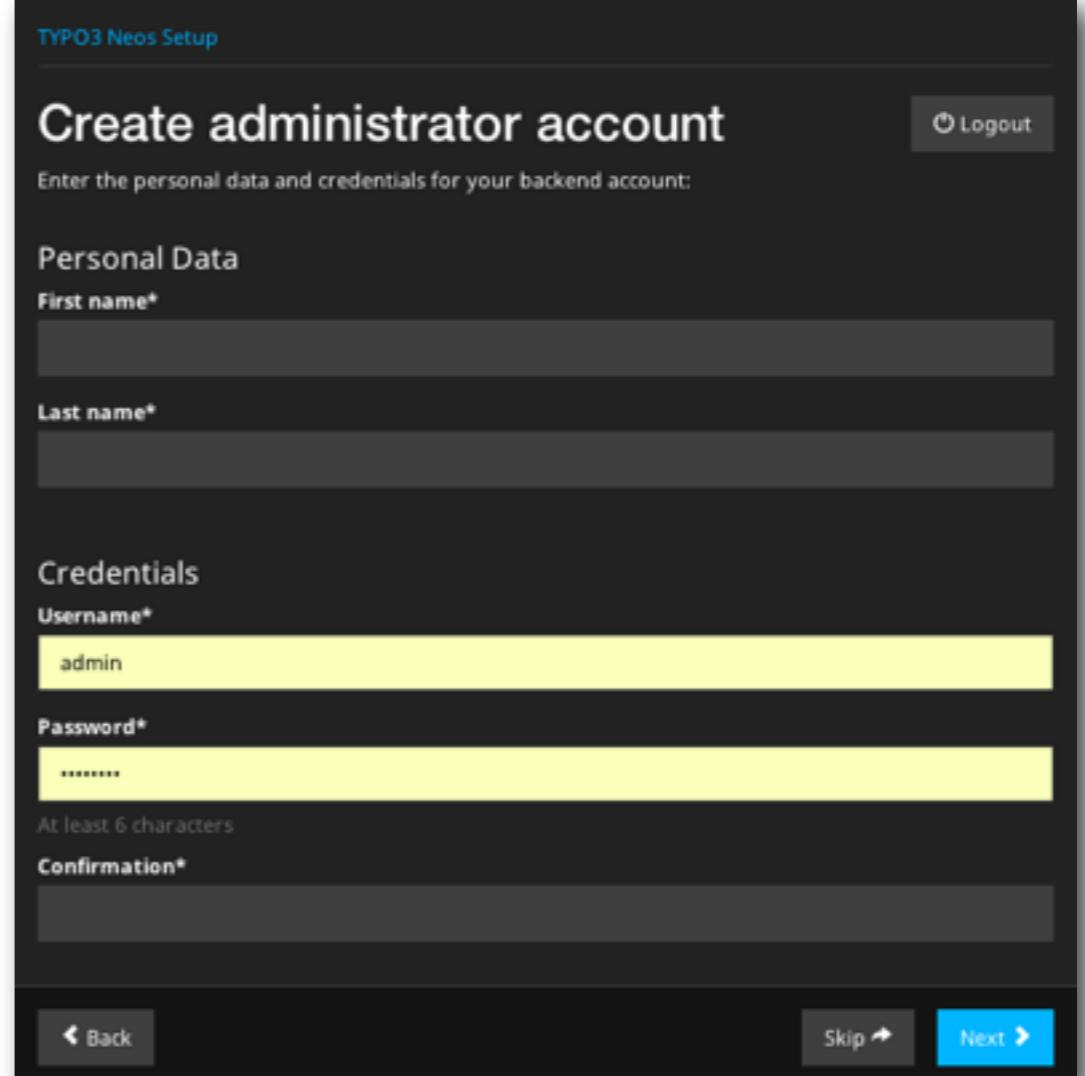
admin

Password*

At least 6 characters

Confirmation*

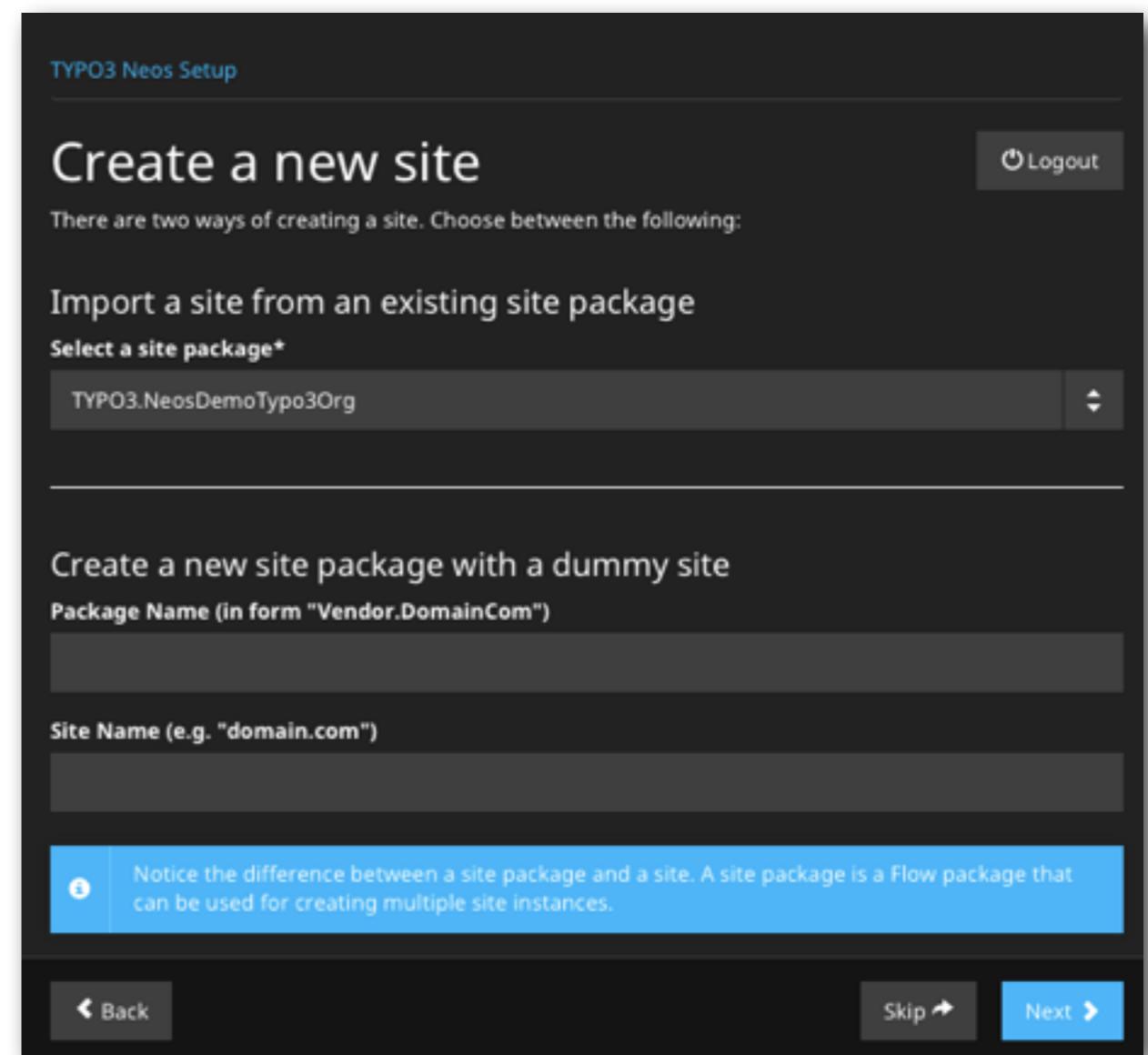
Back Skip Next



```
./flow typo3.neos:user:create username password firstname lastname
```

Installation von TYPO3 Neos - Setup

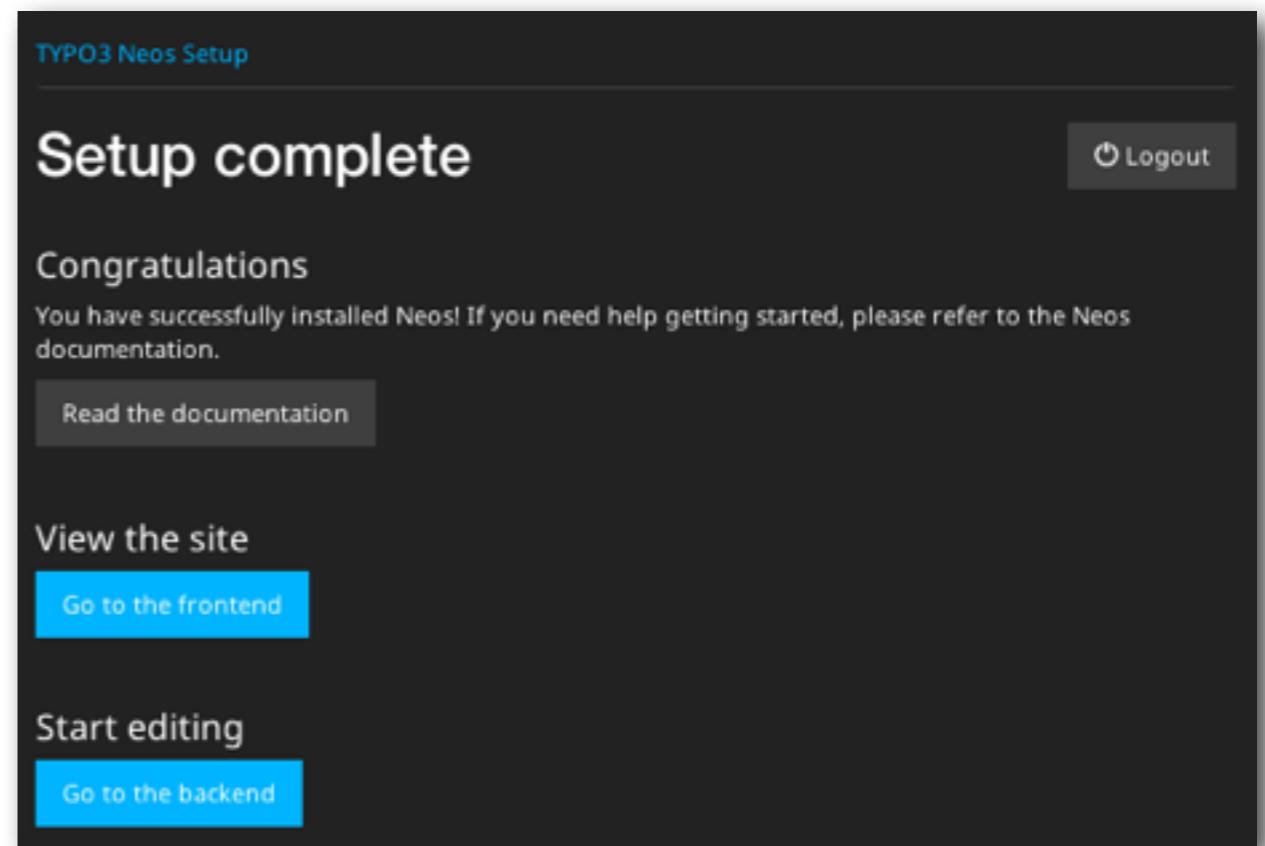
- Nun kann man entweder eine Demo-Site importieren (empfohlen)
- Oder mit einer leeren Website starten
- Sobald in diese beiden unteren Formularfelder etwas eingetragen ist, wird eine neue Site angelegt



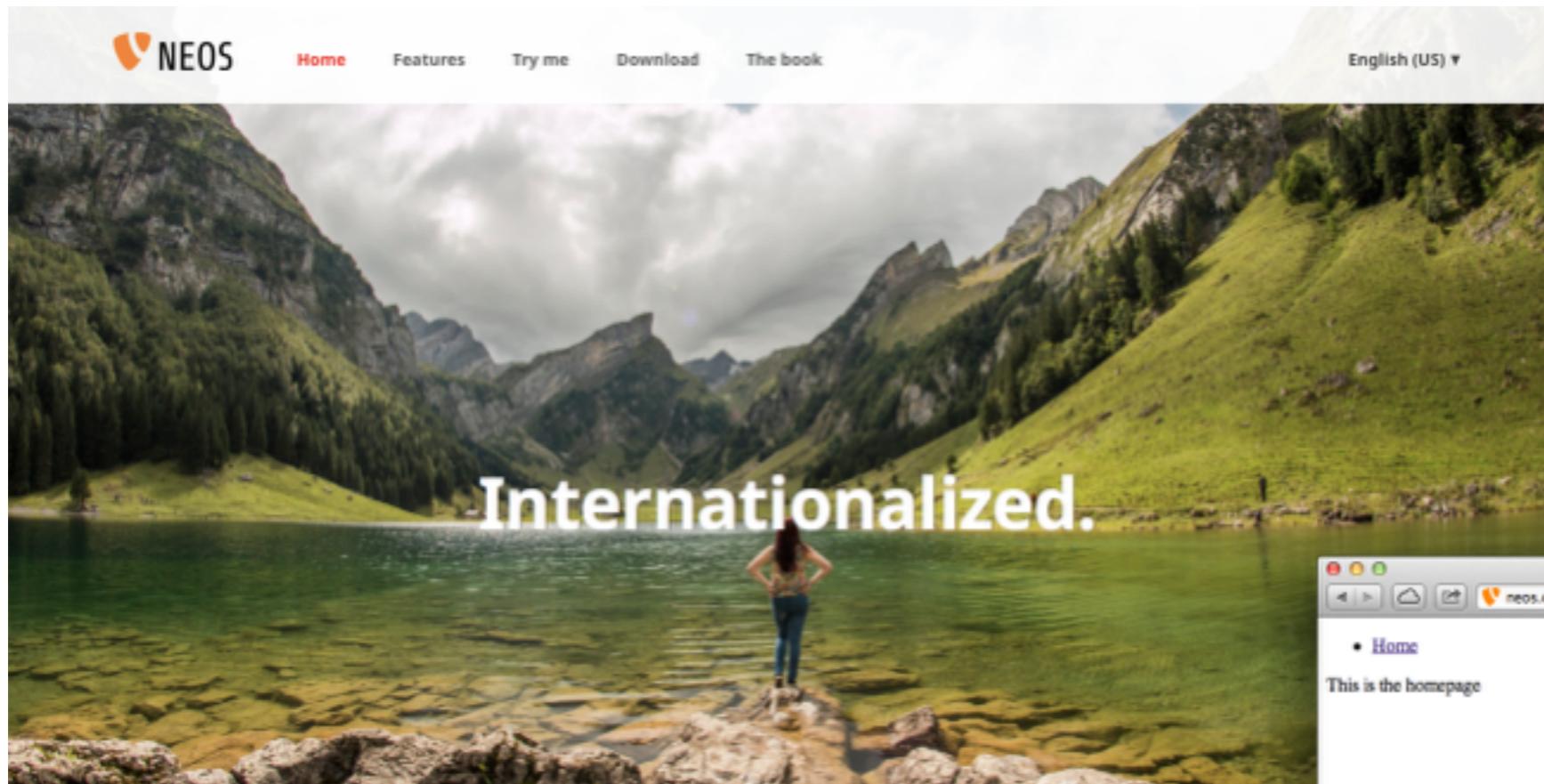
Installation von TYPO3 Neos - Setup

- Wenn die Installation erfolgreich durchgelaufen ist, erscheint der entsprechende Hinweis
- Die Dokumentation befindet sich unter

<http://docs.typo3.org/neos/>

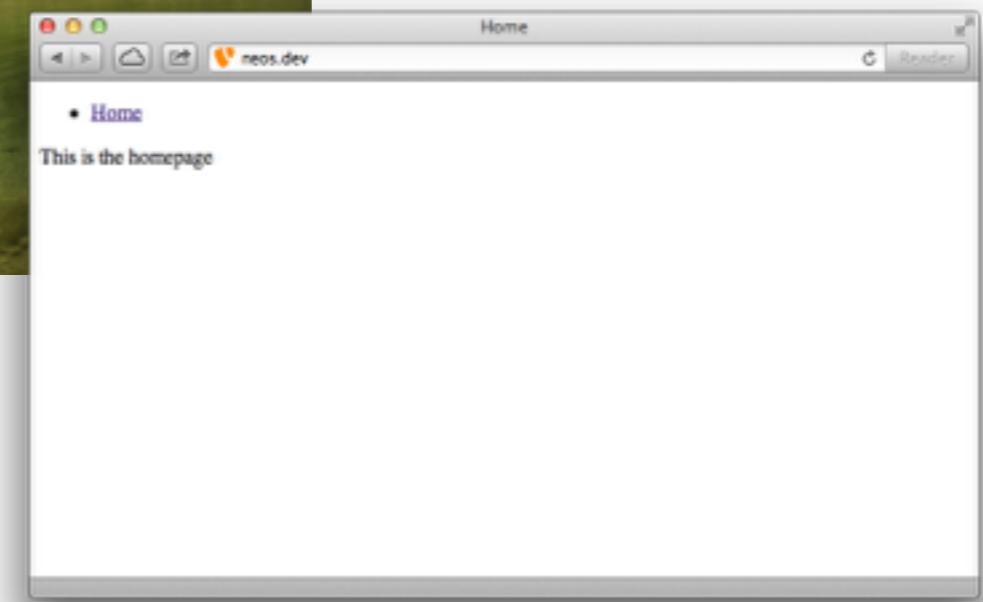


Installation von TYPO3 Neos - Setup



So wie oben sieht das Frontend aus, wenn man die Demo-Site **TYPO3.NeosDemoTypo3Org** importiert hat.

Wenn man eine leere Site angelegt hat, erhält man den folgenden Screen:



Installation von TYPO3 Neos - Setup

Zugang zur Installation:

- Frontend

`http://neos.demo/`

- Backend

`http://neos.demo/neos/`

- Setup

`http://neos.demo/setup/`

Installation auf einem **domainFACTORY Server**

Installation auf einem domainFACTORY Server

Auf domainFACTORY Server muss man einer leicht abweichenden Installationsanleitung folgen

- Credits gehen an:
 - **Christian Schwerdt** (die Medienagenten oHG)
 - **Timo Nußbaum** (Korrekturen)
- Die Anleitung findet sich auch hier:

<https://github.com/cschwerdt/TYPO3-NEOS-Install>

Installation auf einem domainFACTORY Server

- Composer laden

```
curl -s https://getcomposer.org/installer | /usr/local/bin/php5-53STABLE-CLI
```

- TYPO3 Neos 1.2.x laden:

```
/usr/local/bin/php5-53LATEST-CLI composer.phar create-project --no-dev typo3/  
neos-base-distribution TYPO3-Neos
```

- In den domainFACTORY-Settings muss nun die Domain so konfiguriert werden, dass sie auf **/Web** zeigt

Installation auf einem domainFACTORY Server

- PHP-Pfad im Flow-CLI ändern
(hier „vi“ - man kann aber natürlich auch jeden anderen Texteditor verwenden)

```
cd TYPO3-Neos
vi flow
```

Die erste Zeile muss nun von

```
#!/usr/bin/env php
```

in die folgende geändert werden:

```
#!/usr/local/bin/php5-53STABLE-CLI
```

Anschließend abspeichern mit

```
: !wq
```

Installation auf einem domainFACTORY Server

- Settings.yaml in Configuration/ anpassen (Editor „vi“ oder anderer)

```
cd Configuration/
cp Settings.yaml.example Settings.yaml
vi Settings.yaml

//set database host
db: 'mysql5.<yourdomain.com>'

//uncomment core: & phpBinaryPathAndFilename
//and change phpBinaryPathAndFilename to:
core:
  phpBinaryPathAndFilename: '/usr/local/bin/php5-53STABLE-CLI'
  subRequestPhpIniPathAndFilename': '/usr/local/bin/php5-53STABLE-CLI'
```

Nun Abspeichern via:

```
:wq!
```

Installation auf einem domainFACTORY Server

- Settings.yaml in Development/ anpassen (Editor „vi“ oder anderer)

```
cd Development/  
cp Settings.yaml.example Settings.yaml  
vi Settings.yaml
```

```
//set dbname, dbuser, dbpassword:  
dbname: '<dbname>'  
user: '<dbuser>'  
password: '<password>'
```

Nun Abspeichern via:

```
:wq!
```

Installation auf einem domainFACTORY Server

- Flow testen
`./flow help`
- Datenbank migrieren
`./flow doctrine:migrate`
- Site kickstarten

Syntax: `./flow kickstart:site [PackageKey] [SiteName]`
`./flow kickstart:site Your.Demopage Website`

- Sites auflisten
`./flow site:list`

Installation auf einem domainFACTORY Server

- Neos Backend User anlegen

```
./flow user:create <username> <password> <firstname> <lastname>
```

- Admin Userrolle zum User zufügen

```
./flow user:addrole <username> Administrator
```

- Fertig :-)

- ACHTUNG:

Die Standard-Collation-Type für neue Datenbanken sind per default nicht richtig gesetzt (collation_server=utf8_unicode_ci). Wenn man nun eigene Extensions verwendet, führt das zu einem doctrine:migrate Fehler.

Installation auf einem all-inkl.com Server

Installation auf einem all-inkl.com Server

Auf den Severn der „ALL-INKL.COM - Neue Medien Münnich“ muss man einer leicht abweichenden Installationsanleitung folgen

- Credits gehen an: **Mario Janetzko** (die Medienagenten oHG)
- Ab der Version TYPO3 Neos 1.1 muss lediglich in der .htaccess eingetragen werden, welche PHP-Version benutzt wird (AddHandler). Die restliche Anleitung wird *nicht mehr* benötigt!

Installation auf einem all-inkl.com Server

- Vor dem Installieren müssen folgende Schritte durchgeführt werden
- Öffnen der Datei `Bootstrap.php` im Pfad `/Packages/Framework/TYPO3.Flow/Classes/TYPO3/Flow/Core/`
- Einfügen der folgenden Zeile nach `namespace TYPO3\Flow\Core;`

```
<?php  
namespace TYPO3\Flow\Core;  
  
ini_set('date.timezone','Europe/Berlin');  
ini_set('memory_limit','256M');  
  
...
```

Installation auf einem all-inkl.com Server

- Auskommentieren der Zeilen 557 - 560 in der Datei Bootstrap.php

ALT:

```
if (version_compare(PHP_VERSION, '5.4', '<') && get_magic_quotes_gpc() === 1) {  
    echo('Flow requires the PHP setting "magic_quotes_gpc" set to Off. (Error  
#1224003190)');  
    exit(1);  
}
```

NEU:

```
/* if (version_compare(PHP_VERSION, '5.4', '<') && get_magic_quotes_gpc() === 1) {  
    echo('Flow requires the PHP setting "magic_quotes_gpc" set to Off. (Error  
#1224003190)');  
    exit(1);  
} */
```

Installation auf einem all-inkl.com Server

- Hinzufügen der folgenden Zeilen in die Datei Web/index.php ab Zeile 23:

```
...
require($rootPath . 'Packages/Framework/TYPO3.Flow/Classes/TYPO3/Flow/Core/
Bootstrap.php');

putenv ("FLOW_REWRITEURLS=1");
putenv ("FLOW_CONTEXT=Production");
if (substr($_SERVER['HTTP_HOST'],0,4) === 'dev.') {
    putenv ("FLOW_CONTEXT=Development");
}

$context = getenv('FLOW_CONTEXT') ?: (getenv('REDIRECT_FLOW_CONTEXT') ?:
'Development');
...
```

- Nun kann die Installation gestartet werden!

Installation auf einem Uberspace Server

Installation auf einem Uberspace Server

Auf den Uberspace-Servern (<http://www.uberspace.de/>) muss man einer leicht abweichenden Installationsanleitung folgen

- Credits gehen an: **Kerstin Huppenbauer** (digiparden GmbH)
- Das Original befindet sich unter der URL:
<http://www.flowrocks.de/posts/2014/06/18/typo3-neos-in-einem-uberspace-installieren..html>
- **(NAME** durch den eigenen Uberspace-Namen und **DOMAIN** durch die eigene Domain ersetzen)

Domains

Domains müssen extern registriert werden und lassen sich über

uberspace-add-domain -d DOMAIN -w

als Aliasdomain der Webserver Konfiguration hinzufügen.

Die lange Version findet sich hier:

<https://uberspace.de/dokuwiki/domain:verwalten>

Composer

Composer installieren und ins **bin** Verzeichnis verschieben:

```
curl -sS https://getcomposer.org/installer | php
mv composer.phar /home/NAME/bin/composer
```

PHP

Als nächstes die aktive PHP-Version überprüfen. Da Flow auch PHP-CLI nutzt, muss die Version identisch sein. In meinem Fall war unter `/package/host/localhost/` der PHP-Interpreter mit `php-5.5.10` verlinkt - `which php` lieferte aber `/package/host/localhost/php-5.5.5/bin/php` zurück. Dies lässt sich einfach in der Datei `phpversion` ändern: `cat ~/etc/phpversion` und dort die passende PHP-Version eintragen, mit `killall php-cgi` den PHP-Interpreter neu starten und aus der Shell aus und wieder einloggen.

PHP Module installieren

Um eigene PEAR oder PECL Module zu installieren, muss zunächst eine Konfigurationsdatei angelegt werden:

```
pear config-create ~ ~/.pearrc
```

Anschließend kann z.B. Imagick über

```
uberspace-install-pecl imagick
```

installiert werden.

Ausführliche Informationen finden sich hier - auch wie eine eigene `php.ini` angelegt werden kann:

<https://uberspace.de/dokuwiki/development:php>

Neos installieren

Nach den Vorbereitungen kann nun Neos installiert werden.

Der Document-Root bei Uberspace ist /home/NAME/html, was ein Symlink auf /var/www/virtual/NAME/html ist. Da Neos das Web-Verzeichnis als Web-Root nutzt, sollte Neos also nicht im Document-Root sondern irgendwo daneben installiert werden:

```
cd /var/www/virtual/NAME
git clone https://git.typo3.org/Neos/Distributions/Base.git
cd Base
composer update
```

Um nun unsere vorhin hinzugefügte Domain mit der Neos Installation zu verbinden, legen wir im Root unseres Uberspaces einfach einen Symlink an. Ohne eigene Domain würde der Symlink aus dem Document Root auf das Web-Verzeichnis zeigen.

```
cd /var/www/virtual/NAME
ln -s /var/www/virtual/NAME/Base/Web DOMAIN
```

Neos installieren

Der Symlink muss dabei dem Hostnamen entsprechen. Soll die Seite über weitere Hostnamen erreichbar sein (z.B. für Developement und Production Context), können einfach weitere Symlinks mit den Hostnamen angelegt werden. Da Development der Default Context ist, kann der Production Context in der `.htaccess` wie folgt gesetzt werden

```
SetEnvIf Host "^\w{3,4}\.\w{2,4}\.\w{2,4}\.\w{2,4}" FLOW_CONTEXT=Production
```

MySQL

Das MySQL Passwort wurde von Uberspace gesetzt und befindet sich in der .my.cnf

cat .my.cnf

Passwort kopieren, Neos Backend aufrufen und der Spaß kann beginnen :-)

Deployment mit TYPO3 Surf auf einem Uberspace Server

Deployment mit TYPO3 Surf auf einem Uberspace Server

Hier eine Anleitung, um TYPO3 Neos mit Hilfe von TYPO3 Surf auf einem Uberspace-Servern (<http://www.uberspace.de/>) zu deployen.

- Credits gehen an: **Benjamin Albrecht** (@beelbrecht, beelbrecht.de)
- Das Original befindet sich unter der URL:
<https://gist.github.com/beelbrecht/fa8baf9f22faca64711e>
- Infos zu Surf gibt es hier:
<http://de.slideshare.net/kfish/deploying-typo3-neos-websites-using-surf>
<http://etobi.de/blog/2012/10/deployment-write-a-deployment-recipe-for-typo3-surf/>
<http://forge.typo3.org/projects/package-typo3-surf/repository/revisions/master/entry/Documentation/Guide/Index.rst>

Deployment mit TYPO3 Surf auf einem Uberspace Server 1/4

```
<?php

// TYPO3 Surf script to deploy and update TYPO3 Neos at your uberspace.de account
// before or after the initial deployment, you have to setup some things manually at your host //
// (e.g. DB credentials in Settings.yaml)

// Note: replace placeholders such as [PLACEHOLDER] with the correct information

// Create a simple workflow based on the predefined 'SimpleWorkflow'.
$workflow = new \TYPO3\Surf\Domain\Model\SimpleWorkflow();

// Create and configure a simple shell task to add the FLOW_CONTEXT to your .htaccess file
// If you're using NEOS at your local machine to develop a NEOS site, you won't add this file and
// this setting to your Git repo...
$editHtaccessCommandOptions = array(
    'command' => 'echo -e "\nSetEnv FLOW_CONTEXT Production \n" >> {releasePath}/Web/.htaccess'
);

$workflow->defineTask('foobar:editHtaccess', 'typo3.surf:shell', $editHtaccessCommandOptions);

$workflow->addTask('foobar:editHtaccess', 'finalize');
```

Deployment mit TYPO3 Surf auf einem Uberspace Server 2/4

```
// Add the workflow to the deployment. The $deployment instance is created by Surf.  
$deployment->setWorkflow($workflow);  
  
// Create and configure your node / nodes (host / hosts).  
$node = new \TYPO3\Surf\Domain\Model\Node('uberspace');  
$node->setHostname('[username].[host].uberspace.de');  
  
// If you don't use SSH-Keys to log into your remote host via SSH (as recommended),  
// you have to add the following line:  
// $node->setOption('password', '[PASSWORD]');  
// But you don't want to have any passwords as string in your deployment script ;-)  
$node->setOption('username', '[username]');  
  
// Define your application and add it to your node.  
$application = new \TYPO3\Surf\Application\TYPO3\Flow('['[sitename]');  
  
// The deployment path is not the document root!  
// The document root has to be: '[deploymentPath]/releases/current/Web'.  
// At uberspace: create a symlink from '/var/www/virtual/[user]/html'  
// to '[deploymentPath]/release/current/Web'  
$application->setDeploymentPath('/var/www/virtual/[username]/[sitename]');
```

Deployment mit TYPO3 Surf auf einem Uberspace Server 3/4

```
// Be sure, that your node has read-access to your git repository.  
// In most cases, you can use the git- or https-protocol for public repositories or  
// SSH to read from private repositories.  
$application->setOption('repositoryUrl', '[urlToYourNeosDistributionGitRepository]');  
  
// Be sure, that you have installed composer  
$application->setOption('composerCommandPath', '/home/[username]/bin/composer');  
  
$application->setOption('keepReleases', '5');  
  
$application->addNode($node);  
  
// remove unused task.  
// you can't run a command with "sudo" at shared hosting environments  
$deployment->onInitialize(function() use ($workflow, $application) {  
    $workflow->removeTask('typo3.surf:typo3:flow:setfilepermissions');  
});  
  
// Add the application to your deployment.  
$deployment->addApplication($application);
```

Deployment mit TYPO3 Surf auf einem Uberspace Server 4/4

Um das Deployment Script zu verwenden, muss TYPO3 Surf zunächst installiert werden:

```
./flow package:import TYPO3.Surf
```

Nun legt man das Deployment-Script im folgenden Verzeichnis ab:

```
%FLOW_ROOT%/Build/Surf/NeosDeployment.php
```

Infos zu dem Deployment gibt es hiermit:

```
./flow surf:describe NeosDeployment
```

Simulieren kann man das Deployment hiermit:

```
./flow surf:simulate NeosDeployment
```

Hiermit wird das Deployment durchgeführt:

```
$ ./flow surf:deploy MyDeployment
```

Troubleshoot

FAQ / Hilfe

Installations-Alternativen und Hilfen

- TYPO3 Neos Vagrant Box
<https://github.com/tlayh/vagrant-typo3neos>
- TYPO3 Neos Installation auf Mac OS X Snow Leopard und MAMP
<http://www.content-driven-e-commerce.de/typo3-neos-install/>
- .git ignore File
[https://git.typo3.org/Neos/Distributions/Base.git/blob/
HEAD:.gitignore](https://git.typo3.org/Neos/Distributions/Base.git/blob/HEAD:.gitignore)
- Launchr (TYPO3 Neos Online testen)
<https://launchr.com/>
- Mittwald Hosting - 30 Tage TYPO3 Neos Account
<https://www.mittwald.de/neos-testen/>
- Manuelle Installation von jweiland.net
[http://jweiland.net/typo3/neos/neos-installation/manuelle-neos-
installation.html](http://jweiland.net/typo3/neos/neos-installation/manuelle-neos-installation.html)
- Installation in einer Shared Hosting Umgebung (Hostsharing eG)
https://wiki.hostsharing.net/index.php?title=TYPO3_Neos_installieren

Installationsprobleme

- Im PHP-Log oder auf der Konsole bei Aufruf von ./flow help oder im Frontend kommt eine Fehlermeldung, die ähnlich der folgenden ist:

- ...'"typo3eel" is not a valid package key...

Abhilfe: Verwendung einer fehlerkorrigierten Composer-Datei:

<http://downloads.sourceforge.net/project/typo3flow/patched-1147-composer.phar>

- Es gibt eine Fehlermeldung die ähnlich der folgenden ist:

- ...Class 'PHPUnit_Framework_Constraint' not found in...

oder

- ...imagine... (irgendwas weißt auf imagine hin)

Abhilfe: Löschen der PackageStates-Datei

[rm Configuration/PackageStates.php](#)

Nginx Config für TYPO3 Neos

Nginx Config für TYPO3 Neos

- Credits gehen an:
 - **Christian Kuhn**
 - **Christian Müller**
 - **Anja Leichsenring**

Nginx Config für TYPO3 Neos

- Datei **/etc/nginx/sites-available/neos.demo** (Domain sei neos.demo)

```
# Upstream to abstract backend connection(s) for php
upstream neos.demo {
    server unix:/var/run/neos.demo_fpm.sock;
}
server {
    server_name neos.demo;
    root /pfad/zum/webserver/TYPO3-Neos/Web;
    index index.php;
    error_log /pfad/zum/webserver/TYPO3-Neos/logs/error_log;
    access_log /pfad/zum/webserver/TYPO3-Neos/logs/access_log;
    ## Disable .htaccess and other hidden files
    location ~ /\. {
        deny all;
        access_log off;
        log_not_found off;
    }
    location = /favicon.ico {
        log_not_found off;
        access_log off;
    }
    location = /robots.txt {
        allow all;
        log_not_found off;
        access_log off;
    }
}
```

Nginx Config für TYPO3 Neos

- Datei **/etc/nginx/sites-available/neos.demo** (...FORTSETZUNG...)

```
location /_Resources/ {  
    access_log off;  
    log_not_found off;  
    expires max;  
    break;  
}  
location /_Resources/Persistent/ {  
    access_log off;  
    log_not_found off;  
    expires max;  
    rewrite "(.{40})/.+\.(.+)" /_Resources/Persistent/$1.$2 break;  
    rewrite "([a-z0-9]+/(.+)?[a-f0-9]{40})/.+\.(.+)" /_Resources/Persistent/$1.$2 break;  
}  
###  
# stop rewriting by existing files | is instead of -> location / { rewrite ".+" /index.php last; }  
# use this if you want to run other PHP-Applications in TYPO3-Flow/Web directory  
###  
try_files $uri $uri/ /index.php?$args;
```

Nginx Config für TYPO3 Neos

- Datei **/etc/nginx/sites-available/neos.demo** (...FORTSETZUNG...Fertig!)

```
location ~ \.php$ {  
  
    fastcgi_index index.php;  
    ###  
    # for FLOW3 <= 1.1.x only | see note #15 on http://forge.typo3.org/issues/8923  
    ###  
    # fastcgi_param    FLOW3_CONTEXT          Development;  
    # fastcgi_param    FLOW3_CONTEXT          Production;  
    # fastcgi_param    FLOW3_REWRITEURLS     1;  
    ###  
    # Make sure that you set the environment vars for new versions \  
    # of TYPO3-XXXXXX(TYPO3-Neos) products properly  
    # see note #15 on http://forge.typo3.org/issues/8923  
    ###  
    fastcgi_param    FLOW_CONTEXT          Development;  
    fastcgi_param    FLOW_REWRITEURLS     1;  
    fastcgi_split_path_info ^(.+\.php)(.*)$;  
    fastcgi_param    SCRIPT_FILENAME      $document_root$fastcgi_script_name;  
    fastcgi_param    PATH_INFO           $fastcgi_path_info;  
    fastcgi_pass    neos.domain.tld;  
    include        /etc/nginx/fastcgi_params;  
}  
}
```

Nginx Config für TYPO3 Neos

- Datei **/etc/php5/fpm/pool.d/neos.demo.pool.conf**

```
[neos.demo]
listen = /var/run/neos.demo.sock
listen.owner = neos.demo
listen.group = demo
listen.mode = 0660
user = neos.demo
group = demo
pm = dynamic
pm.max_children = 50
pm.start_servers = 5
pm.min_spare_servers = 3
pm.max_spare_servers = 10
pm.max_requests = 200
request_terminate_timeout = 360s
chdir =
php_admin_value[session.save_path] = "/pfad/zum/webserver/TYPO3-Neos/sessions"
# Wahrend der Installation kann das nicht ausreichen, ich hab danach wieder aufgeraeumt.
# Neos sagt das aber, dann eintragen, php-fpm neu starten, weitermachen. Zum Betrieb reicht das hier aus.
php_admin_value[open_basedir] = "/pfad/zum/webserver/TYPO3-Neos/:/usr/share/pear:/usr/share/php:/tmp:/usr/local/lib/php:/usr/bin/php"
```

Nginx Config für TYPO3 Neos

- Datei **Configuration/Settings.yaml**

```
TYPO3:  
  Flow:  
    persistence:  
      backendOptions:  
        dbname: ...  
        user: ...  
        password: ...  
        host: 127.0.0.1  
    core:  
      subRequestPhpIniPathAndFilename: /etc/php5/cli/php.ini
```

Upgrade von TYPO3 Neos

Upgrade von TYPO3 Neos

Um eine existierende Installation auf das letzte Patch-Level zu heben, kann man wie folgt vorgehen:

```
cd /pfad/zum/webserver/  
composer update --no-dev "typo3/*"  
# Cache loeschen!  
./flow flow:cache:flush --force  
./flow doctrine:migrate
```

Gegebenenfalls muss man **php composer.phar** verwenden!

Switching zum Development Master

Um die letzten Änderungen zu teste, kann man auf den Development Master umschalten:

```
cd /pfad/zum/webserver/
composer require --no-update "typo3/neos:dev-master"
composer require --no-update "typo3/neos-nodetypes:dev-master"
composer require --no-update "typo3/neosdemotypo3org:dev-master"
composer require --no-update "typo3/neos-kickstarter:dev-master"
composer require --no-update "typo3/buildessentials:dev-master"
composer update
# Cache loeschen!
./flow flow:cache:flush --force
./flow doctrine:migrate
```

Gegebenenfalls muss man **php composer.phar** verwenden!

Virtuelle Maschinen

Virtuelle Maschinen

Es gibt einige Projekte, die virtuelle Maschinen für TYPO3 Neos (und Flow) anbieten:

- **Vagrant-Umgebung** (TYPO3 muss darin selbst installiert werden)
<https://github.com/swiftlizard/VagrantTYPO3>
- **Vagrant-Umgebung** (TYPO3 NEOS muss darin selbst installiert werden)
<https://github.com/mrimann/VagrantTYPO3Flow>
- **Komplett funktionsfähiges TYPO3 Neos**
<https://launchr.com/typo3-neos>

Vor-Konfigurierte Neos Server bei Amazon Web Services (AWS)



AWS - Features



- **keine Installation notwendig (alle Komponenten sind vor-installiert)**
- **Betriebssystem: Debian Linux (64 Bit)**
- **Server Location auswählbar (zur Zeit: USA, Irland, Deutschland, Singapur, Japan, Australien und China)**
- **uneingeschränkter "root" Zugang zum Server via SSH**
- **uneingeschränkter "Administrator" Zugang zu Neos**
- **Server ist einfach skalierbar (CPU, Speicher, Disks, etc.)**
- **keine Mindestvertragslaufzeit ("pay as you go": berechnet werden nur aktive Server auf stundenbasis)**
- **eignet sich besonders als Grundlage für eine neue Neos Website oder um Neos zu testen oder auszuprobieren**

AWS - Voraussetzungen



- ein Account bei Amazon Web Services (AWS)
- grundlegende Kenntnisse in der System Administration (Linux)
- ca. 2 bis 3 Minuten Zeit, bis der Server in der Cloud hochgefahren ist :-)

Danke an Michael Schams (<http://schams.net/>)
für die Anleitung! :-)

AWS - Vorgehen zum Starten eines Neos Servers



- Schritt 1: im AWS Marketplace das Neos Machine Image finden:

[https://aws.amazon.com/marketplace/seller-profile/?
id=3c5e5f3c-d60e-4405-a9ca-aae8abfa3e2b](https://aws.amazon.com/marketplace/seller-profile/?id=3c5e5f3c-d60e-4405-a9ca-aae8abfa3e2b)

- Schritt 2: Server auswählen, konfigurieren (Geo-Location, Serverleistung, Firewall, SSH Schlüssel, etc.) und starten
- Schritt 3: mittels SSH auf den Server verbinden, um automatisch generierte Passwörter zu ermitteln

AWS - Sonstige Informationen



- Die Nutzung der "Software" ist kostenlos - es fallen lediglich die Gebühren für die AWS Infrastruktur an (beginnend bei US\$0.013 pro Stunde), siehe <http://aws.amazon.com/ec2/pricing/>
- AWS bieten einen "Free Tier", womit unter Umständen ein einfacher Neos Server (t2.micro Instanz) für 12 Monate kostenlos genutzt werden kann (allerdings nur bedingt für den Produktiveinsatz geeignet), siehe: <http://aws.amazon.com/free>
- Credits gehen an: Michael Schams (schams.net), der bereits für die TYPO3 CMS Machine Images im AWS Marketplace verantwortlich ist.
- Ausführliche Dokumentation und weitere Informationen unter <http://schams.net/typo3-on-aws>

Release Notes

Release Notes TYPO3 Neos 1.0.1 (13.12.2013)

- [TASK] Update references in documentation
- [BUGFIX] Handle inline loading of pages without metadata correctly
- [BUGFIX] Use FQ TypoScript paths for lastVisitedNode functionality
- [BUGFIX] Add a safeguard to the LIBXML_PARSEHUGE constant
- [BUGFIX] Fix wrong type annotation that causes compilation problems
- [BUGFIX] Fix built-in Menu TypoScript object template
- [BUGFIX] Prototypes don't use FQN
- [BUGFIX] Correctly link the Neos logo with the NodeViewHelper
- https://git.typo3.org/Packages/TYPO3.Neos.git/blob_plain/e87653620afa44112fc8b32414c9d90d464b3b00:/Documentation/Appendices/ChangeLogs/101.rst

Release Notes TYPO3 Neos 1.0.2 (04.03.2014) - Teil 1

TYPO3.Neos

- [TASK] Update references in documentation
- [BUGFIX] Secondary inspector is rendered outside viewport in FF
- [BUGFIX] A better way of parsing the HTML of asynchronous page loads
- [TASK] Optimize typical filter usages of EEL children operation
- [BUGFIX] Default attributes property of menu TypoScript object
- [BUGFIX] "node" used in label for create new dialog
- [BUGFIX] Security policies in Neos are too strict
- [BUGFIX] Link inceptor handling of local links
- [BUGFIX] Node tree filter too wide in Firefox
- [BUGFIX] Sites management module widget action button
- [TASK] Improve usability of position menus in navigate component
- [BUGFIX] Title missing for paste/new buttons in navigate component
- [BUGFIX] Shortcut to siteroot has no active state
- [TASK] Fix minor errors in documentation
- [!!!!] [BUGFIX] Only accepts URLs with the configured URI suffix
- [BUGFIX] YAML indentation and typo in integrators cookbook
- [TASK] Fix various CGL violations
- [BUGFIX] Opacity of datetime editor inspector field
- [TASK] Add missing grunt-trimtrailingspaces dependency

Release Notes TYPO3 Neos 1.0.2 (04.03.2014) - Teil 2

- [FEATURE] Reworked MenuImplementation
- [TASK] Minor coding fixes for users management module
- [BUGFIX] Remove obsolete route that might break routing
- [TASK] Fix Page documentation in TS reference
- [BUGFIX] Correctly rename site root nodes
- [BUGFIX] Menu section collapse arrow styling
- [BUGFIX] Headline alignment is not persisted
- [BUGFIX] Shortcut rendering is broken in combination with layout
- [BUGFIX] Dateselector should do "previous" too

TYPO3.TYPO3CR

- [BUGFIX] Node references do not respect context workspace
- [BUGFIX] Prevent database error with too long index
- [BUGFIX] materialize NodeData in removeProperty()
- [TASK] Fix wrong hint in method docblock

TYPO3.TypoScript

- [BUGFIX] isEven & isOdd not supported in TYPO3.Neos:ContentCollection

Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 1

TYPO3.Neos

- [TASK] Update references in documentation
- [TASK] Add ContentCache documentation to the documentation index
- [TASK] Add Content Cache documentation
- [BUGFIX] Content element wrapping should fail gracefully
- [!!!!] [BUGFIX] Menu state should not be calculated based on a shortcut
- [BUGFIX] Exception on deleting used Assets
- [BUGFIX] UserPreferenceController indexAction missing template
- [BUGFIX] The site import / export does not handle properties of type array
- [TASK] add alignment example for Aloha
- [BUGFIX] Alignment configuration for Aloha editor broken
- [BUGFIX] Site import service duplicates image resources
- [BUGFIX] Content cache should be cleared when discarding changes
- [BUGFIX] Use NodeNameGenerator to ensure unique node names
- [BUGFIX] Hide formatting button when no options are available
- [BUGFIX] Page tree reloads on every page change
- [BUGFIX] Title used in publishing notice relies on page reload
- [TASK] Add generated Eel helper documentation

Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 2

TYPO3.Neos

- [BUGFIX] ConvertUris throws exception with NULL values
- [BUGFIX] Find FlowQuery operation returns array with NULL
- [BUGFIX] Filter operation instanceof only works with node interface
- [BUGFIX] Instanceof Fizzle operator only works with nodes
- [BUGFIX] Asset editor calls server without identifier
- [BUGFIX] Children operation optimization bypasses filters
- [BUGFIX] Inline editable properties re-initialized after publishing
- [BUGFIX] Add correct changelog for TYPO3 Neos 1.1.0-beta3

TYPO3.Neos.NodeTypes

- [TASK] Work around Image serialize bug

TYPO3.Neos.Kickstarter

- [TASK] Update page TypoScript to match best practice

Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 3

TYPO3.TYPO3CR

- [BUGFIX] createVariantForContext() fails if workspace differs from source
- [TASK] Method to find entity relations in Node properties
- [TASK] Method to find if a given path exists anywhere in the CR
- [BUGFIX] Fix three risky unit tests
- [BUGFIX] Copy into the correct reference node

TYPO3.TypoScript

- [BUGFIX] TypoScriptRuntime should not intercept Login redirect

TYPO3.TypoScript

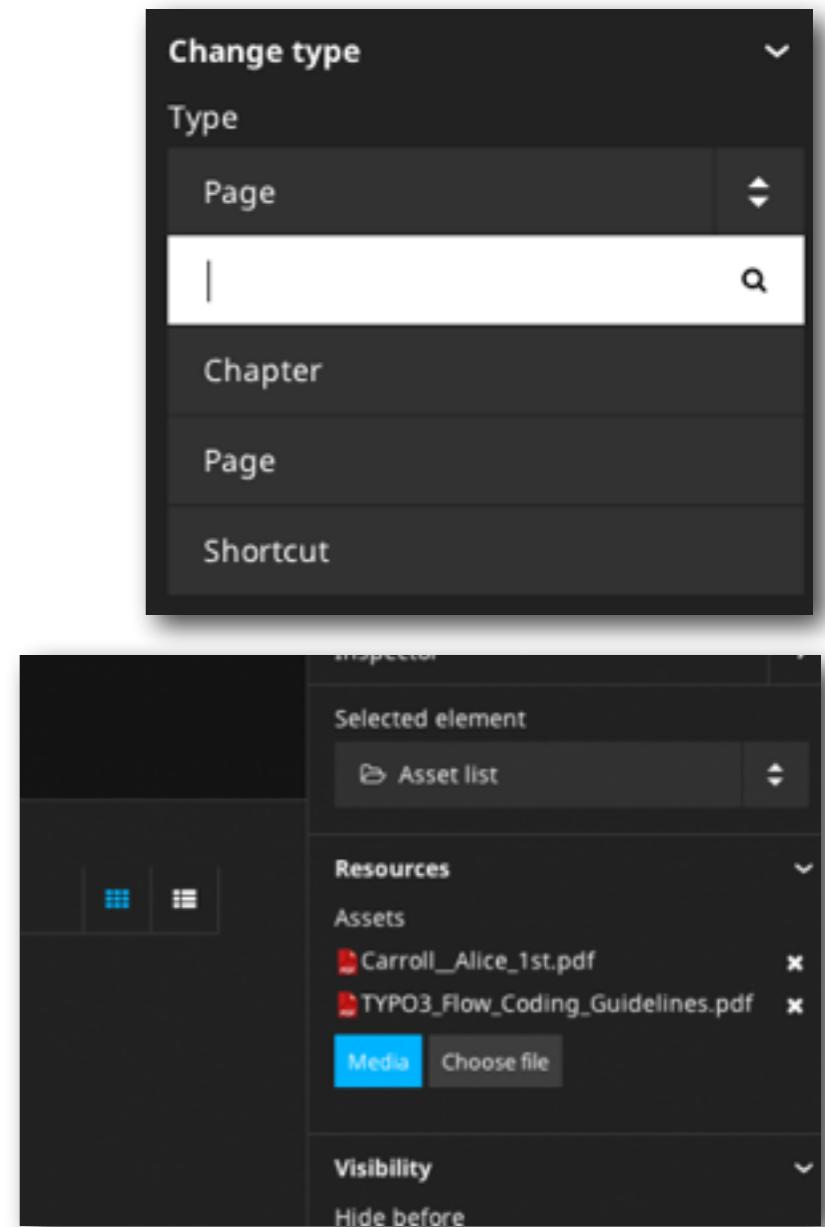
- [TASK] Add the homepage to the main menu
- [TASK] Add AssetList example element
- [TASK] Update exported site (fixes duplicate images)
- [BUGFIX] Use node full label instead of title for chapter pager
- [TASK] Use new BreadcrumbMenu TS object instead of custom template

Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 4

- Der **Content Cache** ist ein Layer innerhalb des Neos View-Parts, welcher einen Cache für gerenderten Content zur Verfügung stellt. Dieser kann per TypoScript konfiguriert werden.
- **Performance Boost:** Vor allem im Production Kontext Mode ist Neos nun deutlich schneller geworden.
- **Node Tree:** Wenn eine neue Document Node erstellt wird, springt das Content Modul direkt an diese Stelle. Sobald man Nodes kopiert oder bewegt, wird die „nach“ der Position per Default durchgeführt (und nicht mehr „innerhalb“).

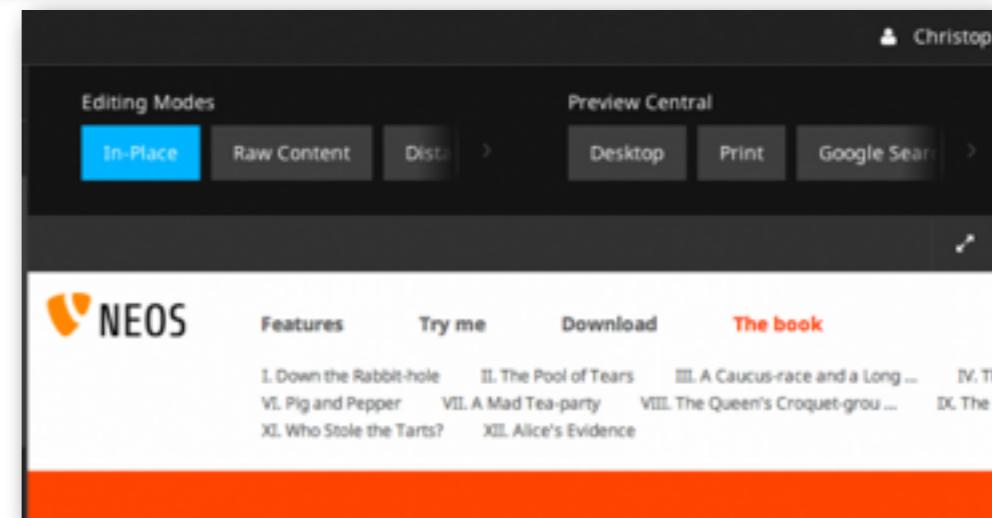
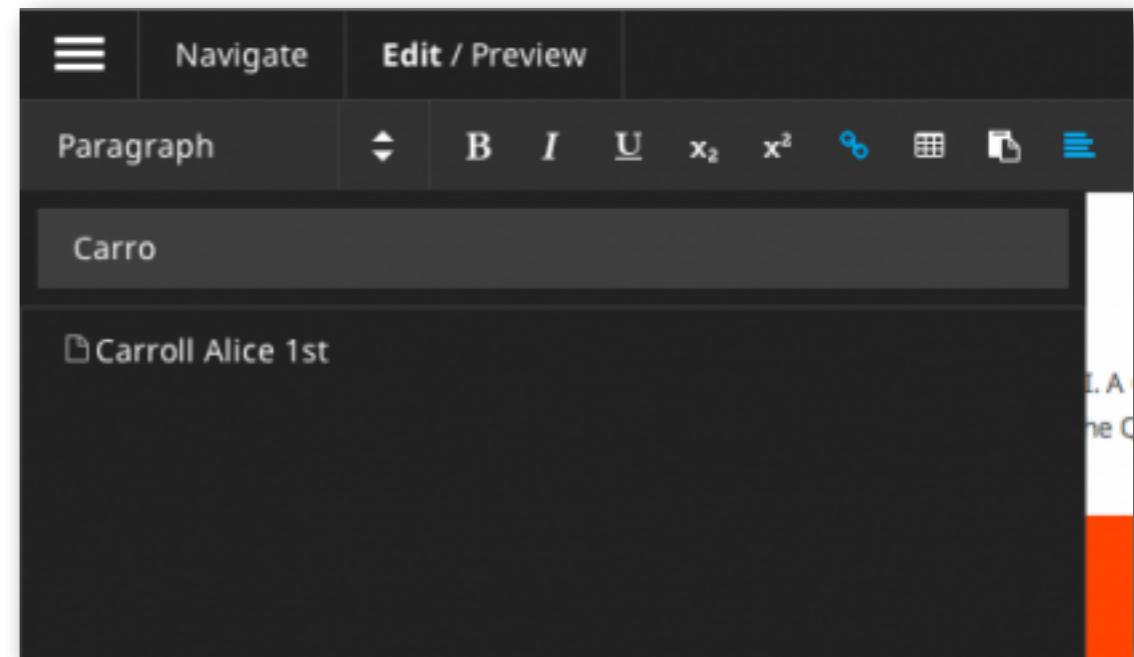
Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 5

- **Node Type Wechsel:** Node Types können nun geändert werden - dies ist für Seiten und für Inhalte möglich. Alle Eigenschaften, die im zu ändernden NodeType auch vorhanden sind, bleiben erhalten.
- **Asset Editor:** Der neue Asset Editor stellt zwei zusätzliche Property-Types für Nodes zur Verfügung um einzelne oder mehrere Assets im Inspektor zu editieren. Der Editor wird verwendet für die Property-Types „TYPO3MediaDomainModelAsset“ und “array<TYPO3MediaDomainModelAsset>”



Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 6

- **Asset Linking:** Neben dem Asset Editor, kann man Assets auch direkt im Content verlinken.
- **Visuelle Änderungen:** Das Edit/Preview Panel haben einen responsive Slider spendiert bekommen. Zusätzlich gibt es einen neuen Loading-Indicator, der per orangen Balken (anstelle eines Spinners) signalisiert, wann die aktuelle Aktion fertig gestellt ist.



Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 7

- **TypoScript**
 - Das Exception-Handling in TypoScript wurde verbessert und zeigt die originale Exception im Falle eines Fehlers.
 - TypoScript Object Implementations müssen nun „AbstractArrayTypoScriptObject“ implementieren, damit die Sub-Properties automatisch gemappt werden (**BREAKING CHANGE** für eigene Objekt-Implementierungen)
 - Es wurde ein neues BreadcrumbMenu TypoScript Objekt eingeführt, welches die Item States wie das Menu-Objekt hat.
- **Fluid**
 - Die neos:link.node und neos:uri.node ViewHelper-Argumente wurden mit den existierenden Fluid-ViewHelper synchronisiert.

Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 8

- **EEL and FlowQuery**
 - Die FlowQuery find-Operation arbeitet nun auch rekursiv - dies ermöglicht Abfragen, wie die folgenden:
 - Finde eine Node rekursiv in einer Site anhand eines Identifiers

```
 ${q(site).find('#60216562-0ad9-86ff-0b32-7b7072bcb6b2')}
```
 - Finde alle Nodes eines speziellen Typs rekursiv innerhalb einer Site

```
 ${q(site).find('[instanceof TYPO3.Neos.NodeTypes:Text'])}
```
 - Die Filter-Operation unterstützt nun den != Operator

```
 titlePropertyIsNotEmpty = ${q(node).is('title!="")}
```
 - Der instanceof Operator (als Filter) arbeitet nun auch auf Attributen:

```
 imagePropertyIsImage = ${q(node).is([image instanceof TYPO3\Media\Domain\Model\ImageVariant])}
```

Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 9

- **Content Repository**
 - Im TYPO3CR können nun Varianten einer Node - die so genannten „Content Dimensions“ gespeichert werden.
 - Eine Content-Dimension ist ein Aspekt eines Inhalts wie Lokalisierung, Personalisierung, oder Spezialisierung für einen Channel. Nodes können mehrere Varianten besitzen, jedes mit einem oder mehreren Werten für jede Dimension.
 - Die Dimensionen werden über die Settings (“TYPO3.TYPO3CR.contentDimensions”) konfiguriert und sind generisch. Jede Dimension hat einen Default-Wert, welche verwendet wird, wenn keine spezifische Dimension angegeben wird.
 - Content-Dimensions sind allerdings noch nicht im Backend verfügbar.

Release Notes TYPO3 Neos 1.1.0 (19.06.2014) - Teil 10

- **Node Types**

- Die NodeType-Konfiguration (`NodeTypes.yaml`) kann nun in mehrere Dateien aufgesplittet werden. Dies ist insbesondere für größere Seiten wichtig, um eine bessere Strukturierung der NodeType-Definitionen zu erhalten.
- Alle Dateien mit dem Pattern `NodeTypes.*.yaml` werden als NodeType-Definition verwendet und inkludiert. Eine einzelne `NodeTypes.yaml` Datei wird aber nach wie vor unterstützt.

Release Notes TYPO3 Neos 1.1.1 (27.08.2014) - Teil 1

TYPO3.Neos

- [TASK] Update references in documentation
- [BUGFIX] Children operation should only work with nodes
- [TASK] Improve ViewHelper documentation, add reference
- [BUGFIX] Inspector Image Cropper should not open if dummy-image is shown
- [TASK] Add hint for fixing symlinks in Behat testing documentation
- [BUGFIX] Remove neos-contentelement class from page metainformation
- [TASK] Change RUBY_VERSION check to match versions > 1.9
- [TASK] Use primary button styling in Site management
- [TASK] Remove link interception for Aloha floating menu
- [BUGFIX] lastVisitedNode should be reset when changing sites
- [BUGFIX] Throw exception for missing node in editable view helper
- [TASK] Add missing title attribute for "Toggle context structure"
- [TASK] Swap some access checks with workspace checks
- [BUGFIX] Class names missing for setup steps
- [BUGFIX] Doctrine eventListener registered without key
- [TASK] Fix code in nodetype examples
- [BUGFIX] Clicking outside the body should deselect active element
- [BUGFIX] Emptying a collection breaks create new
- [BUGFIX] Inline link editor shown when linking is disabled

Release Notes TYPO3 Neos 1.1.1 (27.08.2014) - Teil 2

TYPO3.Neos

- [BUGFIX] NodeSearchService working as before
- [TASK] Document missing configuration options for aloha
- [BUGFIX] Behat Tests are green again
- [BUGFIX] Set correct cache mode for ContentCollection by default
- [BUGFIX] Structure tree contains removed nodes after removal
- [BUGFIX] Content element overlay not displayed
- [BUGFIX] Find FlowQuery operation breaks with empty context
- [BUGFIX] Pages created without entering a name should work
- [BUGFIX] Fix small typo in 1.1.0 change log introduction
- [BUGFIX] (Sites Management) Importing site fails if kickstarter is not installed
- [BUGFIX] Secondary Inspector Panel should close on changed node
- [TASK] Add cache configuration to documentation example for shared footer
- [TASK] Add release notes to 1.1.0 change log

Release Notes TYPO3 Neos 1.1.1 (27.08.2014) - Teil 3

TYPO3.Neos.NodeTypes

- [BUGFIX] Avoid exception when rendering new asset list element

TYPO3.TYPO3CR

- [BUGFIX] ContextFactory does not check dimension configuration
- [BUGFIX] Behat Feature works with PHP 5.3
- [BUGFIX] Remove removed nodes without existing node when publishing
- [TASK] Make properties optional in ContextFactory->create()
- [BUGFIX] Danish/Norwegian letters converted incorrect in node name

TYPO3.TypoScript

- [BUGFIX] Underscore is not allowed as a TypoScript path value

TYPO3.NeosDemoTypo3Org

- [TASK] Remove unnecessary cache mode configuration for ContentCollection
- [TASK] Position YouTube content element video properties uppermost

Release Notes TYPO3 Neos 1.1.2 (03.09.2014) - Teil 1

Base Distribution

- [TASK] Update release scripts to include Media with Neos

TYPO3.Neos

- [TASK] Update references in documentation
- [BUGFIX] Better error checking for missing content collection nodes
- [BUGFIX] Focus lost for inline editing after deletion on every save
- [TASK] Enhance documentation
- [BUGFIX] Inline editable properties for documents not visible
- [BUGFIX] Content collections marked unpublished for new pages
- [BUGFIX] Wrapping pages results in PHP notice
- [BUGFIX] Backend redirects breaks with invisible/inaccessible nodes
- [BUGFIX] Enable/disable edit in create uses different selector
- [BUGFIX] respect "workspace" argument consistently when auto-creating child nodes
- [BUGFIX] Content collection falls back to first child if nodePath is empty
- [BUGFIX] Site Export should be able to handle broken assets/images
- [BUGFIX] Importing of resources broken due to wrong argument order

Release Notes TYPO3 Neos 1.1.2 (03.09.2014) - Teil 2

TYPO3.TYPO3CR

- [BUGFIX] Searching node properties works only case sensitive

TYPO3.TypoScript

- [BUGFIX] Ignore empty tags in @cache.entryTags configuration

TYPO3.Media

- [TASK] Unify button styling in Media management
- [TASK] Escape caption text in thumbnail view
- [BUGFIX] Functional tests use sys_get_temp_dir()

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 1

Unterstützung von Sprachen

- TYPO3 Neos unterstützt nun sogenannte „Content Dimensions“ - damit können beliebige Varianten einer Node angelegt werden. In Neos 1.2 wurden diese verwendet um Übersetzung zu realisieren. Die verfügbaren Sprachen müssen vom Integrator definiert werden und schon kann Content übersetzt werden.

Constraints

- In einem typischen Projekt erzeugt man eine Vielzahl eigener NodeTypes. Viele Nodes sollen aber nur in einem speziellen Kontext verwendet werden und nicht überall. Über sogenannte „node type constraints“ kann man die erlaubten Childnodes innerhalb einer gegebenen Node einschränken. Neben der Einschränkung der Kinder können damit auch die Enkel eingeschränkt werden.
- Node type constraints werden an folgenden Plätzen in Neos unterstützt:
 - "Create New" Panel in Content-Element Steuerungselementen
 - "Create New" Panel in den Navigations-Komponenten (Document- und Structure-Tree)
 - Bewegen in der Navigations-Komponente
 - Wechsel des NodeTypes im Inspektor

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 2

Editor Features

- Link Editor im Inspektor für Nodes, Assets und externe Links
- Möglichkeit per Shortcut ein externes Ziel anzugeben
- Seitenverhältnis beim Croppen von Bildern im Image Editor kann vordefiniert werden
- Im Inspektor können nun Settings als Tabs gruppiert werden
- Die aktuellste Aloha-Editor-Version wurde integriert
- Das Exception-Handling beim Rendering wurde stark verbessert
- Im Structure-Tree werden nicht publizierte Nodes farblich markiert
- Das „Menu-Content-Element“ hat nun eine manuelle Auswahlmöglichkeit für die inkludierten Seiten
- Neues Content-Element „Insert Records“ zum Einfügen von Records (Referenz auf ander Content-Elemente)
- Die Usability des Publishing-Button wurde optimiert
- Der Typ „date“ hat nun erweiterte Default-Values die gesetzt werden können
- Im Reference-Editor im Inspector werden nun Icon für die Nodes angezeigt
- Dialoge können nun auch per ESC-Taste geschlossen werden

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 3

Editor Features

- Sobald die Benutzersession ausläuft, wird der User gewarnt
- Der User wird vor Datenverlust gewarnt, wenn er während Speicher/Publizierungs-Prozessen navigiert
- Verbesserung der Stabilität beim Speichern
- Wenn die aktuelle Seite gelöscht wird, endet dies nun in einem 404 (und nicht mit einem Sprung zur nächsten Seite wie bisher)
- Wenn der Login fehl schlägt, bleibt der Benutzername ausgefüllt
- Im Aloha-Link-Feld werden nun die korrekten Icons für die Nodes verwendet
- Im Inline-Link-Editor wird nun ein Platzhalter verwendet und „http://“ entfernt
- Usability der „New/Paste“ Buttons wurde verbessert (Indikator, dass ausgeklappt werden kann)
- Im „New-Panel“ (Navigate/Content) können die dort enthaltenen Nodes nun sortiert werden

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 4

Inspector Features

- Es gibt nun die Möglichkeit, die Options im SelectBoxEditor dynamisch vom Server zu laden
- Unterstützung für die Auswahl und Gruppierung mehrerer Optionen im SelectBoxEditor
- Der Date-Editor unterstützt nun auch die Einstellung der Uhrzeit
- Es gibt nun einen Textarea-Editor um mehrzeilige Werte im Inspector zu pflegen (z.B. Description)
- Button-Labels und Highlighting-Modes für Editor-Buttons sind im Inspector nun konfigurierbar (Einführung der Optionen buttonLabel und highlightingMode für die editorOptions)
- Das Datumsformat im Inspector wurde zu „PHP date“ geändert
- Es ist nun möglich bestimmte Nodes aus der Anzeige im NodeTree zu exkludieren

Inline-Editing Features

- Es gibt nun eine Placeholder-Unterstützung für Inline-Editable Eigenschaften
- Möglichkeit <code> im Aloha zu verwenden (deaktiviert per Default)

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 5

TypeScript 2 Features

- Es gibt nun einen TypeScript Prototype um HTTP Headers und Response Status zu handeln

```
prototype(TYPO3.TypoScript:Http.ResponseHead)
prototype(TYPO3.TypoScript:Http.Message)
```
- Es ist nun möglich, mehrere TS-Dateien mit einer Anweisung zu inkludieren

```
include: SomeDirectory/* (Alle .ts2 Dateien in diesem Verzeichnis)
include: SomeDirectory/**/* (Alle .ts2 Dateien in diesem Verzeichnis und darunter)
```
- @if Meta-Property um Pfade konditional zu evaluieren: `@if.isSpecial = ${q(node).property('special')}`
- Es gibt nun ein NodeUri TypeScript-Objekt, um auf Nodes zu verlinken

```
prototype(TYPO3.Neos:NodeUri)
```
- Es gibt nun zwei TS-Objekte um Bilder zu verarbeiten (Analog zum Media:Image ViewHelper)

```
prototype(TYPO3.Media:ImageTag)
prototype(TYPO3.Media:ImageUri)
```

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 6

TypeScript 2 Features

- Einführung der FlowQuery Operation „has“ - Filter für Nodes die das spezifizierte Child enthalten
TYPO3\TYPO3CR\Eel\FlowQueryOperations\HasOperation
- Das TS-Objekt BreadCrumMenu ersetzt BreadCrumb (wurde bereits in 1.1 so eingeführt)
- Die Attribute des Menu-Items haben nun direkten Zugang zum Item im Kontext
- Der Renderer kann nun direkt angegeben werden

```
root {
    fooMatcher {
        condition = ${TRUE}
        renderer = ${q(node).property('title')}
    }
}
```

Backend-Interaction Features

- Es gibt nun einen externen Event, sobald Nodes erzeugt oder entfernt werden
- Das Neuladen des Backends kann nun auch extern angestoßen werden
- Es gibt nun externe Events, sobald Panels geöffnet bzw. geschlossen werden und wenn das Layout geändert wurde (Neos.LayoutChanged, Neos.NavigatePanelOpened, Neos.NavigatePanelClosed, Neos.InspectorPanelOpened, Neos.InspectorPanelClosed, Neos.EditPreviewPanelOpened, Neos.EditPreviewPanelClosed, Neos.MenuPanelOpened, Neos.MenuPanelClosed)

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 7

Fluid Features

- Im link.node ViewHelper kann nun auf das Label zugegriffen werden
`<n:link.node node="subpage">{linkedNode.label}</n:link.node>`
- Analog zum link.module ViewHelper gibt es nun auch einen uri-module ViewHelper
- Im NodeLinkingService kann nun zur Site-Node verlinkt werden (indem ~ im link.node ViewHelper verwendet wird)

Zusätzliche Features

- Es gibt nun ein Dimensions/Language-Menu um zu anderen Dimensionen zu verlinken
`languageMenu = TYPO3.Neos:DimensionMenu {
 dimension = 'language'
}`
- Konfigurationen (Settings, NodeTypes, Policies, Routes, Caches, Objects, Views) werden im neuen Backend-Modul „Configuration“ angezeigt. Eventuelle Fehler in der Konfiguration werden hier ebenfalls angezeigt

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 8

Zusätzliche Features

- Es wurden zahlreiche Mixins in den NodeTypes eingefügt, die in eigenen Objekten verwendet werden können
 - 'TYPO3.Neos.NodeTypes:TextMixin':
 - 'TYPO3.Neos.NodeTypes:ImageMixin':
 - 'TYPO3.Neos.NodeTypes:ContentImageMixin':
- Einführung von REST-Controller, die Backend Requests handeln (XML und JSON)
beispielweise: GET http://foo.com/neos/service/nodes/123467890abcdef
- Einführung von „Data sources“ um Content im Backend via AJAX zur Verfügung zu stellen
(siehe <https://git.typo3.org/Packages/TYPO3.Neos.git/commit/58f7d7d401734b38e6083ace0f6fe390a435864b>)
- In der Funktion ChangePropertyValue wurde ein einfacher Search/Replace Mechanismus inkludiert (erlaubt search & replace in Text properties bei Node Migrationen)
- Die Flow-Kommandozeile wurde um einen Befehl zur Aktivierung und Deaktivierung von Domains ergänzt
 - ./flow domain:activate & domain:deactivate
- Node type properties können nun auch Arrays sein (um multiple Auswahlen im SelectBoxEditor zu ermöglichen)

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 9

BREAKING CHANGES

- FlowQueryOperations befinden sich nun unter: TYPO3\TYPO3CR\Eel\FlowQueryOperations
- Links zu Shortcut-Nodes zeigen nun auf auf das End-Target und nicht auf die dazwischen liegende URL
- TYPO3.Neos/Inspector/Editors/HtmlEditor (YAML) wurde ersetzt durch TYPO3.Neos/Inspector/Editors/CodeEditor
- Im Aloha-Editor wurden `` und `<i>` durch `` und `` ersetzt. Um nicht formatierte Blöcke werden nun automatisch `<p>` Tags platziert. Externe Links bekommen nun automatisch das Attribut `_blank`
- Die alten Pfade Private/TypoScripts (/Library)/Root.ts2 wurden nun geändert zu /Private/TypoScript/Root.ts2
- Der IncludeJavaScriptViewHelper ViewHelper wurde als veraltet deklariert
- Die „document layout“ Properties im Inspector sind per Default nicht sichtbar
- Im Inline-Editor sind die Auszeichnungen „Underline“, „Superscript“ und „Subscript“ per Default deaktiviert

Release Notes TYPO3 Neos 1.2.0 beta1 (14.11.2014) - Teil 10

BREAKING CHANGES

- Das NodeLabel-Management wurde überarbeitet:

```
'My.NeosSite:SomeNodeType':  
    label: "{q(node).property('someLabelProperty')}"
```

oder

```
'My.NeosSite:SomeNodeType':  
    label:  
        generatorClass: 'My\\NeosSite\\TYPO3CR\\SomeNodeLabelGenerator'
```

- Kickstart einer neuen Site funktioniert (zur Zeit) nicht, wenn die Demo-Site aktiv ist

Daher vorher: ./flow package:deactivate TYPO3.NeosDemoTypo3Org

Release Notes TYPO3 Neos 1.2.0 beta2 (25.11.2014) - Teil 1

- Die TypoScript-Objekte `ImageUri` und `ImageTag` wurden von `TYPO3.Media` nach `TYPO3.Neos` verschoben
- Das Workspaces-Modul wurde leicht überarbeitet (Usability, Optik)
- Update Aloha 1.1.3 zu 1.1.5
- Das Label einer Node wird nun nach 100 Zeichen abgeschnitten und zudem gecropt:

```
'TYPO3.Neos:Node':  
    label: "${String.cropAtWord(String.trim(String.stripTags(q(node).property('title') ||  
        q(node).property('text') || ((node.nodeType.label ||  
            node.nodeType.name) + ' (' + node.name + ')'))), 100, '...')}"
```

- Der Grenzwert (ab wie vielen eingegebenen Buchstaben nach einer Referenz gesucht wird) in der Suche des Reference-Editors kann nun eingestellt werden - default ist 2.

```
editorOptions:  
    threshold: 4
```

- Im Demo-Paket „TYPO3.NeosDemoTypo3Org“ wurden vier Nodes mit den jeweils möglichen Shortcut-Arten auf der Seite „Shortcuts“ zugefügt

Release Notes TYPO3 Neos 1.2.0 (11.12.2014)

- Keine weiteren Änderungen zu 1.2.0 beta 3 und 1.2.0 beta 2

Features von TYPO3 Neos

Features von TYPO3 Neos

- Maximal erweiterbares Enterprise CMF - Content Management Framework
- Basis ist das stabile TYPO3 Flow Framework, welches bereits in Version 2.0 vorliegt
- Einfacher und flexibler Content Export/Import (bisher nur per Kommandozeile)
- Multi-Domain Support
- Multi-Language Support
- Modernste Technologie und Paradigmen unter der Haube (DDD, AOP, RequireJS, EmberJS, ...)
- Intuitives Benutzer-Interface
- Wireframe Mode - Content-Editing ohne Template
- Webbasiertes Package Management

Features von TYPO3 Neos

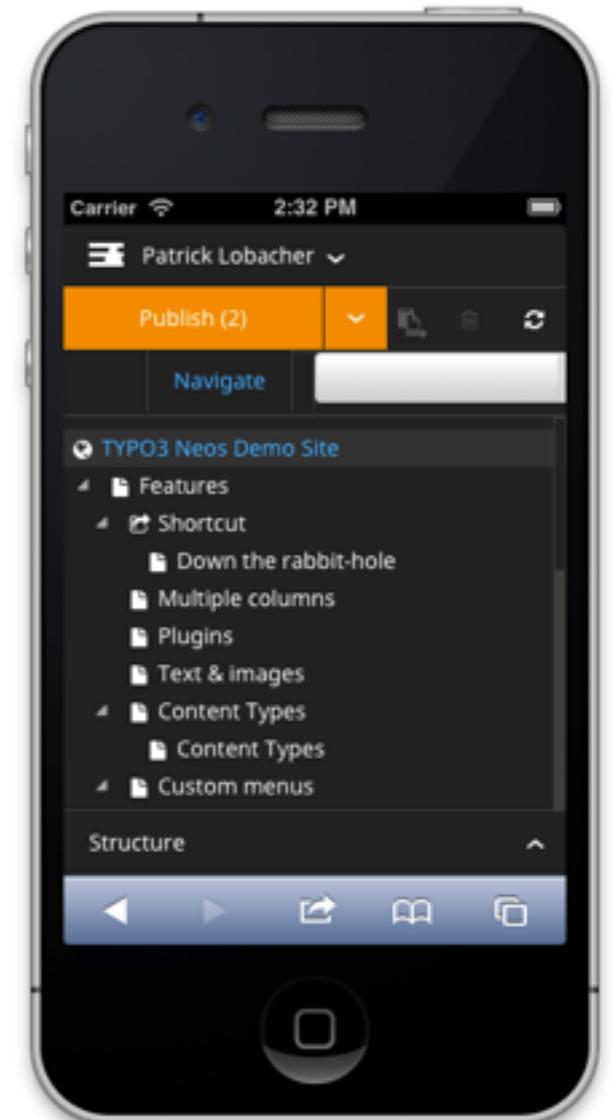
- In-place Content-Editing
- Eigene Content-Elemente (inkl. In-place Content-Editing) leicht möglich
- Integration von TYPO3 Flow Applikationen als Plugins (Packages)
- TypoScript 2 / Eel / FlowQuery
- Workspaces (Nicht Multiuser-fähig bislang)
- Custom Single-Sign-On / Custom Authentication per Programmierung möglich
- Audit Logging
- TYPO3 Surf für das automatische Deployment (Integration mit CI Server wie Jenkins)

Bislang fehlende Features

- Webbased Installer (Kommandozeile mit Root-Zugriff notwendig)
- Mehrsprachigkeit im User Interface (geplant für 1.3)
- Benutzer ohne Admin-Rechte (Redakteure, es gibt zwar einen „Redakteur-User“, dieser ist aber bislang lediglich ein „Non-Admin“) (geplant für 1.3)
- Benutzerrechte / Benutzergruppen (geplant für 1.3)
- ACLs (geplant für 1.3)
- Öffentliche Extensions (es gibt noch kein Package Repository)
- Gemeinsam genutzte Workspaces
- Versionierung (UI-Interface)
- Non-Core-Features: News, Anbindung an 3rd-Party-Systeme, Slider, SSL, Google Analytics Einbindung, ... (einiges davon geplant für 1.3)
- REST (im Backend bereits implementiert und genutzt) (geplant für 1.3)

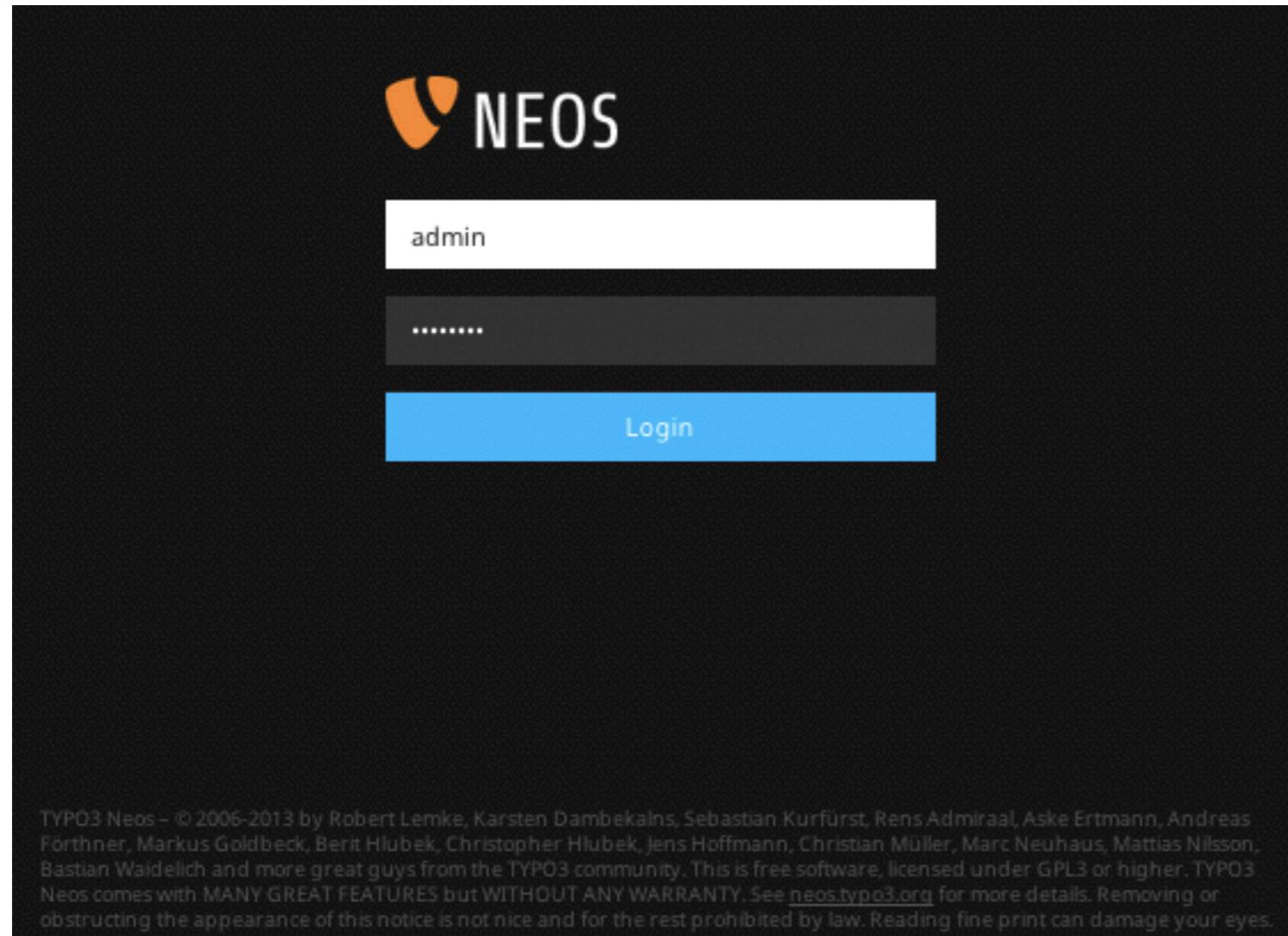
Bislang fehlende Features

- Logs (User, Admin, ...)
- SEO-Features (Seiteneigenschaften: noindex,nofollow, keywords, description, canonical, ...) (geplant für 1.3)
- Richtiges RWD-Interface (Interface funktioniert nur mit ca. 1000px und größer)
- RWD-Rendering von Bildern
- Digital Asset Management (es gibt einen MediaBrowser)
- Formular-Designer (man kann lediglich programmatisch vorbereitete Formulare auswählen)
- Content Synchronization and Syndication
- Admin-Interface um beliebige Datensätze zu bearbeiten
- ...



Aufbau der Neos Oberfläche

Login

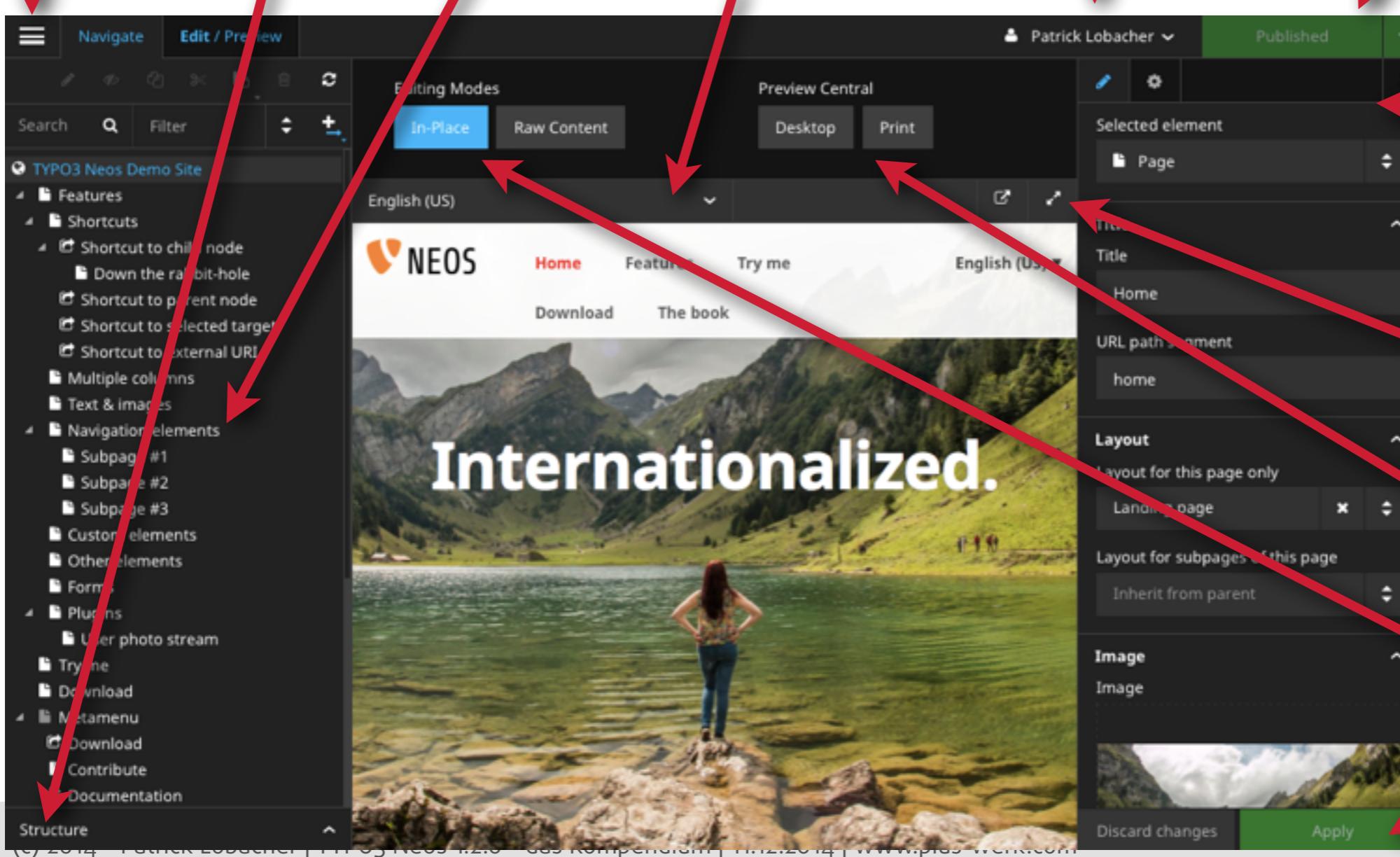


Aufbau der Admin-Oberfläche

Funktionen Struktur-
Menü Ansicht Page-Browser
 (Seitenbaum)

Umschaltung
der Dimension
(hier „Sprache“)

User-Administration
Logout und Settings



Publisher
Publizieren, bei Klick auf Pfeil Möglichkeit zu „Auto-Publish“

Inspector
Content-bezogene Eigenschaften, z.B. Seiten-eigenschaften, o.ä.

Voll-Bild
ohne Backend

Preview-Mode
z.B. Print-Anzeige

Editing-Modes
z.B. Anzeige ohne Design

Apply

Inspector

Selektiertes Element

z.B. Headline, Content-Collection, Page, ...

The screenshot shows the TYPO3 Neos editor interface. On the left is a large preview image of a woman standing by a lake with mountains in the background, with the word "Internationalized." overlaid. To the right is the inspector panel. At the top, it shows the user "Patrick Lobacher" and the status "Published". Below that is a toolbar with icons for edit, preview, and settings. A red arrow points from the text "Selected element" to a dropdown menu that says "Page". The main part of the panel contains sections for "Title" (set to "Home"), "URL path segment" (set to "home"), "Layout" (set to "Landing page"), "Layout for subpages of this page" (set to "Inherit from parent"), and "Image" (with a preview of a mountain scene). At the bottom are "Discard changes" and "Apply" buttons.

Properties

Kontextsensitive Eigenschaften, z.B.
Seiten-eigenschaften, o.ä.

The screenshot shows the "Properties" panel for a selected page element. It includes sections for "Selected element" (set to "Page") and "Visibility". The "Visibility" section has a "Hide before" field set to "No date set" and a calendar view showing December 2014. The calendar highlights the 12th of December. At the bottom are buttons for "Today" and "Apply".

Funktionen-Menü

Content

Hier können die verschiedenen Sites umgeschaltet werden

Workspaces

Verwaltung der Arbeitsbereiche

Media-Browser

Verwaltung der Medien

User Management

Benutzer-Verwaltung

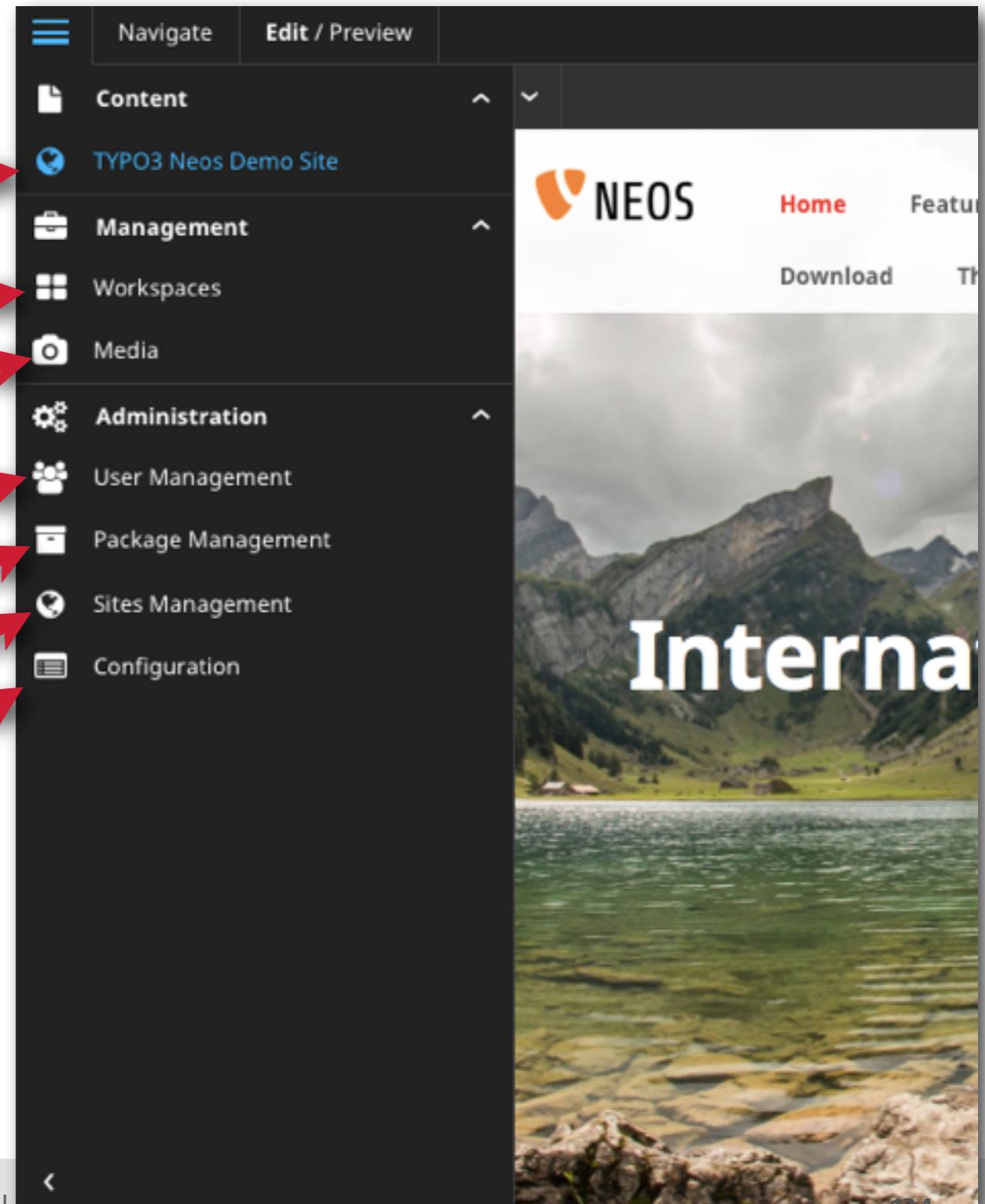
Package Management

Paket-Verwaltung

Sites Management

Verwaltung der Websites

Configuration



Workspaces Verwaltung

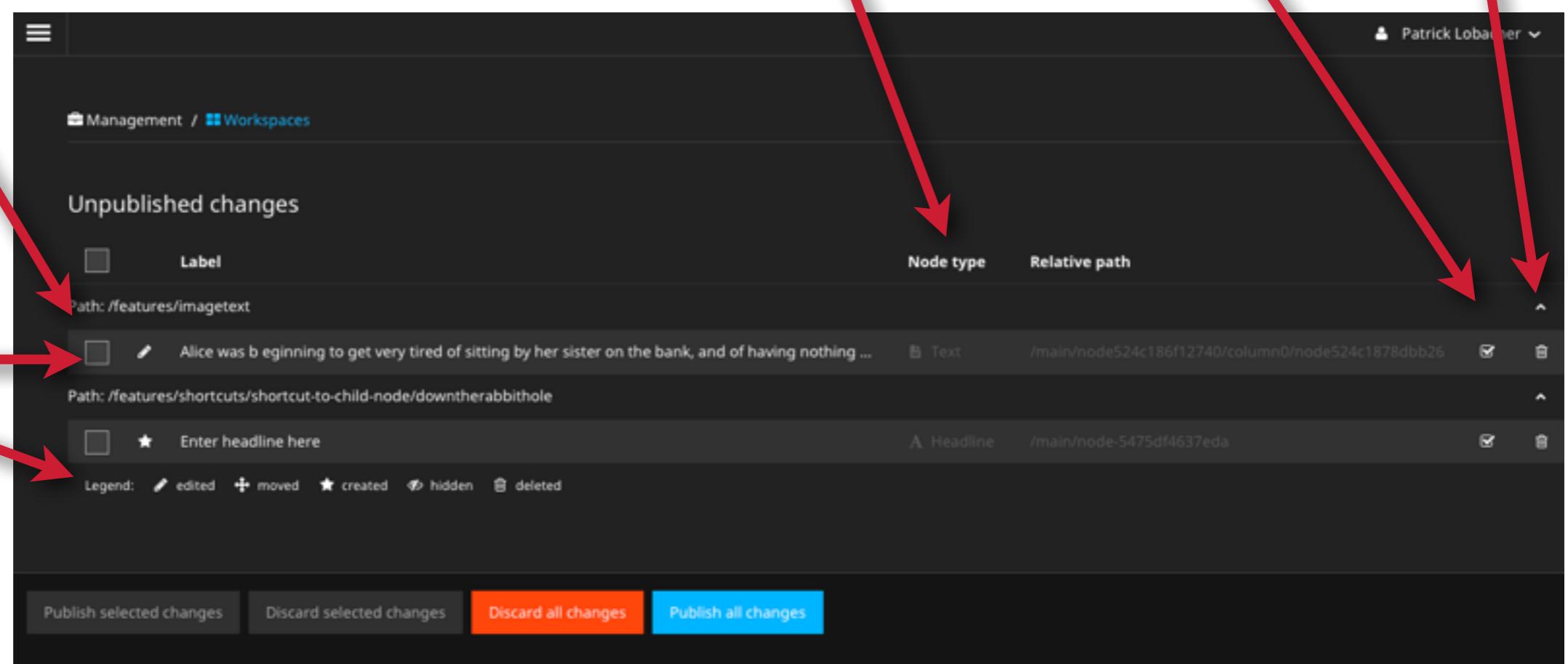
Auswahl
einzelner
Änderungen

Typ der
Änderung

Angabe
zum
Node-Typ

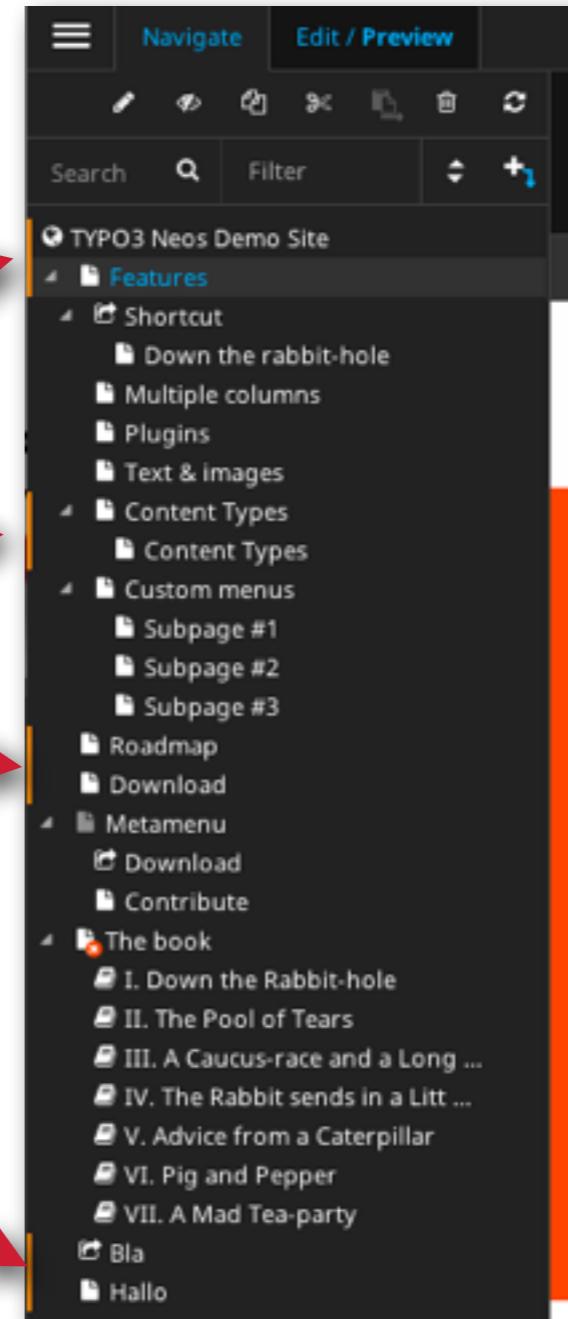
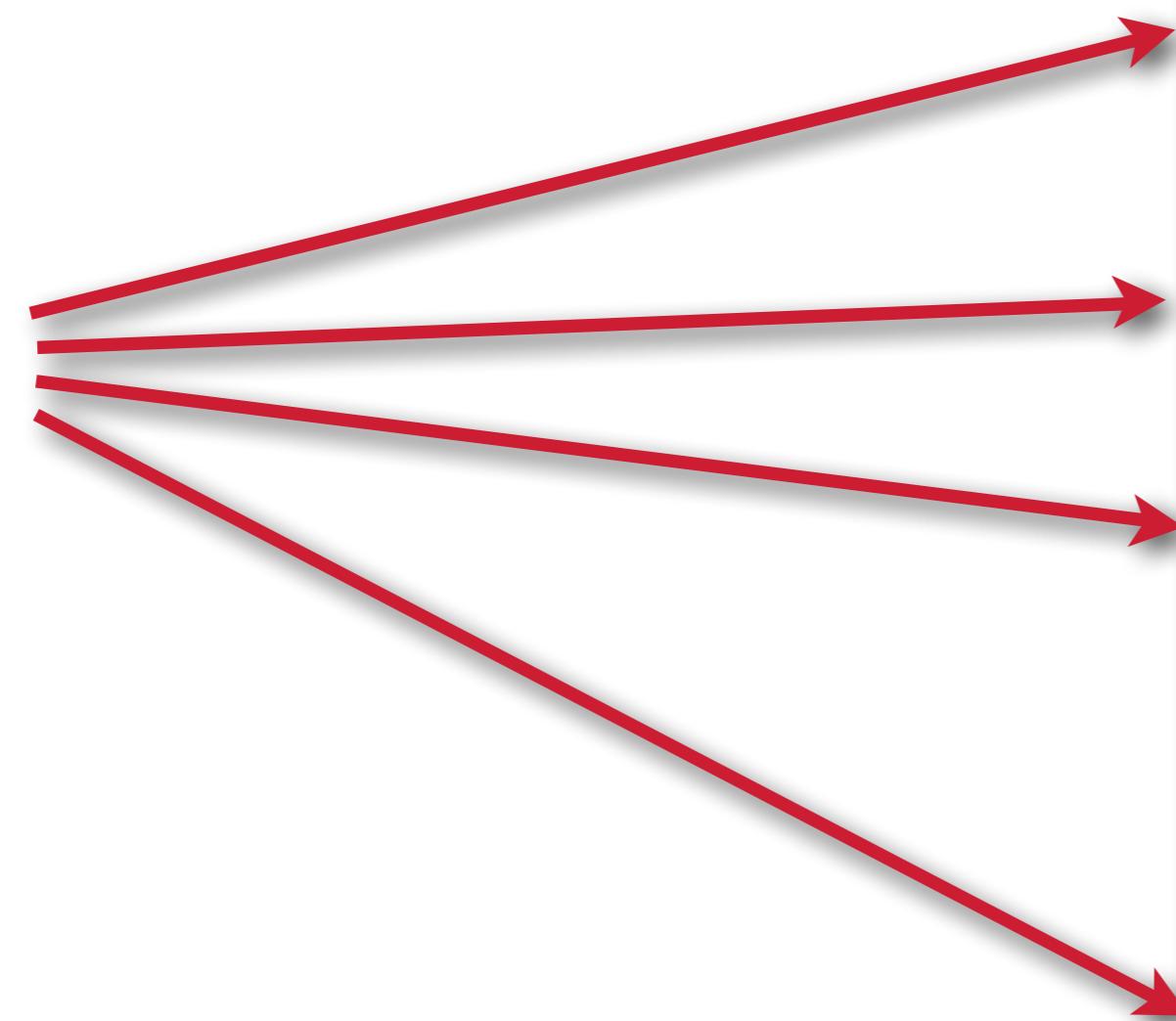
Publish
Publizieren

Discard
Verwerfen



Workspaces Verwaltung

Seiten, die Änderungen enthalten werden im Seitenbaum orange markiert



User-Verwaltung

Rolle

View
Ansehen

Edit
Editieren

Delete
Löschen

The screenshot shows the TYPO3 Neos User Management interface. At the top, there's a navigation bar with 'Administration / User Management'. Below it is a table with two rows of user data:

Username	Name	Roles	Actions
rocky	Rocky Lobacher	TYPO3.Neos:Editor (TYPO3.TYPO3CR:Administrator)	Info Edit Delete
admin	Patrick Lobacher	TYPO3.Neos:Administrator (TYPO3.Neos:Editor)	Info Edit Delete

A red arrow points from the 'Create new user' button at the bottom left to the 'Create new user' text above the table. Another red arrow points from the 'Roles' column header to the 'Roles' column. A third red arrow points from the 'Edit' column header to the edit icons in the table. A fourth red arrow points from the 'Delete' column header to the delete icons in the table.

Neuen
Benutzer
anlegen

User-Verwaltung - Neuer User

Einen neuen User anlegen - es muss lediglich ein Username, das Passwort und die persönlichen Daten angegeben werden. Zusätzlich wird die Rolle (Admin oder Editor) gewählt.

The screenshot shows the 'Create a new user' form in the TYPO3 Neos User Management. It is divided into two main sections: 'User data' on the left and 'Personal Data' on the right. In the 'User data' section, the 'Username' field contains 'Rocky'. In the 'Personal Data' section, the 'First Name' field also contains 'Rocky'. Both password fields ('Password' and 'Repeat password') show four asterisks ('****'). Under 'Role', the 'Editor' radio button is selected. At the bottom, there are 'Cancel' and 'Save user' buttons, with 'Save user' being highlighted in blue.

Administration / User Management

Create a new user

User data

Username

Rocky

Password

.....

Repeat password

.....

Role

Administrator Editor

Personal Data

First Name

Rocky

Last Name

Lobacher

Cancel Save user

User-Verwaltung - Editieren

Hier können die Userdaten editiert werden und zudem auch weitere Daten angeben werden - z.B. Title

Zusätzlich kann man hier beliebig viele „Electronic Addresses“ anlegen - also z.B. Email-Adressen oder Skype

Durch die Angabe „Primary“ wählt man die Haupt-Adresse aus

Electronic addresses			
Type	Usage	Identifier	Primary
Email	Home	rocky@lobacher.de	<input type="radio"/>
Aim			<input type="radio"/>
ICQ			<input type="radio"/>
Jabber			<input type="radio"/>
Msn			<input type="radio"/>
Sip			<input type="radio"/>
Skype			<input type="radio"/>
Url			<input type="radio"/>
Yahoo			<input type="radio"/>

Add electronic address

Cancel Save user

Administration / User Management

Edit user "rocky"

User Data Personal Data

Username	Title
rocky	
Password	First Name
	Rocky
Repeat password	Middle Name
Role	Last Name
<input type="radio"/> Administrator	Lobacher
<input checked="" type="radio"/> Editor	Other Name
	Alias
	rocky

Electronic addresses

Type	Usage	Identifier	Primary
			<input type="button" value="Add electronic address"/>

Cancel Save user

User-Verwaltung - Info

Die Funktion „Info“ zeigt eine Übersicht über alle eingegebenen Daten an

The screenshot shows the TYPO3 Neos User Management interface. On the left, there's a sidebar with navigation links like 'Administration', 'Content', 'Media', 'User Management', 'Log Out'. Below it, a list of users is shown with 'rocky' selected. The main content area displays 'Details for user: rocky'. It's divided into 'User Data' and 'Personal Data' sections. Under 'User Data', the information is:

Username	rocky
Alias	rocky
Creation date	12.12.2013 - 06:23:43
Roles	TYPO3.Neos:Editor (TYPO3.TYPO3CR:Administrator)

Under 'Personal Data', the information is:

First Name	Rocky
Last Name	Lobacher

At the bottom of the main view, there are three buttons: 'Back', 'Delete User' (highlighted with a red arrow), and 'Edit User'. A red arrow also points from the 'Delete User' button to a lightbox dialog on the right. The dialog asks 'Do you really want to delete user "rocky"?'. It contains a message: 'This will delete the user and his personal workspace, including all unpublished content. This operation cannot be undone.' There are 'Cancel' and 'Yes, delete the user' buttons.

Ein Klick auf „Delete“ öffnet eine Lightbox, in welcher noch einmal nachgefragt wird, ob man wirklich löschen möchte

Package-Verwaltung

Auswählen

Aktionen
auf der
Auswahl

Freeze & Unfreeze Deactivate & Activate Delete

<input type="checkbox"/>	Package Name	Version	Package Key	Package Type	Freeze & Unfreeze	Deactivate & Activate	Delete
Application							
Framework							
<input type="checkbox"/>	typo3/eel		TYPO3.Eel	typo3-flow-framework	*		trash
<input type="checkbox"/>	typo3/flow		TYPO3.Flow	typo3-flow-framework	*		trash
<input type="checkbox"/>	typo3/fluid		TYPO3.Fluid	typo3-flow-framework	*		trash
<input type="checkbox"/>	typo3/kickstart		TYPO3.Kickstart	typo3-flow-framework	*		trash
<input type="checkbox"/>	typo3/party		TYPO3.Party	typo3-flow-framework	*		trash
Libraries							
Sites							
<input type="checkbox"/>	typo3/neosdemotypo3org		TYPO3.NeosDemoTypo3Org	typo3-flow-site	*		trash
<input type="checkbox"/>	typovision/demo		Typovision.Demo	typo3-flow-site	*		trash
<input type="checkbox"/>	typovision/test		Typovision.Test	typo3-flow-site	*		trash

Site-Verwaltung Übersicht

Neue Seite anlegen	Status der Site	Edit	Deactivate & Activate	Delete				
Add new site	Administration / Sites Management	Name TYPO3 Neos Demo Site	Rootnode name neosdemotypo3org	Resource package key TYPO3.NeosDemoTypo3Org	State Active			

Site-Verwaltung Editieren

The screenshot shows the TYPO3 Neos Site Management interface. On the left, the 'Site' configuration for 'TYPO3 Neos Demo Site' is displayed, including fields for Name, Root node name, and State. On the right, the 'Domains' section lists a single domain 'neos.dev' with status 'Active'. A blue button labeled 'Add domain' is visible. Red arrows point from the text labels on the right to specific UI elements: one arrow points to the edit icon in the domain list, another to the deactivate icon, and a third to the delete icon. A fourth red arrow points to the 'Add domain' button.

Administration / Sites Management

TYPO3 Neos Demo Site

Site

Name

TYPO3 Neos Demo Site

Root node name

neosdemotypo3org

State

Active

Domain

neos.dev

Status

Active

Domains

Add domain

Cancel Delete this site Deactivate site Save

Domain
editieren

Domain
deaktivieren

Domain
löschen

Domain
zufügen

Raw Content Mode - Anzeige der Elemente ohne Design

The screenshot shows the TYPO3 Neos backend interface in Raw Content mode. The left sidebar contains a navigation tree for the 'TYPO3 Neos Demo Site' under 'Features'. The main content area displays the 'MAIN' page with sections for 'What Makes Neos Special' and 'Developer Happiness For The World'. The 'Inspector' panel on the right shows details for the selected 'Page' element, including its title 'Features', name (URL) 'features', and visibility settings. The layout section indicates it uses 'Landing page' for this page and 'Default' for subpages.

MAIN

What Makes Neos Special

TYPO3 Neos is not just another CMS. Instead, the TYPO3 Community has created it based on the previous experiences. It's not the first CMS built, but instead there is a big amount of knowledge which has accumulated over the last 15 years.

Developer Happiness For The World

We know that a website does not only contain static parts. In today's world, you often integrate external services, have to model complex business logic or tweak the output of the website in a very fine-grained way. That's why our main goal is developer happiness. We want you to be effective, being able to re-use concepts and code across your websites.

Of course, that is no free ride: You need to learn the concepts used in Neos. However, by implementing established practices in software development, your knowledge can be applied to a wide variety of languages, frameworks and use-cases. We want to help you become better programmers!

The Learn Once, Run Everywhere Philosophy is the main guidance principle for the development of Neos: All components are usable inside the CMS context, but also in stand-alone web applications. That of course also includes your own components.

Based On A Robust Foundation: TYPO3 Flow

Patrick Lobacher

Published

Inspector

Selected element

Page

Title

Features

Name (URL)

features

Visibility

Hide before

No date set

Hide after

No date set

Hide

Hide in menus

Layout

Landing page

Layout for subpages of this page

Default

Discard changes

Apply

Preview Zentrale - Anzeige-Alternativen

The screenshot shows the TYPO3 Neos Preview Central interface. The top navigation bar includes 'Edit / Preview' and 'Publish (1)'. The left sidebar shows a tree view of the 'TYPO3 Neos Demo Site' structure, with 'Features' selected. The main content area displays several blocks:

- What Makes Neos Special**: A block containing text about Neos being built on TYPO3 Flow.
- Developer Happiness For The World**: A block containing text about the developer-focused nature of Neos.
- Based On A Robust Foundation: TYPO3 Flow**: A block containing text about TYPO3 Flow's role in Neos.
- Built for Extensibility**: A block containing text about Neos' extensibility through plugins.
- In-Page Editing**: A block containing text about in-page editing features.

At the bottom right, there are 'Discard' and 'Apply' buttons.

Seitenbaum

Klick auf Plus-Symbol
legt neue Seite an

Seitentitel editieren

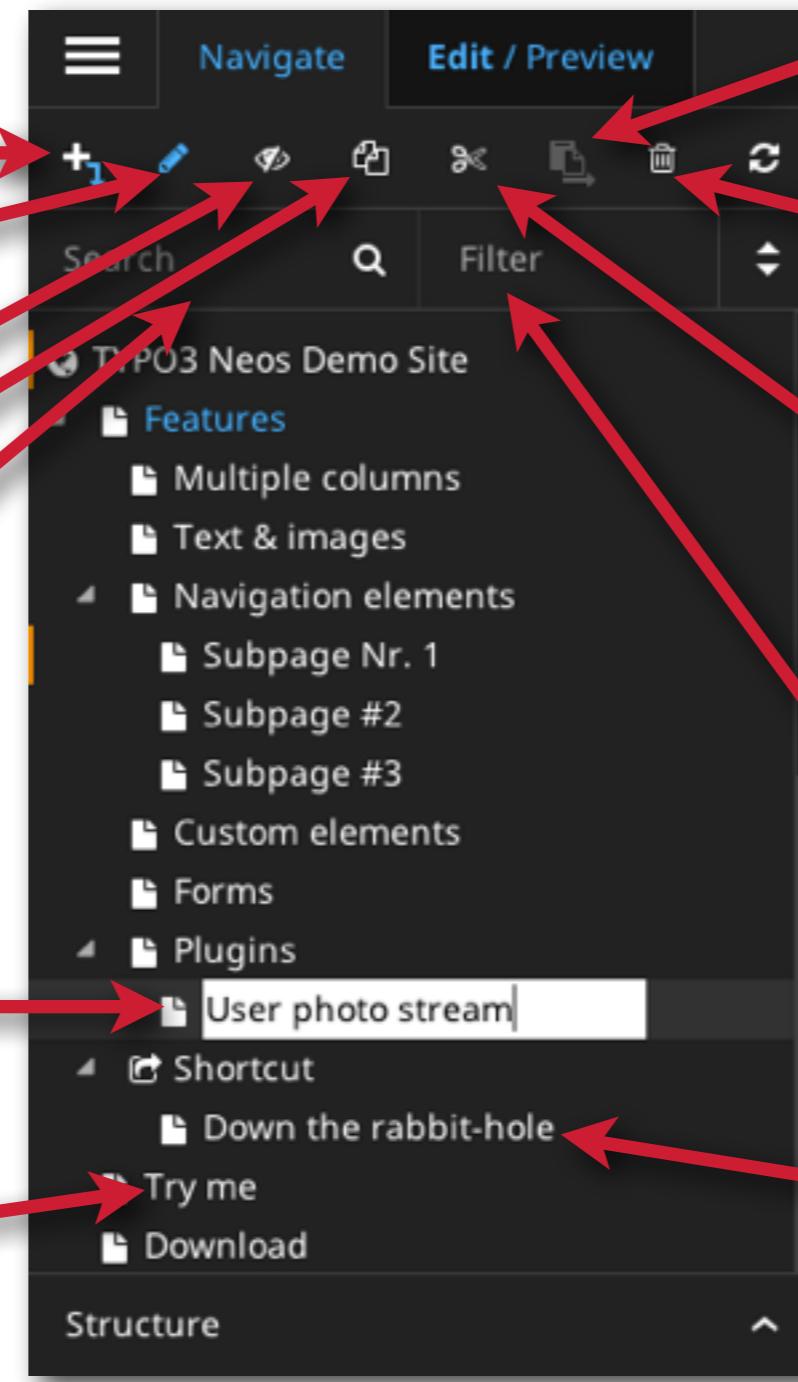
Seite unsichtbar/sichtbar

Seite kopieren

Suche im Seitenbaum

Doppelklicken um den
Seitentitel zu editieren

Klick auf Seitentitel um
Seite in der Preview
rechts zu sehen



Einfügen (innerhalb)

Refresh

Löschen der
ausgewählten Seite

Ausschneiden

Filtern des Seitenbaums nach
Typ (Seite, Shortcut, ...)

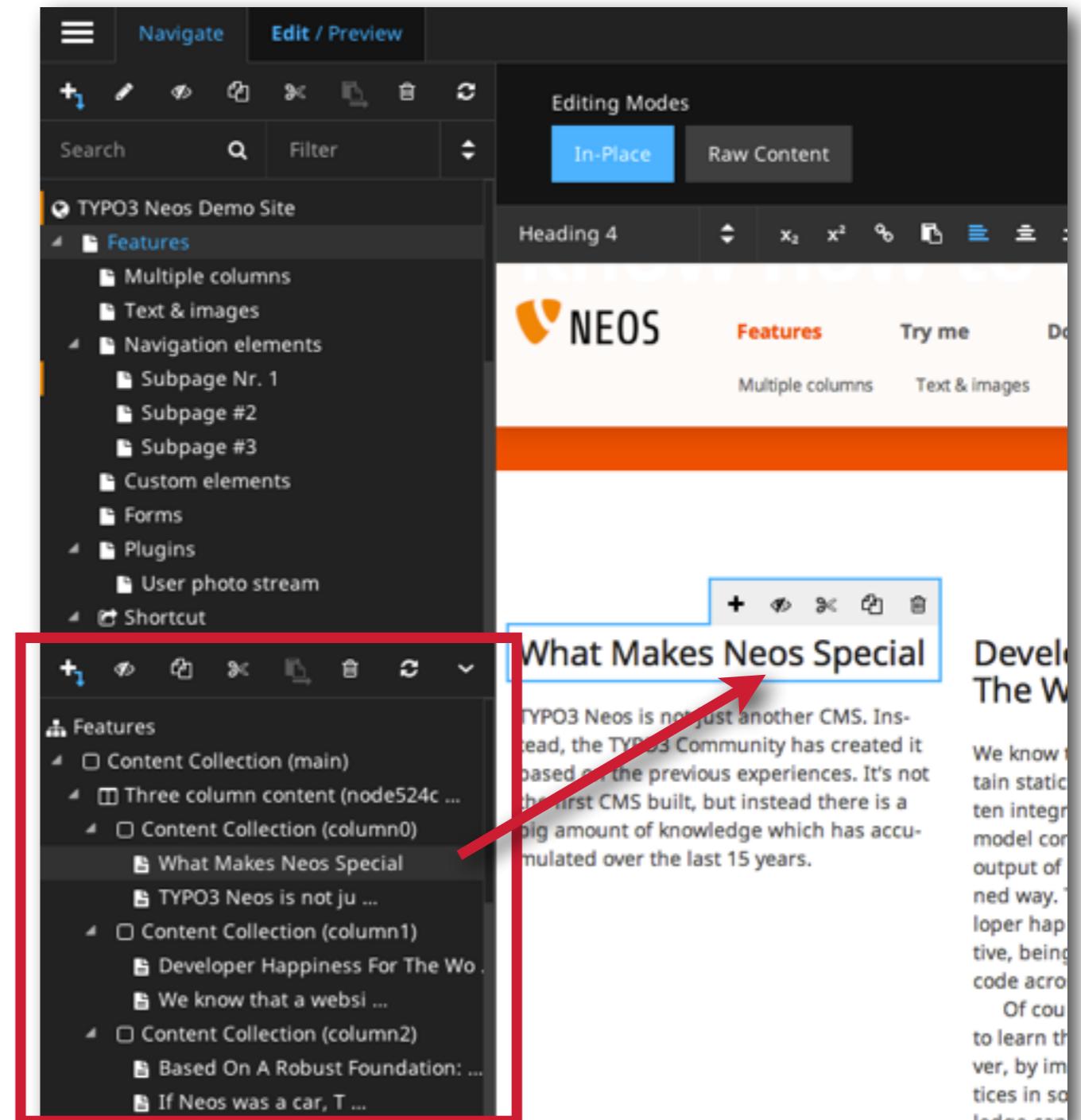
Drag&Drop um Seite
zu verschieben

Strukturbbaum

Im Strukturbbaum werden alle Elemente (sichtbare und unsichtbare) der aktuellen Seite hierarchisch aufgelistet

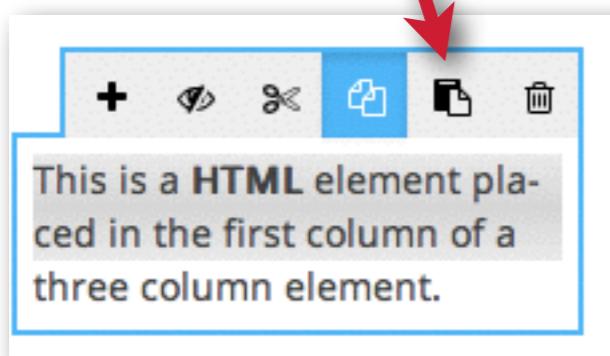
Auch hier hat man die selben Icons für Funktionen wie „Neu“, „Ausblenden“, „Kopieren“, ... zur Verfügung

Klick auf die Überschrift öffnet rechts die Eigenschaften im Inspector und platziert den InPlace-Editor an der entsprechenden Stelle

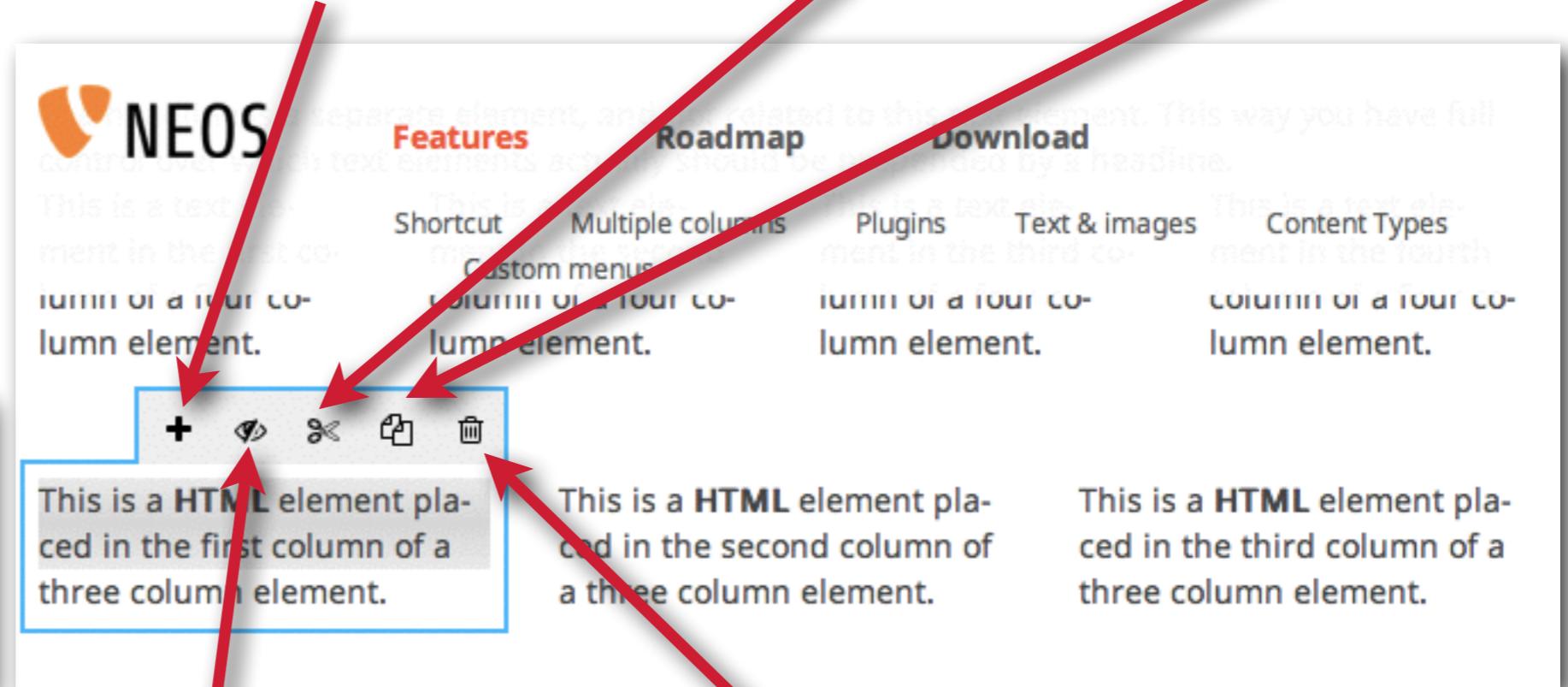


Content-Funktionen

Sobald man auf „Kopieren“ oder „Ausschneiden“ klickt, erscheint „Einfügen“



Neues Content-Element nach dem ausgewählten anlegen



Content-Element ausblenden

Content-Element löschen

Content-Funktionen

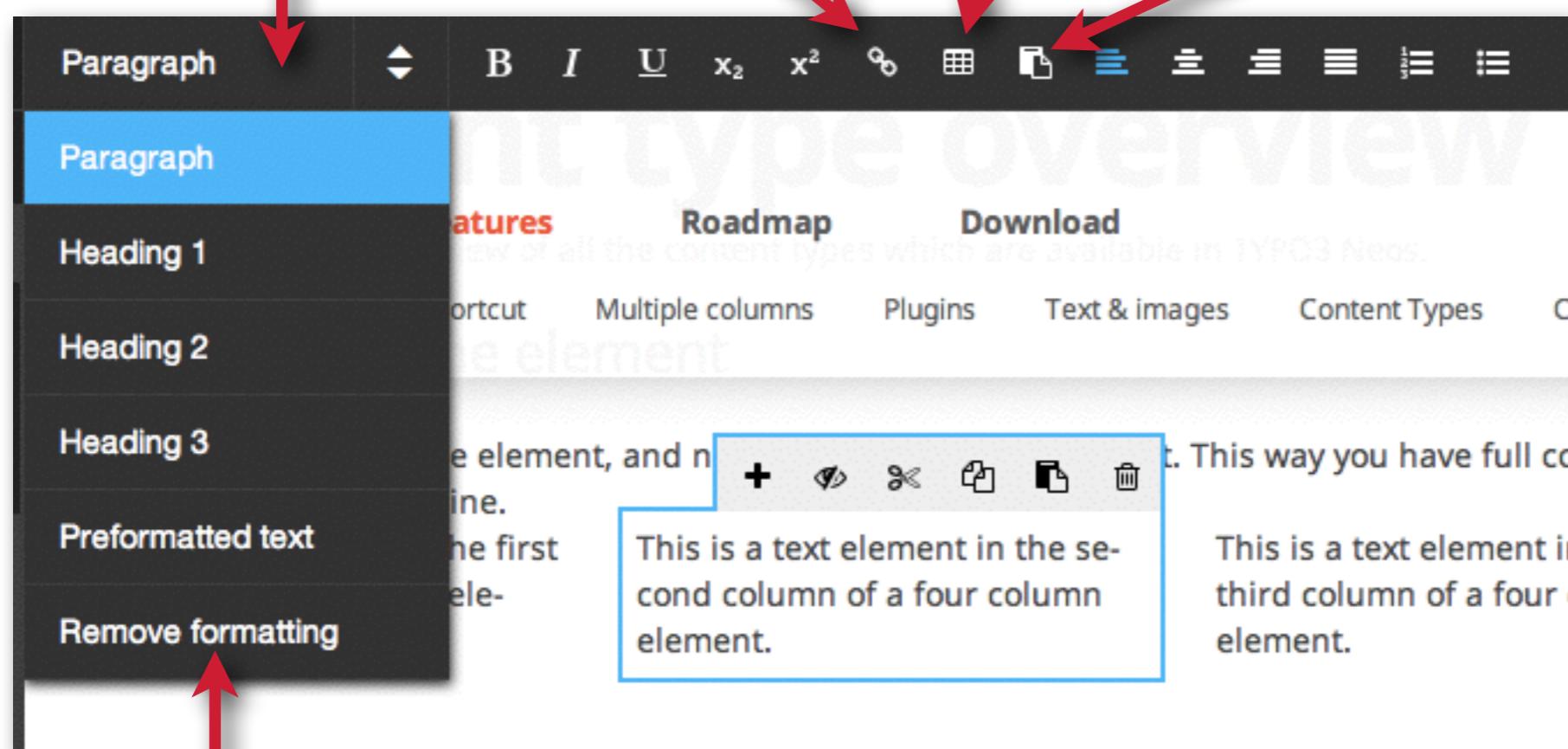
Absatzformate

Paragraph

Link
einfügen

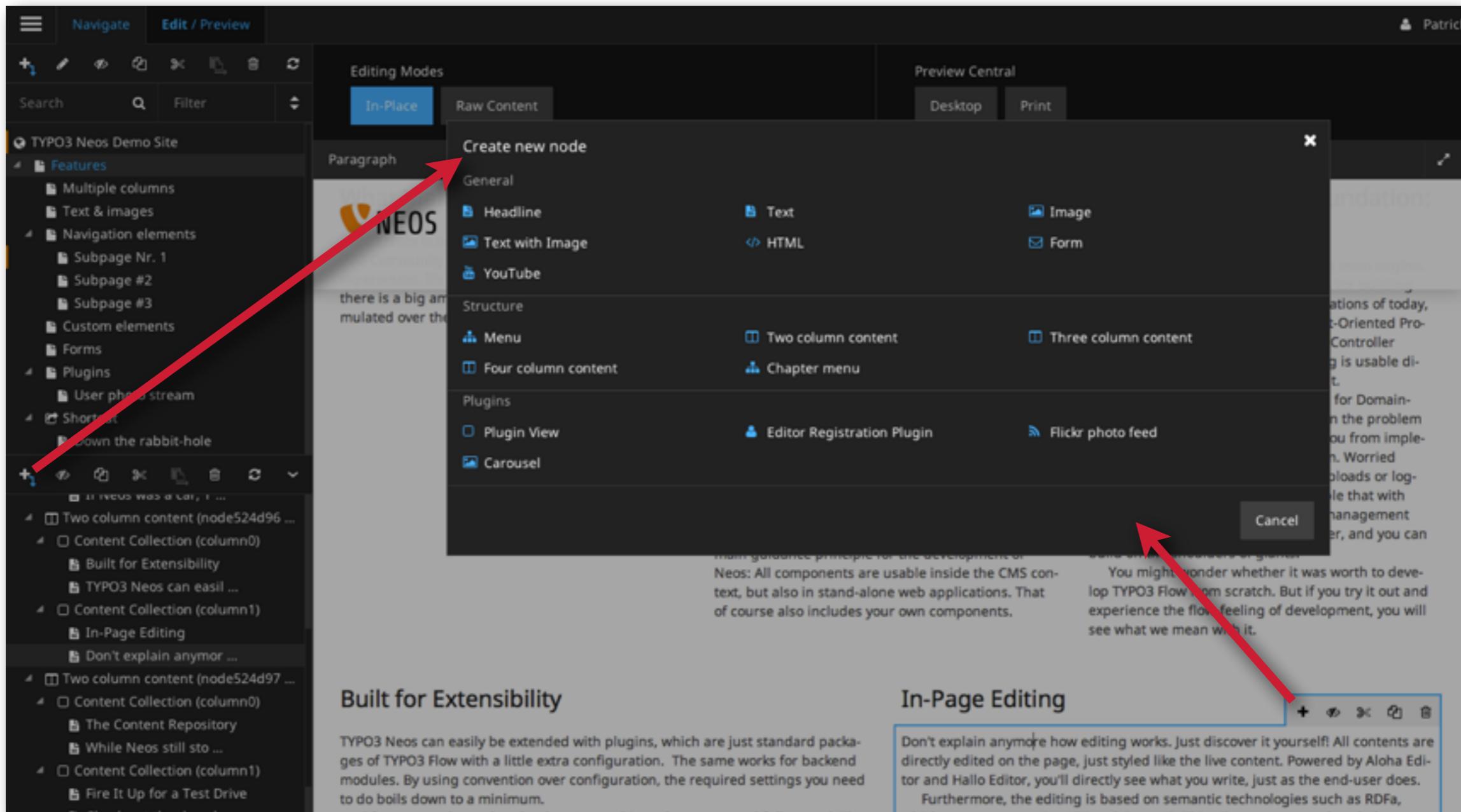
Tabelle
einfügen

Formatfrei
einfügen



Formatierung entfernen

Content Element einfügen



Mehrspaltige Content Elemente

The screenshot shows the TYPO3 Neos content creation interface. A modal window titled "Create new node" is open, listing various content elements under categories: General, Structure, and Plugins. The "General" category includes Headline, Text, Image, and Form. The "Structure" category includes Menu, Two column content, Three column content, Four column content, Chapter menu, Editor Registration Plugin, and Carousel. The "Plugins" category includes Plugin View. In the main content area, there are two columns of text. The left column has a red box around its top section, and a red arrow points from the "Two column content" option in the modal to this red box. The right column also has a red box around its top section, and another red arrow points from the "Two column content" option in the modal to this red box.

In only a few years from now, the world will have entirely new cities. Cities where development has not even started. Cities that can draw from all we know about liveability and functionality.

The same goes for websites. Major websites will be built with the tools we're constructing today. But we don't know how they will be like. Just think of how different sites were 3 years ago. For both cities and websites, one basic rule counts: Things are constantly changing and can never be fully anticipated.

The inspiration from urban planning permeates the UX master plan, giving us crucial guidelines for structuring everything we know - and making room for all the future change we don't know.

Imagine this...

Power when you need it

The distribution of power in a city makes sure that everyone gets what they need. You don't power a shed in the way you power a nuclear plant. The ultra-simple editing already available in the alpha makes editors center their attention on content. If you need to add SEO data next to the content, Neos has an inspector that opens up for that kind of functionality. Just a bit more power, one step at a time...

If you need more, the inspector will be

Formulare auswählen

The screenshot shows the TYPO3 Neos backend interface. On the left, there is a contact form with fields for Name, Email, and Message, and a Send button. On the right, there is a sidebar titled "Selected element" which is currently set to "Form". The sidebar includes sections for "Visibility" (with "Hide before" and "Hide after" options) and "Form" (with "Form identifier" set to "Contact form"). A red arrow points from the "Contact form" field in the sidebar back to the "Name" field in the contact form on the left.

NEOS Features Roadmap Download

Imagine this...

In only a few years from now, the world will have entirely new cities. Cities where development has not even started. Cities that can draw from all we know about liveability and functionality.

The same goes for websites. Major websites will be built with the tools we're constructing today. But we don't know how they will be like. Just think of how different sites were 3 years ago. For both cities and websites, one basic rule counts: Things are constantly changing and can never be fully anticipated.

The inspiration from urban planning permeates our UX master plan, giving us crucial guidelines for structuring everything we know - and making room for all the future change we c

Contact

Name*

Email*

Message*

Your Message

Send

Selected element

Form

Visibility

Hide before

No date set

Hide after

No date set

Hide

Form

Form identifier

Contact form

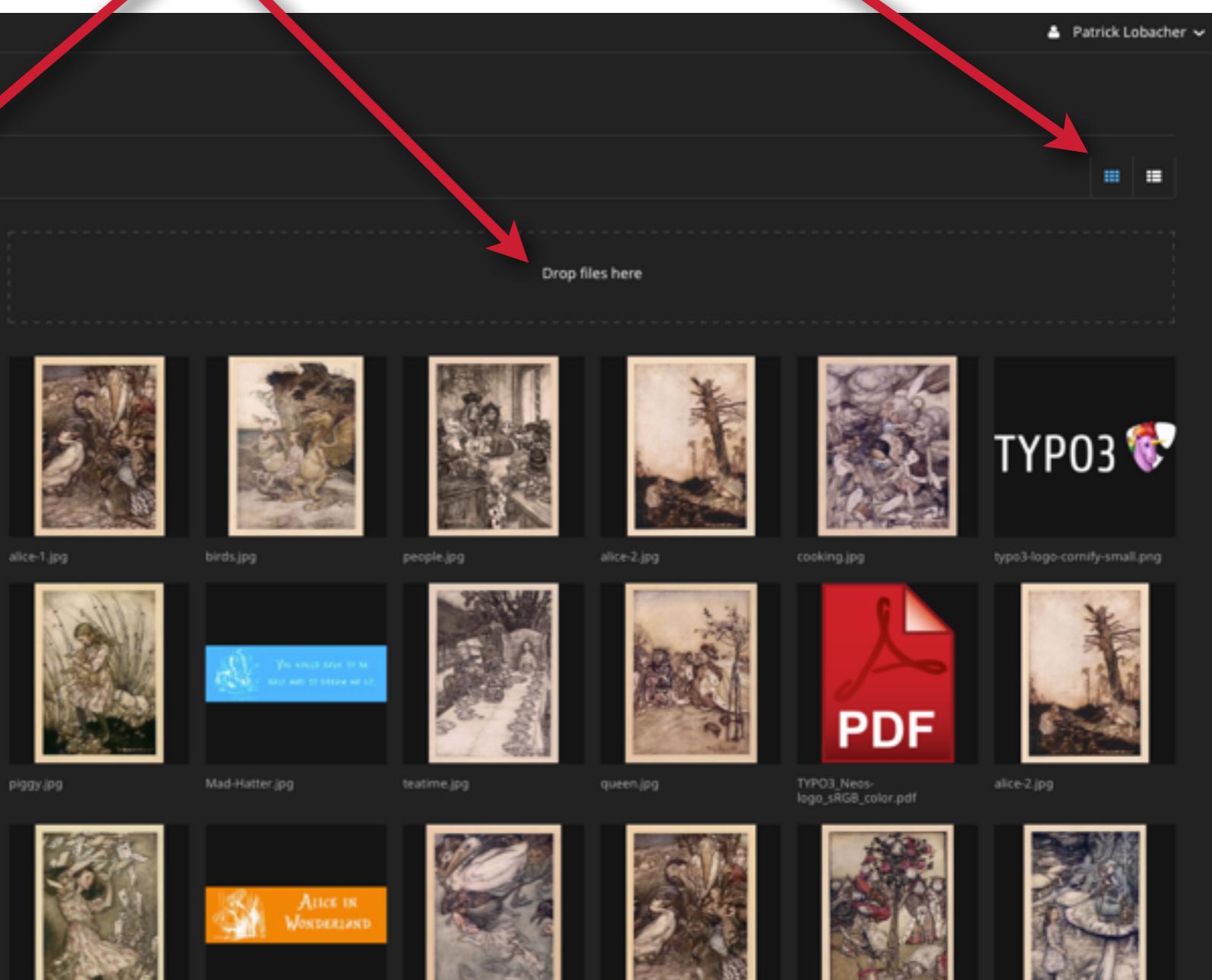
Contact form

Media Browser

Tagging

Zum Taggen wird
die Datei per
Drag&Drop auf das
Tag „gezogen“

Upload



Ansichten

Media Browser

The screenshot shows the TYPO3 Neos Media Browser interface. On the left, there is a sidebar with the title "Management / Media". Below it, there are two sections: "Meta-Daten" and "Aktionen". A red arrow points from the "Meta-Daten" label to the "Caption" field in the sidebar. Another red arrow points from the "Aktionen" label to the "Save" button at the bottom of the screen. The main content area displays a file named "typo3-logo-cornify-small.png" with a last modified date of "2013-12-11 08:37:51 (resource)". To the right of the file information is a preview window showing the logo. The logo features the word "TYPO3" in white, bold letters, with a small purple unicorn icon next to it.

Management / Media

Meta-Daten

Aktionen

Title

Caption

Preview

Filename: typo3-logo-cornify-small.png

Last modified (resource): 2013-12-11 08:37:51

Cancel Delete Save

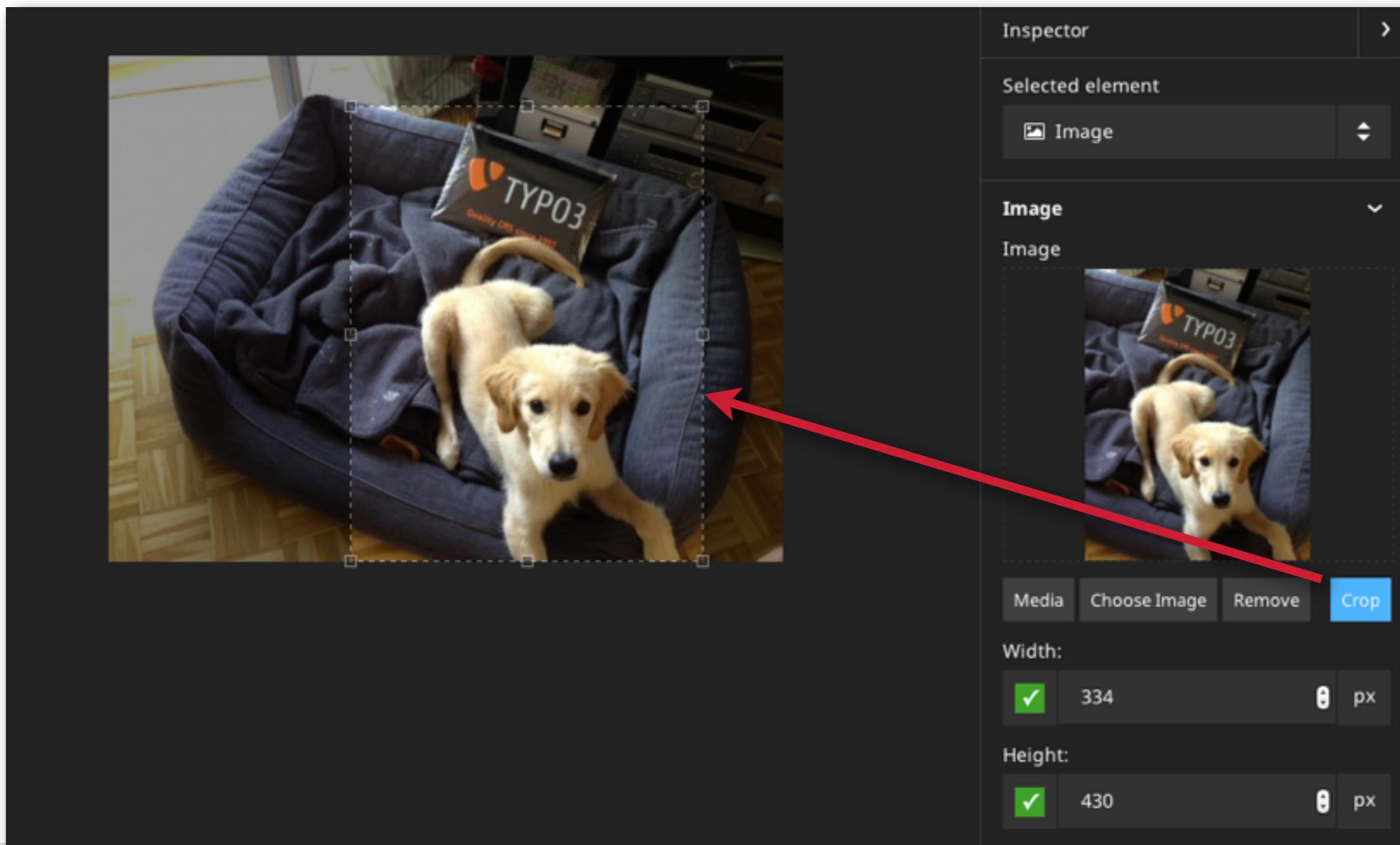
Image Handling

The image shows a workflow for managing images in TYPO3 Neos:

- Create new node:** A modal window titled "Create new node" is open. It contains sections for "General" (Headline, Text with Image, YouTube), "Structure" (Menu, Two column content, Three column content, Four column content, Chapter menu), "Plugins" (Plugin View, Editor Registration Plugin, Flickr photo feed), and "Carousel". On the right side of the modal, there is a preview area showing a placeholder image and some text.
- Media-Browser:** A separate window titled "Media-Browser" is shown. It displays a list of media items: "Image" (selected) and "Form". Below the list is a preview area showing a small thumbnail of an image. The "Selected element" dropdown also shows "Image".
- Upload:** A third window titled "Upload" is shown. It displays a preview area showing a larger version of the same image from the Media-Browser. Below the preview are buttons for "Media", "Choose Image", and "Remove".
- Crop:** A fourth window titled "Crop" is shown. It displays a preview area showing a cropped version of the image. Below the preview are buttons for "Media", "Choose Image", "Remove", and "Crop". The "Width" and "Height" fields are set to 640 and 480 respectively. The "Alignment" dropdown is set to "Default".

Red arrows indicate the flow from the "Image" item in the Media-Browser to the "Image" item in the Upload window, and finally to the "Image" item in the Crop window.

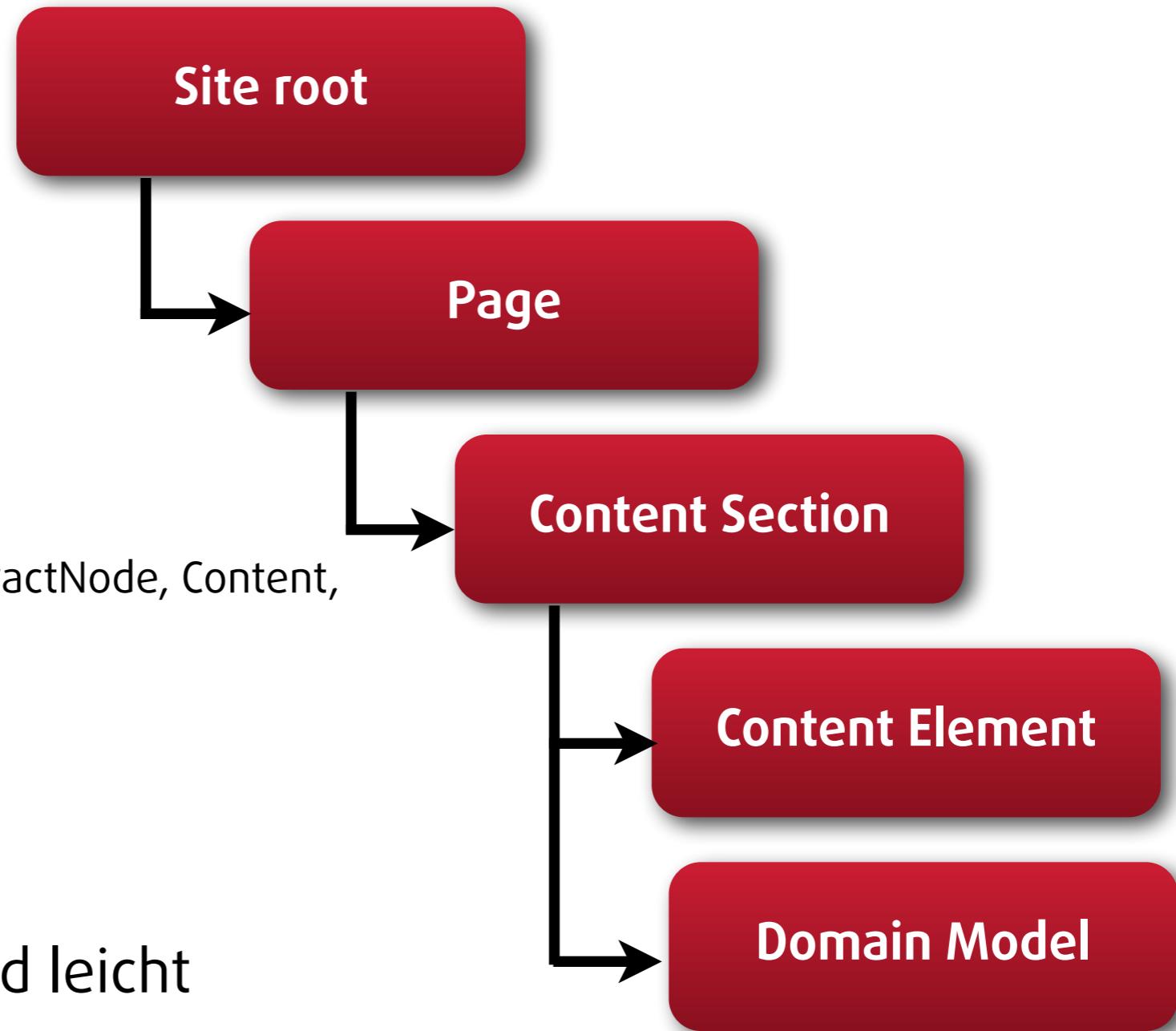
Cropping (unobstrusive)



Internas von **TYPO3 Neos**

Node Structure

- **TYPO3CR Node**
 - Node Name
(dient der Identifikation)
 - Node Type
(z.B. Document, Folder, AbstractNode, Content, ContentCollection...)
 - Properties
(Abhängig vom NodeType)
- Einfach erweiterbar und leicht konfigurierbar



Node Type

- Jede TYPO3CR Node hat einen sogenannten Node Type
- Node Types können in jedem Package definiert werden, indem Sie in der Datei Configuration/NodeTypes.yaml deklariert werden.
- Jeder Node Type kann einen oder mehrere Eltern Typen haben. Wenn diese spezifiziert sind, werden alle Eigenschaften und Settings der Eltern Types vererbt

```
'My.Package:SpecialHeadline':  
  superTypes: [ 'TYPO3.Neos:Content' ]  
  ui:  
    label: 'Special Headline'  
    group: 'general'  
  properties:  
    headline:  
      type: 'string'  
      defaultValue: 'My Headline Default'  
      ui:  
        inlineEditable: TRUE  
    validation:  
      'TYPO3.Neos/Validation/StringLengthValidator':  
        minimum: 1  
        maximum: 255
```

<http://docs.typo3.org/neos/TYPO3NeosDocumentation/IntegratorGuide/ContentStructure.html#nodes-inside-the-typo3-content-repository>

Vordefinierte Node Types:

- **TYPO3.Neos:Node**

Das ist der Basis-Typ, der von allen Content-Typen erweitert werden sollte, welche im Kontext von TYPO3 Neos verwendet werden. Hier gibt es keinerlei Eigenschaften.

- **TYPO3.Neos:Document**

Eine wichtige Unterscheidung wird zwischen Nodes getroffen, die sich wie eine Seite verhalten (und so aussehen) und zwischen „normalen Inhalt“, wie Text, der auf einer Seite gerendert wird. Nodes, die sich wie Seiten verhalten, werden in Neos als „Document Nodes“ bezeichnet. Diese haben beispielsweise eine einzigartige und extern sichtbare URL in Neos. Die Standard-Seite innerhalb von Neos ist implementiert durch den Node Typen **TYPO3.Neos.NodeTypes:Page** welcher direkt von **TYPO3.Neos:Document** abgeleitet wird.

Vordefinierte Node Types:

- **TYPO3.Neos:ContentCollection und TYPO3.Neos:Content**

Jeglicher Content, der sich wie eine Seite verhält, sich aber innerhalb einer solchen befindet, wird durch zwei verschiedene Node Typen implementiert:

- **TYPO3.Neos:ContentCollection type**

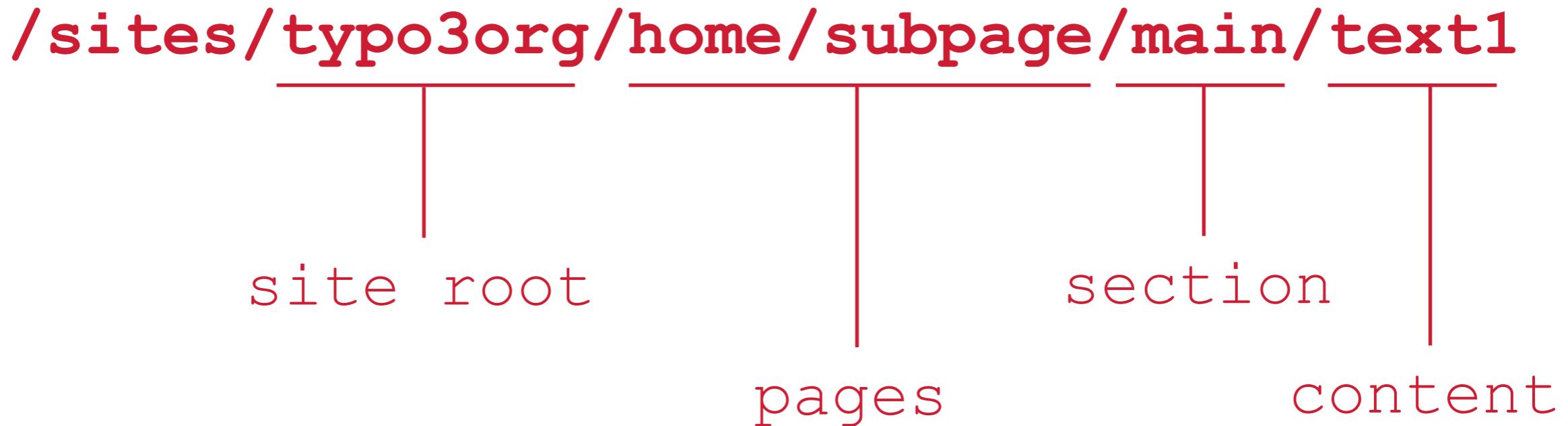
Dieser Node Typ dient der Strukturierung. Normalerweise enthält er keine Eigenschaften sondern eine geordnete Liste von Sub-Nodes, welche innerhalb des Nodes gerendert werden. Momentan sollte von dieser Typ nicht abgeleitet werden, wenn man eigene Node Types erstellt

- **TYPO3.Neos:Content**

Dies ist der Node Typ für alle Standard Elemente, wie „text“, „image“, „youtube“, ...

Node Zugriff

- Der Zugriff auf Nodes erfolgt durch „NodePaths“
- Jede Seite ist ein Node, jedes Element auf einer Seite ebenfalls



TypoScript

- TypoScript ist eine hierarchische, objektorientierte und Prototypen-basierte Verarbeitungssprache
- Wird von Neos verwendet, um den Content flexibel zu rendern
- Objekte sind z.B. Array, Collection, Case, Menu, Page, Template, Plugin, Value, ...
- Objekte haben Eigenschaften, die das Objekt „konfigurieren“
- TypoScript hat Zugriff auf den jeweiligen „Context“ (z.B. Seitenbaum im Objekt „menu“ oder Seiteneigenschaften im Objekt „page“)
- Es gibt „Prozessoren“ die die Eigenschaftswerte verändern können (ähnlich stdWrap-Funktionen in TYPO3 CMS)

TypoScript: Fluid-Template

```
{namespace ts=TYPO3\TypoScript\ViewHelpers}
{namespace bootstrap=TYPO3\Twitter\Bootstrap\ViewHelpers}
<!DOCTYPE html>
<html lang="en">
    <head>
        <f:section name="metadata">
            <meta name="viewport" content="width=device-width, initial-scale=1.0">
        </f:section>
        <f:section name="stylesheets">
            <link rel="stylesheet" href="{f:uri.resource(path: '3/css/bootstrap.min.css', package: 'TYPO3.Twitter.Bootstrap')} media="all" />
        </f:section>
    </head>
    <body>
        <f:section name="body">
            <div class="top-navigation-wrap">
                <div class="container">
                    {parts.mainMenu -> f:format.raw() }
                    {parts.secondLevelMenu -> f:format.raw() }
                </div>
            </div>
            ...
        </f:section>
    </body>
</html>
```

TypoScript: Beispiel - Teil 1

```
include: NodeTypes/Carousel.ts2
namespace: TypoScript=TYPO3.TypoScript
page = Page {
    head {
        stylesheets {
            site = TypoScript:Template {
                templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/Page/Default.html'
                sectionName = 'stylesheets'
                node = ${node}
            }
        }
        metadata = TypoScript:Template {
            templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/Page/Default.html'
            sectionName = 'metadata'
        }
    titleTag {
        // Overwrite the title tags content with a collection to create a breadcrumb
        content = TYPO3.TypoScript:Collection {
            // Retrieve all parent document nodes excluding the homepage
            collection = ${q(documentNode).add(q(documentNode).parents()).slice(0, -1).get()}
            itemName = 'node'
            iterationName = 'nodeIterator'
            // Implode node titles with a dash
            itemRenderer = ${q(node).property('title') + (nodeIterator.isLast ? '' : ' - ')}
            // Always add general site name as suffix
            @process.siteName = ${(value ? value + ' - ' : '') + 'TYPO3 Neos'}
        }
    }
}
```

TypoScript: Beispiel - Teil 2

```
bodyTag.attributes.class = ${q(node).parents().count() >= 1 && q(node).children('[instanceof TYPO3.Neos:Document]').filter('_hiddenInIndex=false').count() > 0 ? 'has-subpages' : ''}

body {
    templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/Page/Default.html'
    sectionName = 'body'

    parts {
        mainMenu = Menu {
            entryLevel = 1
            templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/TypoScriptObjects/MainMenu.html'
            maximumLevels = 3
            itemCollection = ${q(site).add(q(site).children('[instanceof TYPO3.Neos:Document'])).get()}
        }

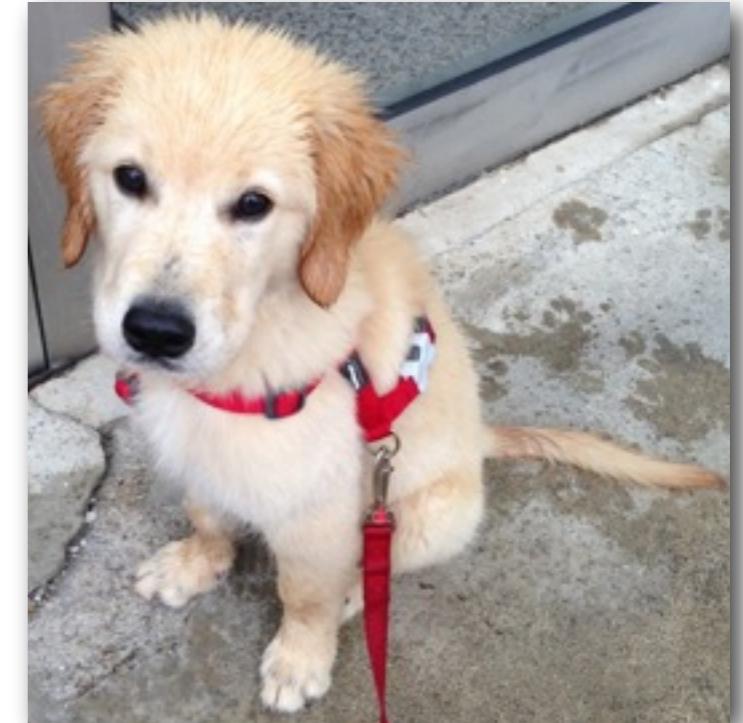
        secondLevelMenu = Menu {
            entryLevel = 2
            templatePath = 'resource://TYPO3.NeosDemoTypo3Org/Private/Templates/TypoScriptObjects/SecondLevelMenu.html'
            maximumLevels = 1
        }
    }
}
```

TypoScript: Prozessoren

- Beispiel:

```
myObject = MyObject {
    property = 'Rocky'
    property.@process.1 = ${'Der ' + value + ' ist der Beste!' }
}

# Resultat ist 'Der Rocky ist der Beste!'
```



- Kann auch so geschrieben werden:

```
myObject = MyObject {
    property = 'some value'
    property.@process.someWrap {
        expression = ${'before ' + value + ' after'}
        @position = 'start'
    }
}
```

- Prozessoren sind Eel Ausdrücke oder TypoScript Objekte, welche auf den Wert des Kontextes angewendet werden.
Auf das aktuelle Objekte kann mittels `this` zugegriffen werden

Eel - Embedded Expression Language

- Während TypoScript Zuweisungen und Prozessoren beinhaltet, kann man mit Eel Ausdrücke der Art `myObject.foo = ${q(node).property('bar')}` formulieren
- Die Embedded Expression Language (Eel) ist ein Baustein um Domain Specific Languages (DSL) zu erstellen.
- Eel stellt eine reichhaltige Syntax zur Verfügung um beliebige Ausdrücke zu erstellen, damit sich der Autor der DSL auf die Semantik konzentrieren kann

```
 ${foo.bar}                                // Traversal
 ${foo.bar()}                               // Methoden-Aufruf
 ${foo.bar().baz()}                          // Verkettter Methoden-Aufruf

 ${foo.bar("arg1", true, 42)}              // Methoden-Aufruf mit Argumenten
 ${12 + 18.5}                             // Kalkulation
 ${foo == bar}                            // Vergleiche

 ${foo.bar(12+7, foo == bar)}             // Alles kombiniert
 ${[foo, bar]}                           // Array Literal
 ${{foo: bar, baz: test}}                // Object Literal
```

FlowQuery

- FlowQuery stellt eine Art jQuery für TYPO3 Flow dar
- FlowQuery stellt damit einen Weg dar, um Content (der ja eine TYPO3CR Node in Neos ist) im Eel zu verarbeiten
- In FlowQuery gibt es Operationen:
 - **property**
Zugriff auf alle Eigenschaften einer Node
 - **filter**
Filterausdrücke in „Fizzle“
 - **children**
Gibt alle Kinder der TYPO3CR Node zurück
 - **parents**
Gibt alle Eltern der TYPO3CR Node zurück
 - **Weitere Operationen: add, count, first, get, is, last, ...**

FlowQuery - Beispiele

- Anzahl der Kommentare = Anzahl der Kinder der aktuellen Node mit dem Namen „comments“, deren Eigenschaft „spam“ auf dem Wert „false“ steht.

```
numberOfComments = ${q(node).children('comments').children("[spam = false]").count()}
```

- Breadcrumb-Menü = Aktuelle Seite + alle Elternseiten

```
prototype(TYPO3.Neos:Breadcrumb) < prototype(TYPO3.TypoScript:Template) {
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/BreadcrumbMenu.html'
    node = ${node}
    items = ${q(node).add(q(node).parents('[instanceof TYPO3.Neos:Document]'))}
}

{namespace neos=TYPO3\Neos\ViewHelpers}
<f:if condition="{items}">
    <ul class="breadcrumbs">
        <f:for each="{items}" as="item" reverse="TRUE">
            <f:if condition="{item.hiddenInIndex} == 0">
                <li>
                    <neos:link.node node="{item}">{item.label}</neos:link.node>
                </li>
            </f:if>
        </f:for>
    </ul>

```

Technische Details

TYPOScript

TypoScript 2.0 - Übersicht

- Nachdem bereits in TYPO3 CMS TypoScript verwendet wurde, bezeichnet man TypoScript in TYPO3 Neos (bzw. in TYPO3 Flow) mit der Versionsnummer 2.0
- Letztlich haben beide Versionen (vor allem technisch) nicht allzuviel miteinander zu tun
- TypoScript wird ausschließlich zum Rendern von Inhalten im Frontend verwendet (keine Konfiguration mehr von Usern im Backend beispielsweise wie durch TSconfig in TYPO3 CMS üblich)
- TypoScript ist hierarchisch, weil es auch hierarchischen Inhalt rendert
- TypoScript ist Prototypen-basiert (ähnlich wie JavaScript), da es beispielsweise erlaubt die Eigenschaften von allen Instanzen gleichzeitig zu ändern
- TypoScript ist eine Verarbeitungssprache, da es die Werte in einem Kontext verarbeitet und in einem einzigen Ausgabe-Wert überführt

TypoScript: Objekte

- TypoScript ist eine Sprache um **TypoScript Objekte** zu beschreiben
- Ein TypoScript Objekt besitzt Eigenschaften - sogenannte **Properties**
- TypoScript Objekte haben Zugriff zu einem „Kontext“, welcher letztlich eine Liste von Variablen ist
- TypoScript überführt diesen Kontext mit Hilfe der Properties in eine Ausgabe
- Intern kann TypoScript auch diesen Kontext verändern, sowie das Rendern von „verschachtelten“ Objekten (TypoScript-Tree) anstoßen
- TypoScript-Objekte werden durch PHP-Klassen realisiert, welche zur Laufzeit instanziert werden. Dabei kann eine Klasse die Basis von mehreren verschiedenen Objekten sein

TypoScript: Objekte

- Gültiges TypoScript wäre beispielsweise wie folgt
- Dabei werden TypoScript-Pfade immer in lowerCamelCase notiert und Objekte (Prototypen) in UpperCamelCase

```
foo = Page
my.object = Text
my.image = TYPO3.Neos.ContentTypes:Image
```

- Wertzuweisungen

```
foo.myProperty1 = 'Some Property which Page can access'
my.object.myProperty1 = "Some other property"
my.image.width = ${q(node).property('foo')}
```

- Werte die Strings darstellen müssen mit Anführungszeichen umschlossen werden (einzelne oder doppelte). Als Werte sind auch Eel-Ausdrücke zugelassen.

TypeScript: Syntax

- Man kann TypeScript auch mittels geschweiften Klammern notieren („ausklammern“)
 - dies bedeutet, dass der Pfad vor der öffnenden Klammer immer allen Pfaden innerhalb der Klammern vorangestellt wird.

```
my {  
    image = Image  
    image.width = 200  
    object {  
        myProperty1 = 'some property'  
    }  
}
```

- Dies ist identisch mit dem folgenden Code

```
my.image = Image  
my.image.width = 200  
my.object.myProperty1 = 'some property'
```

TypeScript: Objekt Instanziierung

- Es ist zudem möglich, Werte direkt bei der Instanziierung zu vergeben, wie das dritte Beispiel zeigt (alle Beispiele sind in der Wirkung identisch):

```
someImage = Image
someImage.foo = 'bar'
```

```
someImage = Image
someImage {
    foo = 'bar'
}
```

```
someImage = Image {
    foo = 'bar'
}
```

TypeScript Objekte sind frei von Seiteneffekten

- Obwohl TypeScript Objekte ihren Kontext verändern können, sind sie ohne „Seiteneffekte“.
- Dafür wird der Kontext nach der Benutzung eines TypeScript-Objektes wieder „aufgeräumt“, auch wenn dieser vorher verändert wurde
- TypeScript Objekte können nur andere TypeScript-Objekte ändern, die verschachtelt sind - aber nicht Objekte welche davor oder danach kommen
- Das sorgt dafür, dass sich ein TypeScript-Pfad zusammen mit seinem Kontext immer gleich verhält - egal an welcher Stelle dieser aufgerufen wird

TypeScript: Prototypen

- Wenn ein TypeScript Objekt instanziert wird (weil z.B. jemand `someImage = Image` schreibt), dann wird der Prototyp für dieses Objekt kopiert und als Basis für dieses neue Objekt (`someImage`) verwendet.
- Ein Prototyp wird wie folgt definiert:

```
prototype(MyImage) {  
    width = '500px'  
    height = '600px'  
}
```

- Nun kann man das Objekt verwenden:

```
# Das Objekt someImage hat eine Breite von 500px und eine Höhe  
von 600 px
```

```
someImage = MyImage
```

```
# Nun ist die Breite 100px (und die Höhe nach wie vor 600px)
```

```
someImage.width = '100px'
```

TypeScript: Prototypen

- Prototypen sind veränderbar:

```
prototype(MyYouTube) {  
    width = '100px'  
    height = '500px'  
}
```

```
# Hiermit wird die Breite für alle Instanzen  
# auf 400px geändert  
prototype(MyYouTube).width = '400px'
```

```
# Man kann auch neue Eigenschaften (Properties) definieren  
prototype(MyYouTube).showFullScreen = ${true}
```

```
# Man kann einen Prototypen auch „vererben“  
prototype(MyImage) < prototype(TYPO3.Neos:Content)
```

TypeScript: Prototypen

- Über Vererbung bleiben Prototypen aneinander „gebunden“. Ändert sich eine Eigenschaft in einer Instanz, wird diese auch in der anderen geändert.

```
prototype(TYPO3.Neos.Content).fruit = 'apple'  
prototype(TYPO3.Neos.Content).meal = 'dinner'
```

```
# MyImage hat nun auch die Eigenschaften "fruit = apple" and "meal = dinner"  
prototype(MyImage) < prototype(TYPO3.Neos:Content)
```

```
# Da MyImage das Objekt Template *erweitert*,  
# hat MyImage.fruit nun ebenfalls den Wert 'Banana'  
prototype(TYPO3.Neos:Content).fruit = 'Banana'
```

- # Da die Eigenschaft „meal“ nun in der Kind-Klasse überschrieben wurde,
hat ein Überschreiben in der Eltern-Klasse keine Auswirkung mehr
prototype(MyImage).meal = 'breakfast'
prototype(TYPO3.Neos:Content).meal = 'supper'

TypeScript: Prototypen

- Prototypen-Vererbung ist nur auf globaler Ebene möglich

```
prototype(Foo) < prototype(Bar)
```

- Nicht möglich wären daher folgende Anweisungen

```
prototype(Foo) < some.prototype(Bar)
other.prototype(Foo) < prototype(Bar)
prototype(Foo).prototype(Bar) < prototype(Baz)
```

TypoScript: Prototypen

- Hierarchische TypoScript Prototypen

```
# Setze die Eigenschaft „bar“ (bzw. „some.thing“) für alle Objekte vom Typ „Foo“  
prototype(Foo).bar = 'baz'  
prototype(Foo).some.thing = 'baz2'  
  
# Für alle Objekte vom Typ „Foo“, welche innerhalb des Pfades „some.path“ auftauchen, setze die Eigenschaft „some“  
some.path.prototype(Foo).some = 'baz2'  
  
# Für alle Objekte vom Typ „Bar“ die innerhalb von Objekten vom Typ „Foo“ auftauchen, setze die Eigenschaft „some“  
prototype(Foo).prototype(Bar).some = 'baz2'  
  
# Kombination aus allen Möglichkeiten zuvor  
prototype(Foo).left.prototype(Bar).some = 'baz2'
```

TypeScript: Namespaces

- Namespaces können bei der Deklaration verwendet werden

```
# Definiert einen Namespace „Acme.Demo“ für den Prototyp „YouTube“  
prototype(Acme.Demo:YouTube) {  
    width = '100px'  
    height = '500px'  
}
```

- Der Namespace ist per Konvention der Package-Key, des Packages wo sich das TypeScript befindet
- Man kann voll qualifizierte Namespaces verwenden:

```
prototype(TYPO3.Neos:ContentCollection.Default) <  
prototype(TYPO3.Neos:Collection)
```

TypeScript: Namespaces

- Verwendet man keinen Namespace, wird per Default immer `TYPO3.Neos` verwendet.
- Man kann aber auch eine Namespace-Direktive verwenden um einen eigenen Namespace zu verwenden:

```
namespace Foo = Acme.Demo
```

Die folgenden beiden Anweisungen sind identisch

```
video = Acme.Demo:YouTube
video = Foo:YouTube
```

TypeScript: Eigenschaften

- Auch wenn TypeScript-Objekte direkt auf den Kontext zugreifen können, sollte man dennoch Eigenschaften (Properties) dafür verwenden

```
# Wir nehmen an, dass es eine Eigenschaft "foo=bar"  
# im aktuellen Kontext an dieser Stelle gibt  
myObject = MyObject
```

```
# Explizites Zuweisen des Wertes der Variable „foo“  
# aus dem aktuellen Kontext und Zuweisen an die  
# „foo“ Eigenschaft des Objektes „myObject“  
myObject.foo = ${foo}
```

- Objekte sollten ausschließlich eigene Eigenschaften verwenden, um den Output zu generieren
- Lediglich bei der Prototypen-Definition kann man direkt auf den Kontext zugreifen:
prototype(MyObject).foo = \${foo}

TypoScript: Manipulation des Kontext

- Der TypoScript-Kontext kann direkt manipuliert werden, indem man die `@override` Meta-Eigenschaft verwendet:

```
myObject = MyObject
myObject.@override.bar = ${foo * 2}
```

- Der obige Code erzeugt nun eine weitere Kontext-Variable mit dem Namen `bar` und dem doppelten Wert von `foo`.

TypoScript: Prozessoren

- Prozessoren erlauben es, TypoScript Eigenschaften zu manipulieren:

```
myObject = MyObject {
    property = 'some value'
    property.@process.1 = ${'before ' + value + ' after'}
}
# Das Ergebnis ist nun 'before some value after'
```

- Es können mehrere Prozessoren verwendet werden
- Die Reihenfolge ergibt sich aus der numerische Position im TypoScript (nach @process) - ein @process.2 würde demnach nach @process.1 ablaufen

TypeScript: Prozessoren

- Man kann für Prozessoren auf eine erweiterte Syntax verwenden

```
myObject = MyObject {
    property = 'some value'
    property.@process.someWrap {
        expression = ${'before ' + value + ' after'}
        @position = 'start'
    }
}
# Das Ergebnis ist nun 'before some value after'
```

- Damit kann man einen Namen (hier „someWrap“) verwenden
- Prozessoren sind Eel Ausdrücke bzw. TypeScript Objekte, die auf Wert des Kontextes angewendet werden. Das aktuelle Objekt erreicht man via `this`

Neos TypoScript Referenz

TYPO3.TypoScript:Array (Teil 1)

- Rendert die verschachtelten TypoScript Objekte und verbindet deren Ausgabe
- Die Reihenfolge wird über das **@position** Argument spezifiziert
- Für dieses Argument gilt die folgende Reihenfolge
 - **start [priority]**
Je höher die Priorität, desto früher wird das Objekt hinzugefügt. Wenn keine Priorität angegeben wird, wird das Element hinter allen Elementen mit Priorität eingesortiert
 - **[numeric ordering]**
Position (aufsteigend sortiert)
 - **end [priority]**
Je höher die Priorität, desto später wird das Objekt hinzugefügt. Wenn keine Priorität angegeben wird, wird das Element vor allen Elementen mit Priorität eingesortiert

TYPO3.TypoScript:Array (Teil 2)

- Zusätzlich kann spezifiziert werden, dass ein Element vor oder nach einem anderen Element eingesortiert wird:
- **before [namedElement] [optionalPriority]**
Fügt dieses Element vor dem Element „namedElement“ ein. Gibt es mehrere Statements, so entscheidet die Priorität wie weit das Element vor „namedElement“ positioniert wird; es gilt also je höher, desto näher. Statements ohne Priorität werden am weitesten vor dem Element platziert. Wenn „namedElement“ nicht existiert, wird das Element nach allen „start“ Positionen platziert.
- **after [namedElement] [optionalPriority]**
Analog „before“ nur genau andersherum („nach namendElement“, „vor allen end Positionen“,...)

TYPO3.TypoScript:Array (Teil 3) (o = Kleines Oh / „Order“)

```
myArray = TYPO3.TypoScript:Array {
    o1 = TYPO3.Neos.NodeTypes:Text
    o1.@position = 'start 12'
    o2 = TYPO3.Neos.NodeTypes:Text
    o2.@position = 'start 5'
    o2 = TYPO3.Neos.NodeTypes:Text
    o2.@position = 'start'

    o3 = TYPO3.Neos.NodeTypes:Text
    o3.@position = '10'
    o4 = TYPO3.Neos.NodeTypes:Text
    o4.@position = '20'

    o5 = TYPO3.Neos.NodeTypes:Text
    o5.@position = 'before o6'

    o6 = TYPO3.Neos.NodeTypes:Text
    o6.@position = 'end'
    o7 = TYPO3.Neos.NodeTypes:Text
    o7.@position = 'end 20'
    o8 = TYPO3.Neos.NodeTypes:Text
    o8.@position = 'end 30'

    o9 = TYPO3.Neos.NodeTypes:Text
    o9.@position = 'after o8'
}
```

TYPO3.TypoScript:Collection (Teil 1)

- Führt eine Schleife auf einer Array-ähnlichen „collection“ aus und rendert jedes Element mit Hilfe des itemRenderer

collection	(array/iterable, required)	Das Array oder das „iterable“ über welches iteriert wird
itemName	(string, required)	Der Name, unter dem jedes Element verfügbar gemacht wird
iterationName	(string)	Meta-Infos über die Iteration: index (beginnend bei 0), cycle (beginned bei 1), isFirst, isLast.
itemRenderer	(nested TypoScript object)	Dieses TypoScript Objekt wird einmal je Durchlauf aufgerufen und dessen Resultat verbunden.

TYPO3.TypoScript:Collection (Teil 2)

```
myCollection = TYPO3.TypoScript:Collection {
    collection = ${[1, 2, 3]}
    itemName = 'element'
    itemRenderer = TYPO3.TypoScript:Template
    itemRenderer.templatePath = '...'
    itemRenderer.element = ${element}
}
```

TYPO3.TypoScript:Case

- Verhält sich wie PHP's switch/case Befehl. Ist die Bedingung „condition“ wahr, so wird „myCase“ auf den spezifizierten Typ gesetzt. Als Fallback wird (ähnlich zu „default“ bei switch/case) der Typ unter „fallback“ gesetzt.

```
myCase = TYPO3.TypoScript:Case
myCase {
    someCondition {
        condition = ${... some eel expression evaluating to TRUE or
FALSE ... }
        type = 'MyNamespace:My.Special.Type'
    }
    fallback {
        condition = ${true}
        type = 'MyNamespace:My.Default.Type'
    }
}
```

TYPO3.TypoScript:Template (Teil 1)

- Rendert ein Fluid-Template, welches durch templatePath angegeben ist

templatePath	(string, required)	Das Template, angeben z.B. mit resource://
partialRootPath	(string)	Der Pfad, unter dem die Partials zu finden sind
layoutRootPath	(string)	Der Pfad, unter dem die Layouts zu finden sind
sectionName	(string)	Die Fluid Section <f:section> die gerendert werden soll (sofern es eine gibt)
[variablen]	(any)	Alle weiteren Strings, werden als Variable an den View weitergereicht

TYPO3.TypoScript:Template (Teil 2)

```
myTemplate = TYPO3.TypoScript:Template {
    templatePath = 'resource://My.Package/Private/path/to/
Template.html'
    someDataAvailableInsideFluid = 'my data'
}
```

TYPO3.TypoScript:Value

- Dies ist ein Wrapper für einen Wert

value	(mixed, required)	Der Wert
--------------	--------------------------	----------

```
myValue = Value {
    value = 'Hello World'
}
```

Technische Details

Eel

Eel - Embedded Expression Language

- Neben einfachen TypoScript-Zuweisungen wie `myObject.foo = 'bar'` ist es mittels Eel möglich, Ausdrücke wie`myObject.foo = ${q(node).property('bar')}` zu formulieren.
- Jeder Eel-Ausdruck wird mittels `${...}` notiert
- Eel ist JavaScript sehr ähnlich
- Eel unterstützt keine Variablen-Zuweisungen oder Kontrollstrukturen
- Eel unterstützt die üblichen JavaScript Operatoren für Arithmetik und Vergleiche
- Eel unerstüzt den Dreifach-Operator für Vergleiche: `<condition> ? <ifTrue> : <ifFalse>`
- Sobald Eel auf eine Objekt-Eigenschaft zugreift, wird der entsprechende Getter aufgerufen
- Objekt-Zugriff über die Offset-Notation wird wie folgt unterstützt: `foo['bar']`

Eel - Beispiele

- **Folgende Ausdrücke sind beispielsweise gültig:**
- Eel selbst definiert keine Funktion oder Variablen, sondern verwendet das Eel Context Array. Funktionen und Objekte, auf die man zugreifen will, können dort definiert werden.
- Daher eignet sich Eel perfekt als "domain-specific language construction kit", welches die Syntax, nicht aber die Semantik der DSL (Domain Specific Language) zur Verfügung stellt.
- Für Eel innerhalb von TypoScript ist die Sematik wie folgt definiert:
 - Alle Variablen des TypoScript Kontextes sind im Eel Kontext zugänglich
 - Die spezielle Variable **this** zeigt immer auf die aktuelle TypoScript Objekt Implementierung
 - Es gibt eine Funktion **q()**, die ihre Argumente in einem FlowQuery-Objekt einhüllt

Eel - Sematik

- **Folgende Ausdrücke sind beispielsweise gültig:**

```
 ${foo.bar}                                // Traversal
 ${foo.bar()}                               // Methoden-Aufruf
 ${foo.bar().baz()}                         // Verketter Methoden-Aufruf

 ${foo.bar("arg1", true, 42)}   // Methoden-Aufruf mit Argumenten
 ${12 + 18.5}                            // Kalkulation
 ${foo == bar}                            // Vergleiche

 ${foo.bar(12+7, foo == bar)}  // Alles kombiniert
 ${[foo, bar]}                           // Array Literal
 ${{foo: bar, baz: test}}    // Object Literal
```

Technische Details

FlowQuery

FlowQuery - jQuery für Flow

- FlowQuery stellt eine Art jQuery für Flow dar und wurde auch von jQuery stark inspiriert
- FlowQuery ist ein Weg um Inhalte (Content = TYPO3CR Node innerhalb von Neos) im Eel Kontext zu verarbeiten
- FlowQuery Operationen werden durch PHP-Klassen implementiert
- Für jede FlowQuery-Operation, muss das Package installiert werden, welches die Operation (Klasse) enthält
- Jedes Package kann seine eigenen FlowQuery Operationen hinzufügen
- Das TYPO3.Eel Package beinhaltet bereits ein Set an Basis-Operationen

FlowQuery - Beispiele

- Zugriff auf eine Kontext Variable

```
 ${myContextVariable}
```

- Zugriff auf den aktuellen Node

```
 ${node}
```

- Zugriff auf Node-Eigenschaften

```
 ${q(node).getProperty('foo')} // Möglich, aber nicht empfohlen
```

```
 ${q(node).property('foo')} // Besser: Benutzung von FlowQuery
```

- Zuweisung an eine Variable

```
 text = ${q(node).property('text')}
```

FlowQuery - Beispiele

- Ermittlung aller Eltern-Nodes einer Node

```
 ${q(node).parents()}
```

- Ermittlung der ersten Node innerhalb der Sub-Nodes mit dem Namen „left“

```
 ${q(node).children('left').first()}  
 ${q(node).children().filter('left').first()}
```

- Ermittlung aller Eltern Nodes und Hinzufügen des aktuellen Nodes

```
 ${node.parents().add(node)}
```

- Ermittlung der Anzahl aller Nodes, deren Kinden innerhalb von „comments“ die Eigenschaft „spam“ auf dem Wert „false“ haben:

```
 numberOfComments = ${q(node).children('comments').children("[spam = false]").count()}
```

FlowQuery - **add** Operation (TYPO3.Eel)

- Die add Operation fügt ein weiteres flowQuery Objekt zum aktuellen hinzu
- **Beispiel:**

```
items = ${q(node).add(q(node).parents())}
```

Hier wird die aktuelle Node ermittelt und dazu alle Eltern-Nodes hinzugefügt. Dies wird beispielsweise verwendet, um eine Breadcrumb-Navigation zu erstellen. Hierfür wird die Rootline der Nodes benötigt, die mit Hilfe des obigen Statements ermittelt wird

FlowQuery - **count** Operation (TYPO3.Eel)

- Die count Operation gibt die Anzahl der Objekte (oder Array-Items) zurück
- **Beispiel:**

```
test.value = ${q(node).add(q(node).parents()).count()}
```

Hier wird zunächst die Rootline ermittelt und dann mittels count() durchgezählt. Die Zahl wird zurückgegeben.

FlowQuery - **first** Operation (TYPO3.Eel)

- Die first Operation gibt das erste Objekt zurück
- **Beispiel:**

```
test.value = ${q(node).add(q(node).parents())
    .first().property('title')}
```

Hier wird zunächst die Rootline ermittelt und dann mittels first() das erste Element ausgewählt.
Von diesem wird der Titel zurückgegeben.

Wenn es zwei Seiten gibt: **Home > Kompendium**, dann wird nun „**Kompendium**“
zurückgegeben.

FlowQuery - **get** Operation (TYPO3.Eel)

- Wenn FlowQuery verwendet wird, ist das Ergebnis ebenfalls wieder FlowQuery. Die get Operation „befreit“ das Resultat vom „FlowQuery“
- **Beispiel:**

```
test.value = ${q(node).add(q(node).parents()).first().get(0)}
```

Hier wird zunächst die Rootline ermittelt und dann mittels first() das erste Element ausgewählt. Dieses wird nun mittels get() umgewandelt in ein Array (ohne wäre es ein FlowQuery Objekt) und dort wird das erste Element ausgewählt und zurückgegeben.

Wenn es zwei Seiten gibt: **Home > Kompendium**, dann wird folgendes zurückgegeben:

```
Node /sites/demo/homepage/Kompendium [TYPO3.Neos:Page]
```

FlowQuery - **is** Operation (TYPO3.Eel)

- Überprüft, ob mindestens eines der Elemente im Kontext einem gegebenen Filter entspricht
- **Beispiel:**

```
test.value = ${q(node).is(' instanceof TYPO3.Neos:Page ')
    ? 'Seite' : 'Keine Seite'}
```

Der if Operator überprüft zunächst, ob die aktuelle Node eine Instanz des Nodetypes TYPO3.Neos:Page (also eine normale Seite) ist. Wenn dies der Fall ist, wird der Wert „Seite“ zurückgegeben, ansonsten „Keine Seite“

FlowQuery - **last** Operation (TYPO3.Eel)

- Die last Operation gibt das letzte Objekt zurück
- **Beispiel:**

```
test.value = ${q(node).add(q(node).parents())
    .last().property('title')}
```

Hier wird zunächst die Rootline ermittelt und dann mittels last() das letzte Element ausgewählt. Von diesem wird der Titel zurückgegeben.

Wenn es zwei Seiten gibt: **Home > Kompendium**, dann wird nun „**Home**“ zurückgegeben.

FlowQuery - slice Operation (TYPO3.Eel)

- Wendet die Slice Operation auf ein Set von Ergebnissen an
- **Beispiel:**

```
 ${q(site).find('fr/blog').children('[instanceof  
Namespace.Plugin.Blog:Post]').slice() }  
 ${q(site).find('fr/blog').children('[instanceof  
Namespace.Plugin.Blog:Post]').slice(2) }  
 ${q(site).find('fr/blog').children('[instanceof  
Namespace.Plugin.Blog:Post]').slice(0, 2) }
```

Die beiden letzten Statements geben die beiden letzten Blog-Posts zurück

FlowQuery - **property** Operation (TYPO3.Eel)

- Die property Operation gibt die Eigenschaften eines Objekts zurück
- Wenn die Eigenschaft mit `_` beginnt, werden interne Werte ausgelesen, z.B.:
 - `_path` (Node Pfad)
 - `_nodeType.name` (Node Typ)
- **Beispiel:**

```
test.value = ${q(node).property('title')}
```

Gibt die Eigenschaft „title“ der aktuellen Node zurück.

FlowQuery - **children** Operation (TYPO3.Eel)

- Die children Operation gibt die Kind-Objekte des aktuellen Objekts zurück. Dies sind bei einer Seite beispielsweise die Sektionen und dort die Content-Bereiche.
- Es ist möglich eine Filter-Option anzugeben: ...children('main')...
- **Beispiel:**

```
test.value = ${q(node).children().property('_path')}
```

Ergebnis:

/sites/demo/homepage/Kompendium/kompendium1/main

FlowQuery - **filter** Operation (TYPO3.Eel)

- Die filter Operation limitiert das Ergebnis-Set der Objekte. Der Filter-Ausdruck wird mittels Fizzle notiert und unterstützt die folgenden Operatoren:
 - = (Gleichheit)
 - \$= (Wert endet mit dem Operanden)
 - ^= (Wert beginnt mit dem Operanden)
 - *= (Wert enthält den Operanden)
 - instanceof (Prüft ob der Wert eine Instanz des Operanden ist)
- **Beispiel:**

```
test.value = ${q(node).children().filter('main').  
first().property('_path')}
```

Gibt die Eigenschaft „_path“ der ersten Kind-Node im Pfad „main“ zurück.

Technische Details

Fizzle

Fizzle

- Filter Operationen (z.B. filter() in FlowQuery) werden in Fizzle notiert
- Property Name Filters
 - Hier ist nur ein String erlaubt, wie `foo` aber keine Pfade, wie `foo.bar` oder `foo.bar.baz`
- Attribut Filter
 - `baz[foo]`
 - `baz[answer = 42]`
 - `baz[foo = "Bar"]`
 - `baz[foo = 'Bar']`
 - `baz[foo ^= "Bar"]`
 - `baz[foo $= "Bar"]`
 - `baz[foo *= "Bar"]`
- Operatoren
 - `=` (Gleichheit)
 - `$=` (Wert endet mit dem Operator)
 - `^=` (Wert beginnt mit dem Operator)
 - `*=` (Wert enthält den Operator)
 - `instanceof` (Prüft ob der Wert eine Instanz des Operator ist)

Technische Details

NodeTypes.yaml

Aufbau der Datei: **NodeTypes . yaml**

- Jede TYPO3CR Node (im Folgenden schlicht Node genannt) hat einen spezifischen Node Type
- Node Types können in ihrem Package (bzw. in ihrer Site) innerhalb der Datei **Configuration/NodeTypes . yaml** konfiguriert werden
- Jede Node kann ein oder mehrere Eltern Typen besitzen. Ist diese gesetzt, werden alle Eigenschaften und Settings vererbt

```
'My.Package:SpecialHeadline':  
superTypes: [ 'TYPO3.Neos:Content' ]  
ui:  
  label: 'Special Headline'  
  group: 'general'  
properties:  
  headline:  
    type: 'string'  
    defaultValue: 'My Headline Default'  
    ui:  
      inlineEditable: TRUE  
validation:  
  'TYPO3.Neos/Validation/StringLengthValidator':  
    minimum: 1  
    maximum: 255
```

Node-Deklaration: Start-Deklaration

- Start-Deklaration einer Node:

'Namespace.PackageName:nodeName' :

bzw.

'Namespace.SiteName:nodeName' :

- Die Start-Deklaration kann folgende Eigenschaften haben:

- superTypes:
- childNodes:
- abstract:
- ui:
- properties:
- validation:

Node-Deklaration: **superTypes**

- Dies ist ein Array von Eltern-Nodes, von denen die aktuelle Node ableitet

'Namespace.PackageName:nodeName' :
bzw.

'Namespace.SiteName:nodeName' :

- **Beispiele:**

```
'My.Package:SpecialHeadline':  
    superTypes: ['TYPO3.Neos:Content']
```

```
'TYPO3.Neos:Document':  
    superTypes:  
        - 'TYPO3.Neos:Node'  
        - 'TYPO3.Neos:Hidable'
```

Node-Deklaration: **childNodes**

- Liste von Kind-Nodes, die automatisch erstellt werden, wenn ein Node dieses Typs angelegt wird. Für jeden Kind-Element muss der Node Type angegeben werden
- **Beispiele:**

```
childNodes:  
    someChild:  
        type: 'TYPO3.Neos:ContentCollection'  
  
'TYPO3.NeosDemoTypo3Org:Carousel':  
    superTypes: ['TYPO3.Neos:Content']  
    childNodes:  
        carouselItems:  
            type: 'TYPO3.Neos:ContentCollection'
```

Node-Deklaration: ui

- Konfigurations-Optionen für die UI-Repräsentation des Node Types
- **label:**
Das Label, welches angezeigt wird
- **group:**
Name der Gruppe im 'New Content Element' Dialog, in welche das Element einsortiert wird. Mögliche Werte sind (per Default): general, structure, plugins (Packages/Application/TYPO3.Neos/Configuration/Settings.yaml)
- **icon:**
Definiert das Icon - Auswahl ist beschränkt auf „Font Awesome“ (<http://docs.pagelines.com/tutorials/font-awesome>) - z.B. icon-align-center
- **inlineEditable:**
Gibt an, ob eine direkte Interaktion mit der Node möglich ist

```
nodeTypes:  
  groups:  
    general:  
      position: 'start'  
      label: 'General'  
    structure:  
      position: 100  
      label: 'Structure'  
    plugins:  
      position: 200  
      label: 'Plugins'
```

Node-Deklaration: ui

- Konfigurations-Optionen für die UI-Repräsentation des Node Types
- **inspector:**
Konfiguriert den Inspector in der NEOS UI für diesen Node Type
 - **groups:**
Definiert eine Inspector Group, um die Settings später zu gruppieren
 - **label:**
Das Label für die Inspector Group
 - **position:**
Gibt die Position der Inspector Group an - je kleiner die Zahl, desto weiter oben (und vice versa)

Node-Deklaration: properties

- Enthält eine Liste von sog. „named properties“. Für jede dieser „Properties“ gelten die folgenden Eigenschaften:
- **type**:
Definiert den Datentypen der „Property“. Möglich ist ein Simple Type (PHP), ein FQCN (Fully qualified class name) oder einer der drei Spezial-Typen: data, reference oder references. Für DateTime wird „data“ verwendet, für eine Referenz zu anderen Nodes „reference“ (für eine) oder „references“ (für mehrere).
- **defaultValue**:
Default Wert
- **validation**:
Angabe der Validierungsklasse (z.B. TYPO3.Neos/Validation/NotEmptyValidator) - inkl. der Optionen

Node-Deklaration: **properties**

- Enthält eine Liste von benannten Eigenschaften. Für jede dieser Properties gelten die folgenden Settings
- **ui:**
 - **label:**
Angabe eines beschreibenden Texts, der im User Interface ausgegeben wird
 - **reloadIfChanged:**
Angabe, ob das Element bei Änderungen neu gerendert wird
 - **inlineEditable:**
Direkte Änderung des Elements ist möglich
 - **aloha:**
Konfiguriert die vorhandenen Optionen für den Aloha-Editor (welche Funktionen sind erlaubt und welche nicht)

Node-Deklaration: **properties**

- Enthält eine Liste von benannten Eigenschaften. Für jede dieser Properties gelten die folgenden Settings
- **ui:**
 - **inspector:**
 - **group:**
Angabe der Gruppe, in welche das Element eingesortiert wird (dies wird per `ui.inspector.groups` definiert)
 - **position:**
Gibt die Position des Elements an - je kleiner die Zahl, desto weiter oben
 - **editor:**
Name der JavaScript Editor-Klasse
 - **editorOptions:**
Optionen für den Editor

Node-Deklaration: Beispiele

```
'TYPO3.Neos.NodeTypes:Image':  
superTypes: ['TYPO3.Neos:Content']  
ui:  
  label: 'Image'  
  icon: 'icon-picture'  
  inspector:  
    groups:  
      image:  
        label: 'Image'  
        position: 5  
properties:  
  image:  
    type: TYPO3\Media\Domain\Model\ImageVariant  
    ui:  
      label: 'Image'  
      reloadIfChanged: TRUE  
    inspector:  
      group: 'image'
```

```
alignment:  
  type: string  
  defaultValue: ''  
  ui:  
    label: 'Alignment'  
    reloadIfChanged: TRUE  
  inspector:  
    group: 'image'  
    editor: 'TYPO3.Neos/Inspector/Editors/SelectBoxEditor'  
  editorOptions:  
    placeholder: 'Default'  
    values:  
      '':  
        label: ''  
      center:  
        label: 'Center'  
      left:  
        label: 'Left'  
      right:  
        label: 'Right'
```

```
alternativeText:  
  type: string  
  ui:  
    label: 'Alternative text'  
    reloadIfChanged: TRUE  
  inspector:  
    group: 'image'  
hasCaption:  
  type: boolean  
  ui:  
    label: 'Enable caption'  
    reloadIfChanged: TRUE  
  inspector:  
    group: 'image'  
caption:  
  type: string  
  defaultValue: '<p>Enter caption here</p>'  
  ui:  
    inlineEditable: TRUE
```

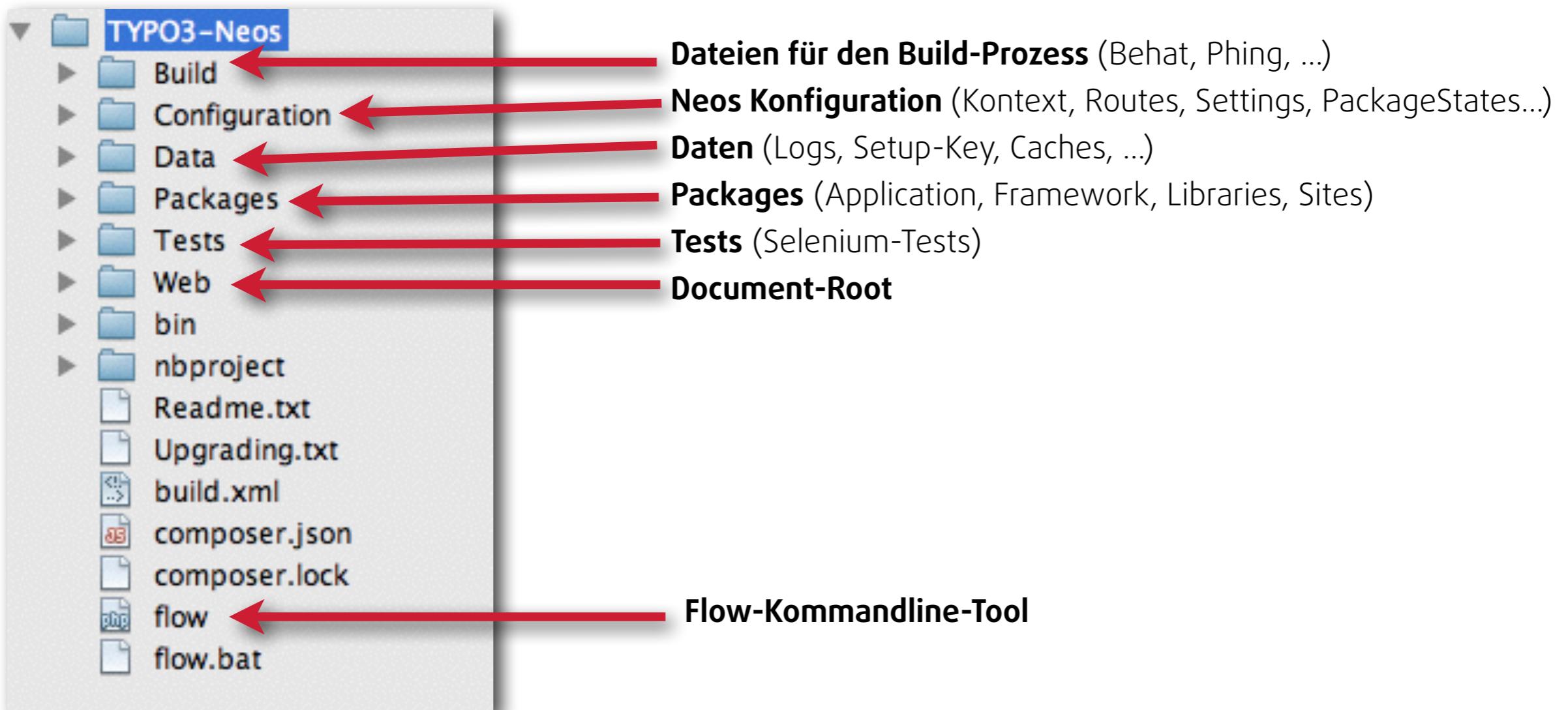
Node-Deklaration: Beispiele

```
aloha:  
  'format':  
    'sub': TRUE  
    'sup': TRUE  
    'p': TRUE  
    'h1': TRUE  
    'h2': TRUE  
    'h3': TRUE  
    'removeFormat': TRUE  
  'link':  
    'a': TRUE
```

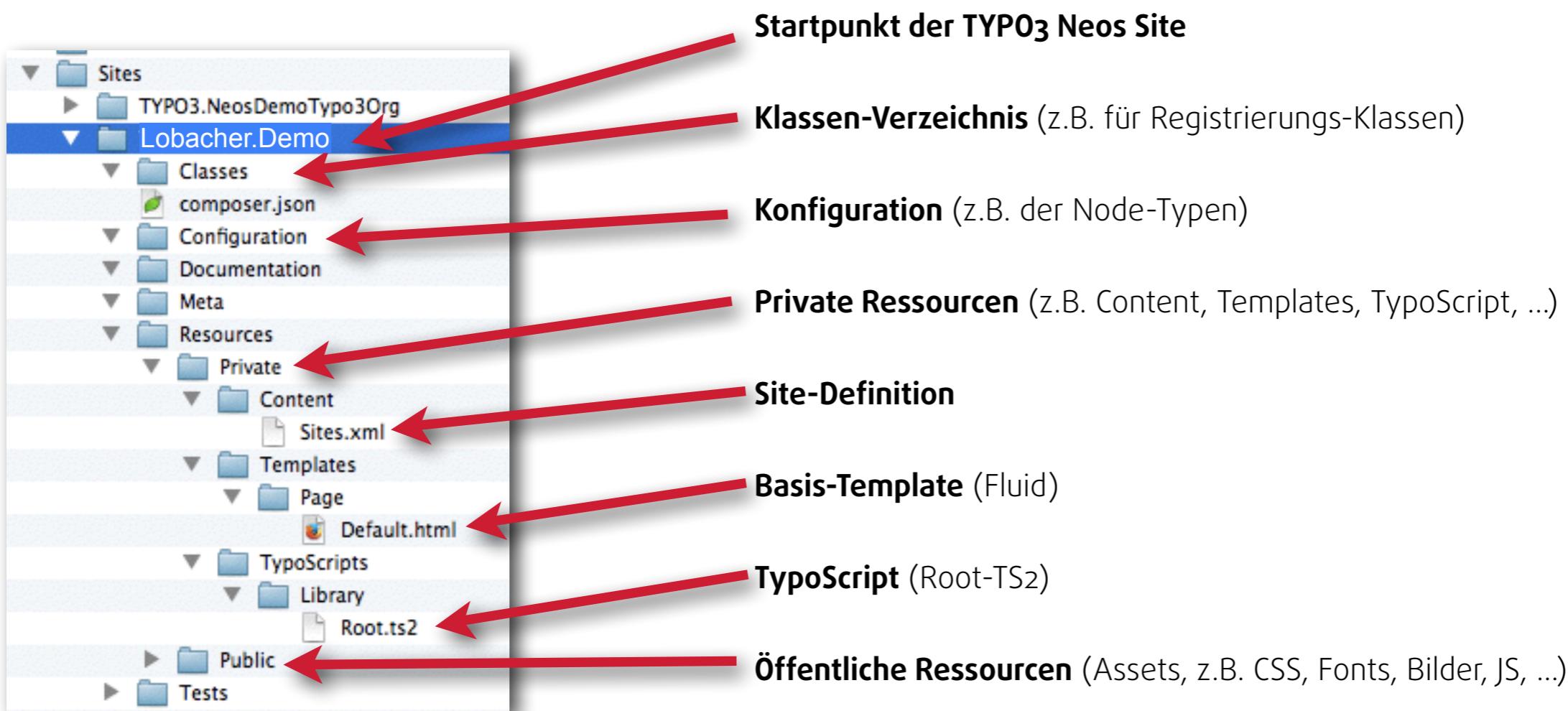
```
'My.Package:SpecialHeadline':  
  superTypes: ['TYPO3.Neos:Content']  
  ui:  
    label: 'Special Headline'  
    group: 'general'  
  properties:  
    headline:  
      type: 'string'  
      defaultValue: 'My Headline Default'  
    ui:  
      inlineEditable: TRUE  
  validation:  
    'TYPO3.Neos/Validation/StringLengthValidator':  
      minimum: 1  
      maximum: 255
```

Aufbau des **TYPO3 Neos Renderings**

Die Verzeichnisstruktur der TYPO3 Flow Basis:



Die Verzeichnisstruktur einer TYPO3 Neos Site:



TypoScript - Lade-Reihenfolge - Root.ts2

- Das Rendering in TYPO3 Neos wird komplett über TypoScript 2 realisiert
- In der Datei Packages/Application/TYPO3.Neos/Configuration/Settings.yaml findet sich dazu die erste Ladenanweisung:

```
TYPO3:  
  Neos:  
    typoScript:  
      autoInclude:  
        'TYPO3.TypoScript': TRUE  
        'TYPO3.Neos': TRUE
```

Dies sorgt dafür, dass folgende beiden Dateien geladen werden:

Packages/Application/TYPO3.TypoScript/Resources/Private/TypoScript/Root.ts2
Packages/Application/TYPO3.Neos/Resources/Private/TypoScript/Root.ts2

TypoScript - Lade-Reihenfolge - Root.ts2

- Achtung: In Pfadangaben wird das „Resources“ weggelassen, z.B.

```
$mergedTypoScriptCode = $this->readExternalTypoScriptFile('resource://  
TYPO3.Neos/Private/TypoScript/Root.ts2') . $siteRootTypoScriptCode;
```

- Dann wird als nächstes die folgende Datei geladen - diese beinhaltet den TypoScript Code der eigenen Site:

Packages/Sites/ [Vendor.Sitename]/Resources/Private/Typoscript/Root.ts2

TypoScript: Root.ts2 (TYPO3.TypoScript) - Teil 1

- Die Datei Root.ts2 im TYPO3.TypoScript Resources-Verzeichnis definiert nun 11 Klassennamen für die jeweiligen Objekte Array, RawArray, Template, Collection, Case, Matcher, Value, Attributes, Tag, UriBuilder und ResourceUri. Die zugehörigen Klassen befinden sich in:

Packages/TYPO3.TypoScript/Classes/TYPO3/TypoScript/TypoScriptObjects:

```
prototype(TYPO3.TypoScript:Array).@class = 'TYPO3\\TypoScript\\
\\TypoScriptObjects\\ArrayImplementation'
```

```
prototype(TYPO3.TypoScript:RawArray).@class = 'TYPO3\\TypoScript\\
\\TypoScriptObjects\\RawArrayImplementation'
```

```
prototype(TYPO3.TypoScript:Template).@class = 'TYPO3\\TypoScript\\
\\TypoScriptObjects\\TemplateImplementation'
```

```
prototype(TYPO3.TypoScript:Collection).@class = 'TYPO3\\TypoScript\\
\\TypoScriptObjects\\CollectionImplementation'
```

TypoScript: Root.ts2 (TYPO3.TypoScript) - Teil 2

- `prototype(TYPO3.TypoScript:Case).@class = 'TYPO3\\TypoScript\\TypoScriptObjects\\CaseImplementation'`
`prototype(TYPO3.TypoScript:Matcher).@class = 'TYPO3\\TypoScript\\TypoScriptObjects\\MatcherImplementation'`
`prototype(TYPO3.TypoScript:Value).@class = 'TYPO3\\TypoScript\\TypoScriptObjects\\ValueImplementation'`

TypoScript: Root.ts2 (TYPO3.TypoScript) - Teil 3

- # Render an HTTP response header

```
prototype(TYPO3.TypoScript:Http.ResponseHead) {
    @class = 'TYPO3\\TypoScript\\TypoScriptObjects\\Http\\
    \\ResponseHeadImplementation'
    headers = TYPO3.TypoScript:RawArray
}
```
- # Render an HTTP message (response)
This is a convenient base prototype for rendering documents.

Usage:

```
# page = TYPO3.TypoScript:Http.Message {
#     httpResponseHead {
#         statusCode = 404
#         headers.Content-Type = 'application/json'
#     }
# }
```

```
prototype(TYPO3.TypoScript:Http.Message) < prototype(TYPO3.TypoScript:Array) {
    httpResponseHead = TYPO3.TypoScript:Http.ResponseHead
    httpResponseHead.@position = 'start 1000'
}
```

TypoScript: Root.ts2 (TYPO3.TypoScript) - Teil 4

- # Renders attributes of a HTML tag

```
#  
# Usage:  
# attributes = TYPO3.TypoScript:Attributes {  
#   foo = 'bar'  
#   class = TYPO3.TypoScript:RawArray {  
#     class1 = 'class1'  
#     class2 = 'class2'  
#   }  
# }  
#  
prototype(TYPO3.TypoScript:Attributes) {  
  @class = 'TYPO3\\TypoScript\\TypoScriptObjects\\AttributesImplementation'  
}
```

TypoScript: Root.ts2 (TYPO3.TypoScript) - Teil 5

- # Renders an HTML tag

```
#  
# Usage:  
# tag = TYPO3.TypoScript:Attributes {  
#   tagName = 'h1'  
#   attributes = {  
#     class = 'some-class'  
#   }  
# }  
#  
prototype(TYPO3.TypoScript:Tag) {  
  @class = 'TYPO3\\TypoScript\\TypoScriptObjects\\TagImplementation'  
  attributes = TYPO3.TypoScript:Attributes  
  omitClosingTag = FALSE  
  selfClosingTag = FALSE  
}
```

TypoScript: Root.ts2 (TYPO3.TypoScript) - Teil 6

- # Renders an URI pointing to a controller/action

```
#  
# Usage:  
# uri = TYPO3.TypoScript:UriBuilder {  
#   package = 'Some.Package'  
#   controller = 'Standard'  
#   action = 'index'  
# }  
#  
prototype(TYPO3.TypoScript:UriBuilder) {  
  @class = 'TYPO3\\TypoScript\\TypoScriptObjects\\UriBuilderImplementation'  
  additionalParams = TYPO3.TypoScript:RawArray  
  arguments = TYPO3.TypoScript:RawArray  
  argumentsToBeExcludedFromQueryString = TYPO3.TypoScript:RawArray  
  
  @exceptionHandler = 'TYPO3\\TypoScript\\Core\\ExceptionHandlers\\  
\\AbsorbingHandler'  
}
```

TypoScript: Root.ts2 (TYPO3.TypoScript) - Teil 7

- # Renders an URI pointing to a resource

Usage:
fileUri = TYPO3.TypoScript:ResourceUri {
path = 'resource://Some.Package/Public/Images/SomeImage.png'
}

prototype(TYPO3.TypoScript:**ResourceUri**) {
 @class = 'TYPO3\\TypoScript\\TypoScriptObjects\\ResourceUriImplementation'
 localize = TRUE

 @exceptionHandler = 'TYPO3\\TypoScript\\Core\\ExceptionHandlers\\
\\AbsorbingHandler'
}

TypoScript: Root.ts2 (TYPO3.Neos)

- Die Datei `Root.ts2` (`Packages/Application/TYPO3.Neos/Resources/Private/TypoScript/Root.ts2`) im TYPO3.Neos Resources-Verzeichnis lädt anschließend folgende Dateien nach:

```
include: DefaultTypoScript.ts2
include: RawContentMode.ts2
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Hier werden zunächst die Prototypen-Definitionen geladen:

```
include: Prototypes/ContentCase.ts2
include: Prototypes/Document.ts2
include: Prototypes/Content.ts2
# Note: The TYPO3.Neos:Template prototype only exists for historical reasons ...
include: Prototypes/Template.ts2
include: Prototypes/PrimaryContent.ts2
include: Prototypes/ContentCollection.ts2
include: Prototypes/Page.ts2
include: Prototypes/Shortcut.ts2
# Note: TYPO3.Neos:Breadcrumb prototype only exists for backwards compatibility
include: Prototypes/Breadcrumb.ts2
include: Prototypes/BreadcrumbMenu.ts2
include: Prototypes/DimensionMenu.ts2
include: Prototypes/Menu.ts2
include: Prototypes/Plugin.ts2
include: Prototypes/PluginView.ts2
include: Prototypes/ConvertUris.ts2
include: Prototypes/ConvertNodeUris.ts2
include: Prototypes/DocumentMetadata.ts2
include: Prototypes/ContentElementWrapping.ts2
include: Prototypes/NodeUri.ts2
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Nun wird das TYPO3.Neos Default-TypoScript geladen und verarbeitet
- Als erstes wird das Objekt **TYPO3.Neos:ContentCase** vom Objekt TYPO3.TypoScript:Case vererbt und die Eigenschaften @position, condition und type innerhalb des Schlüssels default gesetzt.
- Das "case" TypoScript Objekt rendert seine Kind-Elemente in einer Reihenfolge

```
# TYPO3.Neos:ContentCase inherits TYPO3.TypoScript:Case and overrides the
# default case
# with a catch-all condition for the default case, setting the type variable to
# the name of the current nodes' node type
#
prototype(TYPO3.Neos:ContentCase) < prototype(TYPO3.TypoScript:Case) {
    default {
        @position = 'end'
        condition = TRUE
        type = ${q(node).property('_nodeType.name')}
    }
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Dann wird das Objekt **TYPO3.Neos:Document** vom Objekt TYPO3.TypoScript:Template vererbt und die Eigenschaften node auf den aktuellen Node gesetzt.

```
# This is the base Document TypoScript object
# It should be extended by all TypoScript objects which render documents or are
based on node types extending
# TYPO3.Neos:Document.
#
# Note: This object inherits from TYPO3.TypoScript:Template because most
Document types are expected to contain a
# Fluid template. If a Document-based TypoScript object should not render a
template you should still extend this
# prototype and override the @class property.
#
prototype(TYPO3.Neos:Document) < prototype(TYPO3.TypoScript:Template) {
    node = ${node}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 1:** Nun wird das Objekt **TYPO3.Neos:Content** von **TYPO3.TypoScript:Template** vererbt.

```
# This is the base Content TypoScript object
# It must be extended by all elements that are selectable in the backend
#
# Note: This object inherits from TYPO3.TypoScript:Template because most Content Elements are
expected to contain a
# Fluid template. If a Content Element does not render a template (like it is the case for the
Plugin Content Elements)
# you should still extend this prototype and override the @class property (see TYPO3.Neos:Plugin).
#
prototype(TYPO3.Neos:Content) < prototype(TYPO3.TypoScript:Template) {
    node = ${node}

    attributes = TYPO3.TypoScript:Attributes
    attributes.class = ''

    # The following is used to automatically append a class attribute that reflects the underlying
node type of a TypoScript object,
    # for example "typo3-neos-nodetypes-form", "typo3-neos-nodetypes-headline", "typo3-neos-
nodetypes-html", "typo3-neos-nodetypes-image", "typo3-neos-nodetypes-menu" and "typo3-neos-
nodetypes-text"
    # You can disable the following line with:
    # prototype(TYPO3.Neos:Content) {
    #     attributes.class.@process.nodeType >
    # }
    # in your site's TypoScript if you don't need that behavior.
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 2:** Nun wird das Objekt **TYPO3.Neos:Content** von TYPO3.TypoScript:Template vererbt.

```
attributes.class.@process.nodeType = TYPO3.TypoScript:Case {
    @override.nodeTypeClassName = ${String.toLowerCase(String.replaceAll(q(node).property('_nodeType.name'), '/[^[:alnum:]]/','-'))}

    classIsString {
        condition = ${q(value).count() == 1}
        renderer = ${String.trim(value + ' ' + nodeTypeClassName)}
    }

    classIsArray {
        condition = ${q(value).count() > 0}
        renderer = ${Array.push(value, nodeTypeClassName)}
    }
}

# The following line must not be removed as it adds required meta data to all content elements
in backend
@process.contentElementWrapping = TYPO3.Neos:ContentElementWrapping

@exceptionHandler = 'TYPO3\\Neos\\TypoScript\\ExceptionHandlers\\NodeWrappingHandler'
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Dann wird das Objekt **TYPO3.Neos:PrimaryContent** vom Objekt TYPO3.TypoScript:Case vererbt.

```
# Primary content should be used to render the main content of a site and
# provides an extension point for other packages
#
prototype(TYPO3.Neos:PrimaryContent) < prototype(TYPO3.TypoScript:Case) {
    nodePath = 'to-be-defined-by-user'
    @override.nodePath = ${this.nodePath}
    @ignoreProperties = ${['nodePath']}
    default {
        prototype(TYPO3.Neos:ContentCollection) {
            nodePath = ${nodePath}
        }
        condition = TRUE
        type = 'TYPO3.Neos:ContentCollection'
        @position = 'end'
    }
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 1:** Dann wird das Objekt **TYPO3.Neos:ContentCollection** initialisiert, welches wiederum später Content-Elemente aufnehmen kann.

```
# Default case for ContentCollection TS Object
#
prototype(TYPO3.Neos:ContentCollection) < prototype(TYPO3.Neos:Content) {
    @class = 'TYPO3\\Neos\\TypoScript\\ContentCollectionImplementation'

    nodePath = 'to-be-set-by-user'

    itemName = 'node'
    itemRenderer = TYPO3.Neos:ContentCase

    attributes.class = 'neos-contentcollection'

    # You may override this to get your content collection from a different place than the
    # current node.
    # The Eel helper is used to provide a better error message if no content collection could
    # be found.
    @override.contentCollectionNode = ${Neos.Node.nearestContentCollection(node,
this.nodePath)}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 2:** Dann wird das Objekt **TYPO3.Neos:ContentCollection** initialisiert, welches wiederum später Content-Elemente aufnehmen kann.

```
@process.contentElementWrapping {
    node = ${contentCollectionNode}
}

@cache {
    mode = 'cached'

    entryIdentifier {
        collection = ${contentCollectionNode}
        domain = ${site.context.currentDomain}
    }

    entryTags {
        1 = ${'Descendantof_' + contentCollectionNode.identifier}
        2 = ${'Node_' + contentCollectionNode.identifier}
    }

    maximumLifetime = ${q(contentCollectionNode).context({'invisibleContentShown': true}).children().cacheLifetime()}
}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 3:** Dann wird das Objekt **TYPO3.Neos:ContentCollection** initialisiert, welches wiederum später Content-Elemente aufnehmen kann.

```
prototype(TYPO3.Neos:Content) {  
prototype(TYPO3.Neos:ContentCollection) {  
    # Make ContentCollection inside content node types embedded by default.  
    # Usually there's no need for a separate cache entry for container content elements,  
but depending on the use-case  
    # the mode can safely be switched to 'cached'.  
    @cache {  
        mode = 'embed'  
    }  
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 1:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt **TYPO3.TypoScript:Array** vererbt.

```
# TYPO3.Neos:Page is the default object used for rendering in most sites
prototype(TYPO3.Neos:Page) >
prototype(TYPO3.Neos:Page) < prototype(TYPO3.TypoScript:Http.Message) {
    doctype = '<!DOCTYPE html>'
    doctype.@position = 'start 100'
    # Only the opening html tag for the page. This is done to avoid deep nesting of TypoScript paths
    for integrators.
    htmlTag = TYPO3.TypoScript:Tag {
        @position = 'start'
        tagName = 'html'
        omitClosingTag = TRUE
        attributes {
            version = 'HTML+RDFa 1.1'
            version.@if.onlyRenderWhenNotInLiveWorkspace = ${node.context.workspace.name != 'live'}
            xmlns = 'http://www.w3.org/1999/xhtml'
            xmlns.@if.onlyRenderWhenNotInLiveWorkspace = ${node.context.workspace.name != 'live'}
            xmlns:typo3 = 'http://www.typo3.org/ns/2012/Flow/Packages/Neos/Content/'
            xmlns:typo3.@if.onlyRenderWhenNotInLiveWorkspace = ${node.context.workspace.name !=
'live'}
            xmlns:xsd = 'http://www.w3.org/2001/XMLSchema#'
            xmlns:xsd.@if.onlyRenderWhenNotInLiveWorkspace = ${node.context.workspace.name != 'live'}
        }
    }
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 2:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt **TYPO3.TypoScript:Array** vererbt.

```
headerComment = TYPO3.TypoScript:Template {
    @position = 'before headTag'
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/
NeosLicenseHeader.html'
}

# The opening head tag for the page. This is done to avoid deep nesting of TypoScript
paths for integrators.
headTag = TYPO3.TypoScript:Tag {
    @position = 'after htmlTag'
    tagName = 'head'
    omitClosingTag = TRUE
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 3:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt **TYPO3.TypoScript:Array** vererbt.

```
# The content of the head tag,
# integrators can add their own head content in this array.
head = TYPO3.TypoScript:Array {
    @position = 'after headTag'

    characterSetMetaTag = TYPO3.TypoScript:Tag {
        @position = 'start 10'
        tagName = 'meta'
        attributes {
            charset = 'UTF-8'
        }
    }

    titleTag = TYPO3.TypoScript:Tag {
        tagName = 'title'
        content = ${q(node).property('title')}
    }
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 4:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt TYPO3.TypoScript:Array vererbt.

```
neosBackendHeader = TYPO3.TypoScript:Template {
    @position = 'end 10000'
    templatePath = 'resource://TYPO3.Neos/Private/Templates/
TypoScriptObjects/NeosBackendHeaderData.html'
    node = ${node}

    @cache {
        mode = 'uncached'
        context {
            1 = 'node'
        }
    }
    @if.onlyRenderWhenNotInLiveWorkspace = ${
node.context.workspace.name != 'live'}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 5:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt TYPO3.TypoScript:Array vererbt.

```
neosBackendEndpoints = TYPO3.TypoScript:Template {
    @position = 'end 10001'
    templatePath = 'resource://TYPO3.Neos/Private/Templates/
        TypoScriptObjects/NeosBackendEndpoints.html'
    node = ${node}
    account = ${account}

    @cache {
        mode = 'uncached'
        context {
            1 = 'node'
            2 = 'account'
        }
    }
    @if.onlyRenderWhenNotInLiveWorkspace = ${node.context.workspace.name != 'live'}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 6:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt **TYPO3.TypoScript:Array** vererbt.

```
# Script includes in the head should go here
javascripts = TYPO3.TypoScript:Array
# Link tags for stylesheets in the head should go here
stylesheets = TYPO3.TypoScript:Array
}
closingHeadTag = '</head>'
closingHeadTag.@position = 'after head'
# The opening body tag for the page. This is done to avoid deep nesting of
TypoScript paths for integrators.
bodyTag = TYPO3.TypoScript:Tag {
    @position = '20'
    tagName = 'body'
    omitClosingTag = TRUE
}
# Required for the backend to work.
neosBackendDocumentNodeData = TYPO3.Neos:DocumentMetadata {
    @position = 'before body'
    @if.onlyRenderWhenNotInLiveWorkspace = ${node.context.workspace.name != 'live'}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 7:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt TYPO3.TypoScript:Array vererbt.

```
# Content of the body tag. To be defined by the integrator.  
body = TYPO3.TypoScript:Template {  
    @position = 'after bodyTag'  
    node = ${node}  
    site = ${site}  
    # Script includes before the closing body tag should go  
here  
    javascripts = TYPO3.TypoScript:Array  
        # This processor appends the rendered javascripts Array  
        to the rendered template  
        @process.appendJavaScripts = ${value + this.javascripts}  
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 8:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt TYPO3.TypoScript:Array vererbt.

```
# Required for the backend to work.  
  
neosBackendContainer = TYPO3.TypoScript:Template {  
    @position = 'before closingBodyTag'  
    templatePath = 'resource://TYPO3.Neos/Private/Templates/  
        TypoScriptObjects/NeosBackendContainer.html'  
    node = ${node}  
  
    @cache {  
        mode = 'uncached'  
        context {  
            1 = 'node'  
        }  
    }  
    @if.onlyRenderWhenNotInLiveWorkspace = $  
    {node.context.workspace.name != 'live'}  
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 9:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt **TYPO3.TypoScript:Array** vererbt.

```
# This enables redirecting to the last visited page after login
lastVisitedNodeScript = TYPO3.TypoScript:Tag {
    @position = 'before closingBodyTag'
    tagName = 'script'
    attributes {
        data-neos-node = ${q(node).property('_identifier')}
        src = TYPO3.TypoScript:ResourceUri {
            path = 'resource://TYPO3.Neos/Public/JavaScript/LastVisitedNode.js'
        }
    }
}
neosBackendFooter = TYPO3.TypoScript:Template {
    @position = 'after lastVisitedNodeScript 10000'
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/
NeosBackendFooterData.html'
    node = ${node}
    @if.onlyRenderWhenNotInLiveWorkspace = ${node.context.workspace.name != 'live'}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 10:** Dann wird das Objekt **TYPO3.Neos:Page** vom Objekt **TYPO3.TypoScript:Array** vererbt.

```
closingBodyTag = '</body>'  
closingBodyTag.@position = 'end 100'  
closingHtmlTag = '</html>'  
closingHtmlTag.@position = 'end 200'  
  
@cache {  
    mode = 'cached'  
    entryIdentifier {  
        documentNode = ${node}  
        domain = ${site.context.currentDomain}  
    }  
    entryTags {  
        1 = ${'Node_' + node.identifier}  
    }  
}  
@exceptionHandler = 'TYPO3\\Neos\\TypoScript\\ExceptionHandlers\\  
\\PageHandler'  
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 1:** Dann wird das Objekt **TYPO3.Neos:Shortcut** vom Objekt **TYPO3.TypoScript:Template** vererbt.

```
# TYPO3.Neos.Shortcut is given a representation for editing purposes
#
prototype(TYPO3.Neos:Shortcut) < prototype(TYPO3.TypoScript:Template) {
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/
Shortcut.html'

    targetMode = ${q(node).property('targetMode')}
    firstChildNode = ${q(node).children('[instanceof TYPO3.Neos:Document]').get(0)}
    target = ${q(node).property('target')}
    targetConverted = ${Neos.Link.hasSupportedScheme(this.target) ?
Neos.Link.convertUriToObject(this.target, node) : null}
    targetSchema = ${Neos.Link.getScheme(this.target)}

    node = ${node}

    @exceptionHandler = 'TYPO3\\Neos\\TypoScript\\ExceptionHandlers\
\\NodeWrappingHandler'
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 2:** Dann wird das Objekt **TYPO3.Neos:Shortcut** vom Objekt **TYPO3.TypoScript:Template** vererbt.

```
{namespace neos=TYPO3\Neos\ViewHelpers
<div id="neos-shortcut"><p>
    <f:switch expression="{targetMode}">
        <f:case value="selectedTarget">
            This is a shortcut to a specific target:<br />
            <f:if condition="{target}">
                <f:then>
                    <f:switch expression="{targetSchema}">
                        <f:case value="node">Click <neos:link.node node="{targetConverted}">{targetConverted.label}</
neos:link.node> to continue to the page.</f:case>
                        <f:case value="asset">Click <a href="{f:uri.resource(resource: targetConverted.resource)}" target="_blank">{targetConverted.label}</a> to see the file.</f:case>
                        <f:defaultCase>Click <a href="{target}" target="_blank">{target}</a> to open the link.</
f:defaultCase>
                    </f:switch>
                </f:then>
                <f:else>(no target has been selected)</f:else>
            </f:if>
        </f:case>
        <f:case value="firstChildNode">
            This is a shortcut to the first child page.<br />
            Click <neos:link.node node="{firstChildNode}">here</neos:link.node> to continue to the page.
        </f:case>
        <f:case value="parentNode">
            This is a shortcut to the parent page.<br />
            Click <neos:link.node node="{node.parent}">here</neos:link.node> to continue to the page.
        </f:case>
    </f:switch>
</p></div>
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Nun wird das Objekt **TYPO3.Neos:BreadcrumbMenu** vom Objekt **TYPO3.TypoScript:Template** vererbt.

```
# TYPO3.Neos:BreadcrumbMenu provides a breadcrumb navigation based on menu items.
prototype(TYPO3.Neos:BreadcrumbMenu) < prototype(TYPO3.Neos:Menu) {
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/
BreadcrumbMenu.html'
    itemCollection = ${q(node).add(q(node).parents('[instanceof TYPO3.Neos:Document')).get()}
    attributes.class = 'breadcrumb'
}
```

- Zugehöriges Template:**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<f:if condition="{items}">
    <ul{attributes -> f:format.raw()}>
        <f:for each="{items}" as="item" reverse="TRUE">
            <li{ts:render(path:'{item.state}.attributes', context: {item: item}) -> f:format.raw()}>
                <f:if condition="{item.state} == 'current'">
                    <f:then>{item.label}</f:then>
                    <f:else><neos:link.node node="{item.node}" /></f:else>
                </f:if>
            </li>
        </f:for>
    </ul>
</f:if>
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Nun wird das Objekt **TYPO3.Neos:DimensionMenu** vom Objekt TYPO3.Neos:Menu vererbt.

```
# TYPO3.Neos:DimensionMenu provides basic dimension (e.g. language) menu rendering
prototype(TYPO3.Neos:DimensionMenu) < prototype(TYPO3.Neos:Menu) {
    @class = 'TYPO3\\Neos\\TypoScript\\DimensionMenuImplementation'
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/DimensionMenu.html'
    absent.attributes = TYPO3.TypoScript:Attributes {
        class = 'normal'
    }
}
```

- **Zugehöriges Template:**

```
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<ul{attributes -> f:format.raw()}>
<f:for each="{items}" as="item">
    <li{ts:render(path:'{item.state}.attributes', context: {item: item}) -> f:format.raw()}>
        <f:if condition="{item.node}">
            <f:then>
                <neos:link.node node="{item.node}">{item.label}</neos:link.node>
            </f:then>
            <f:else>
                {item.label}
            </f:else>
        </f:if>
    </li>
</f:for>
</ul>
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 1:** Dann wird das Objekt **TYPO3.Neos:Menu** von **TYPO3.TypoScript:Template** vererbt.

```
# TYPO3.Neos:Menu provides basic menu rendering
prototype(TYPO3.Neos:Menu) < prototype(TYPO3.TypoScript:Template) {
    @class = 'TYPO3\\Neos\\TypoScript\\MenuImplementation'
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/Menu.html'
    node = ${node}
    items = ${this.items}
    filter = 'TYPO3.Neos:Document'
    attributes = TYPO3.TypoScript:Attributes
    active.attributes = TYPO3.TypoScript:Attributes {
        class = 'active'
    }
    current.attributes = TYPO3.TypoScript:Attributes {
        class = 'current'
    }
    normal.attributes = TYPO3.TypoScript:Attributes {
        class = 'normal'
    }
    @exceptionHandler = 'TYPO3\\Neos\\TypoScript\\ExceptionHandlers\\NodeWrappingHandler'
    @cache {
        mode = 'cached'
        entryIdentifier {
            documentNode = ${documentNode}
            domain = ${site.context.currentDomain}
        }
        entryTags {
            1 = 'NodeType_TYPO3.Neos:Document'
        }
    }
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 2:** Dann wird das Objekt **TYPO3.Neos:Menu** von TYPO3.TypoScript:Template vererbt.

```
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<ul{attributes -> f:format.raw()}>
    <f:render section="itemsList" arguments="{items: items}" />
</ul>
<f:section name="itemsList">
    <f:for each="{items}" as="item">
        <li{ts:render(path:'{item.state}.attributes', context: {item: item}) -> f:format.raw()}>
            <neos:link.node node="{item.node}" />
            <f:if condition="{item.subItems}">
                <ul>
                    <f:render section="itemsList" arguments="{items: item.subItems}" />
                </ul>
            </f:if>
            </li>
        </f:for>
    </f:section>
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Dann wird das Objekt **TYPO3.Neos:Plugin** vom Objekt TYPO3.TypoScript:Content vererbt.

```
# Abstract Plugin Implementation
# This should be extended by all plugin Content Elements
#
prototype(TYPO3.Neos:Plugin) >
prototype(TYPO3.Neos:Plugin) < prototype(TYPO3.Neos:Content) {
    @class = 'TYPO3\\Neos\\TypoScript\\PluginImplementation'

    @cache {
        mode = 'uncached'
        context {
            1 = 'node'
            2 = 'documentNode'
        }
    }
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Dann wird das Objekt **TYPO3.Neos:PluginView** vom Objekt TYPO3.TypoScript:Content vererbt.

```
# PluginView Implementation
# This represents a View that is always bound to a master Plugin
# Usually you won't need to extend this
#
prototype(TYPO3.Neos:PluginView) >
prototype(TYPO3.Neos:PluginView) < prototype(TYPO3.Neos:Content) {
    @class = 'TYPO3\\Neos\\TypoScript\\PluginViewImplementation'
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Dann wird das Objekt **TYPO3.Neos:ConvertUris** initialisiert

```
# ConvertUris implementation
#
# replaces all occurrences of "node://<UUID>" by proper URIs.
#
prototype(TYPO3.Neos:ConvertUris) {
    @class = 'TYPO3\\Neos\\TypoScript\\ConvertUrisImplementation'
    value = ${value}
    node = ${node}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Dann wird das Objekt **TYPO3.Neos:DocumentMetadata** vom Objekt **TYPO3.TypoScript:Tag** vererbt.

```
# DocumentMetadata implementation
#
# Renders a meta tag with attributes about the current document node
#
prototype(TYPO3.Neos:DocumentMetadata) < prototype(TYPO3.TypoScript:Tag) {
    tagName = 'div'
    forceClosingTag = TRUE
    attributes {
        id = 'neos-document-metadata'
    }

    @process.contentElementWrapping = TYPO3.Neos:ContentElementWrapping
    @if.onlyRenderWhenNotInLiveWorkspace = ${node.context.workspace.name != 'live'}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Dann wird das Objekt **TYPO3.Neos:ContentElementWrapping** initialisiert

```
# ContentElementWrapping implementation
#
# Used as processor this adds metadata attributes to the corresponding
# TypoScript object
# This is used to render required data-neos-* attributes to content elements in
# the backend
#
prototype(TYPO3.Neos:ContentElementWrapping) {
    @class = 'TYPO3\\Neos\\TypoScript\\ContentElementWrappingImplementation'
    node = ${node}
    value = ${value}
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- Dann wird das Objekt **TYPO3.Neos:NodeUri** initialisiert

```
# Renders an URI pointing to a node
#
prototype(TYPO3.Neos:NodeUri) {
    @class = 'TYPO3\\Neos\\TypoScript\\NodeUriImplementation'
    additionalParams = TYPO3.TypoScript:RawArray
    arguments = TYPO3.TypoScript:RawArray
    additionalParams = TYPO3.TypoScript:RawArray
    argumentsToBeExcludedFromQueryString = TYPO3.TypoScript:RawArray
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 1:** Nun wird das Rendering aufgebaut

```
# The root matcher used to start rendering in Neos
#
# The default is to use a render path of "page", unless the requested format is
# not "html"
# in which case the format string will be used as the render path (with dots
# replaced by slashes)
#
root = TYPO3.TypoScript:Case
root {
    shortcut {
        prototype(TYPO3.Neos:Page) {
            body = TYPO3.Neos:Shortcut
        }

        @position = 'start'
        condition = ${q(node).is('[instanceof TYPO3.Neos:Shortcut]')}
        type = 'TYPO3.Neos:Page'
    }
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 2:** Dann wird das Rendering aufgebaut

```
editPreviewMode {  
    @position = 'end 9996'  
    condition = ${editPreviewMode != null}  
    renderPath = ${'/' + editPreviewMode}  
}  
layout {  
    @position = 'end 9997'  
    layout = ${q(node).property('layout') != null &&  
    q(node).property('layout') != '' ? q(node).property('layout') :  
    q(node).parents('[subpageLayout]').first().property('subpageLayout')}  
    condition = ${this.layout != null && this.layout != ''}  
    renderPath = ${'/' + this.layout}  
}  
format {  
    @position = 'end 9998'  
    condition = ${request.format != 'html'}  
    renderPath = ${'/' + String.replace(request.format, '.', '/')}  
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 3:** Dann wird das Rendering aufgebaut

```
default {
    @position = 'end 9999'
    condition = TRUE
    renderPath = '/page'
}

@cache {
    mode = 'cached'

    entryIdentifier {
        node = ${node}
        editPreviewMode = ${editPreviewMode}
        format = ${request.format}
        domain = ${site.context.currentDomain}
    }
}
```

TypoScript: DefaultTypoScript.ts2 (TYPO3.Neos)

- **Teil 4:** Dann wird das Rendering aufgebaut

```
entryTags {  
    # Whenever the node changes the matched condition could change  
    1 = ${'Node_' + documentNode.identifier}  
    # Whenever one of the parent nodes changes the layout could change  
    2 = ${'DescendantOf_' + documentNode.identifier}  
}  
  
# Catch all unhandled exceptions at the root  
@exceptionHandler = 'TYPO3\\Neos\\TypoScript\\ExceptionHandlers\\  
\PageHandler'  
}
```

TypoScript: RawContentMode.ts2 (TYPO3.Neos)

Einstellungen für den RawContentMode - Teil 1

```
prototype(TYPO3.Neos:RawContent) >
  prototype(TYPO3.Neos:RawContent) < prototype(TYPO3.TypoScript:Template) {
    templatePath = 'resource://TYPO3.Neos/Private/Templates/RawContentMode/
  TypoScriptObjects/GeneralContentCollectionRendering.html'
    columns = TYPO3.TypoScript:Collection {
      collection = ${q(node).children('[instanceof TYPO3.Neos:ContentCollection]')}
      itemName = 'node'
      itemRenderer = TYPO3.TypoScript:Template {
        templatePath = 'resource://TYPO3.Neos/Private/Templates/RawContentMode/
  TypoScriptObjects/ContentCollectionTemplate.html'
        node = ${node}
        attributes = TYPO3.TypoScript:Attributes {
          class = 'column'
        }
        columnContent = TYPO3.Neos:ContentCollection {
          nodePath = '.'
        }
      }
    }
  }
```

TypoScript: RawContentMode.ts2 (TYPO3.Neos)

Einstellungen für den RawContentMode - Teil 2

```
rawContent = TYPO3.Neos:Page {
    head {
        stylesheets = TYPO3.TypoScript:Template {
            templatePath = 'resource://TYPO3.Neos/Private/Templates/
RawContentMode/Page/Default.html'
            sectionName = 'headerIncludes'
        }
    }
    bodyTag.attributes.class = 'neos-raw-content-mode'
    body {
        templatePath = 'resource://TYPO3.Neos/Private/Templates/
RawContentMode/Page/Default.html'
        sectionName = 'body'
        allContentCollections = TYPO3.Neos:PrimaryContent {
            nodePath = '.'
            default.type = 'TYPO3.Neos:RawContent'
        }
    }
}
```

TypoScript: Root.ts2 (TYPO3.Neos.NodeTypes)

- **Teil 1:** NodeType Definition (z.B. Menü welches als Content Element eingefügt wird)

```
# Menu TS Object - extends TYPO3.Neos:Menu and is rendering menus inserted as content
elements
prototype(TYPO3.Neos.NodeTypes:Menu) {
    @class = 'TYPO3\\Neos\\TypoScript\\MenuImplementation'
    startingPoint = ${documentNode}
    itemCollection = ${Array.isEmpty(q(node).property('selection')) ?
q(node).property('selection') : {}} ? NULL : q(node).property('selection')
    entryLevel = ${q(node).property('startLevel')}
    entryLevel.@process.1 = ${String.toInteger(value)}
    maximumLevels = ${q(node).property('maximumLevels')}
    maximumLevels.@process.1 = ${String.toInteger(value)}
    active.attributes = TYPO3.TypoScript:Attributes {
        class = 'active'
    }
    current.attributes = TYPO3.TypoScript:Attributes {
        class = 'current'
    }
    normal.attributes = TYPO3.TypoScript:Attributes {
        class = 'normal'
    }
    node = ${node}
    items = ${this.items}
}
```

TypoScript: Root.ts2 (TYPO3.Neos.NodeTypes)

- **Teil 2:** NodeType Definition (Elemente, die als Content Element eingefügt werden)

```
# Headline TS Object
prototype(TYPO3.Neos.NodeTypes:Headline) {
    title.@process.convertUris = TYPO3.Neos:ConvertUris
}

# Image TS Object
prototype(TYPO3.Neos.NodeTypes:Image) {
    maximumWidth = 2560
    maximumHeight = 2560
    imageClassName = ${q(node).property('alignment') ? ('typo3-neos-alignment-' +
q(node).property('alignment')) : ''}
    allowCropping = FALSE
    allowUpScaling = FALSE
    link.@process.convertUris = TYPO3.Neos:ConvertUris {
        forceConversion = true
    }
}

# Text TS Object
prototype(TYPO3.Neos.NodeTypes:Text) {
    text.@process.convertUris = TYPO3.Neos:ConvertUris
}
```

TypoScript: Root.ts2 (TYPO3.Neos.NodeTypes)

- **Teil 3:** NodeType Definition (Elemente, die als Content Element eingefügt werden)

```
# TextWithImage TS Object
prototype(TYPO3.Neos.NodeTypes:TextWithImage) < prototype(TYPO3.Neos.NodeTypes:Image) {
    text.@process.convertUris = TYPO3.Neos:ConvertUris
}

# Basic implementation of a flexible MultiColumn element, not exposed directly but
inherited by all specific MultiColumn content elements
prototype(TYPO3.Neos.NodeTypes:MultiColumn) < prototype(TYPO3.Neos:Content) {
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/NodeTypes/
MultiColumn.html'
    layout = ${q(node).property('layout')}
    attributes.class = ${'container columns-' + q(node).property('layout')}
    columns = TYPO3.TypoScript:Collection {
        collection = ${q(node).children('[instanceof TYPO3.Neos:ContentCollection]')}
        itemRenderer = TYPO3.Neos.NodeTypes:MultiColumnItem
        itemName = 'node'
    }
}
```

TypoScript: Root.ts2 (TYPO3.Neos.NodeTypes)

- **Teil 4:** NodeType Definition (Elemente, die als Content Element eingefügt werden)

```
# Abstract render definition for a single content column in a multi column element
prototype(TYPO3.Neos.NodeTypes:MultiColumnItem) < prototype(TYPO3.TypoScript:Template) {
    node = ${node}
    templatePath = 'resource://TYPO3.Neos.NodeTypes/Private/Templates/NodeTypes/
MultiColumnItem.html'
    attributes = TYPO3.TypoScript:Attributes {
        class = 'column'
    }
    columnContent = TYPO3.Neos:ContentCollection {
        nodePath = '.'
    }
}
# Two Column TS Object
prototype(TYPO3.Neos.NodeTypes:TwoColumn) >
prototype(TYPO3.Neos.NodeTypes:TwoColumn) < prototype(TYPO3.Neos.NodeTypes:MultiColumn)
# Three Column TS Object
prototype(TYPO3.Neos.NodeTypes:ThreeColumn) >
prototype(TYPO3.Neos.NodeTypes:ThreeColumn) < prototype(TYPO3.Neos.NodeTypes:MultiColumn)
# Four Column TS Object
prototype(TYPO3.Neos.NodeTypes:FourColumn) >
prototype(TYPO3.Neos.NodeTypes:FourColumn) < prototype(TYPO3.Neos.NodeTypes:MultiColumn)
```

TypoScript: Root.ts2 (TYPO3.Neos.NodeTypes)

- **Teil 5:** NodeType Definition (Elemente, die als Content Element eingefügt werden)

```
# Form TS Object
prototype(TYPO3.Neos.NodeTypes:Form) {
    presetName = 'default'
    @cache {
        mode = 'uncached'
        context {
            1 = 'node'
            2 = 'documentNode'
        }
    }
}
prototype(TYPO3.Neos.NodeTypes:AssetList).@class = 'TYPO3\\Neos\\NodeTypes\\
\\TypoScriptObjects\\AssetListImplementation'
# Insert records TS Object
prototype(TYPO3.Neos.NodeTypes:Records) {
    records = TYPO3.TypoScript:Collection {
        collection = ${q(node).property('records')}
        itemRenderer = TYPO3.Neos:ContentCase
        itemName = 'node'
    }
}
```

TypoScript: Root.ts2 & Template der Site (1/3)

- Dann wird das TypoScript der Site ausgewertet
- Begonnen wird damit mit der Datei Root.ts2 in folgendem Verzeichnis

Sites/ [Vendor] . [Sitename] /Resources/Private/TypoScript/

- Standardmäßig (leere Website) wird dort folgender Code angelegt:

```
/**  
 * Root TypoScript template for the Website site  
 */  
page = Page {  
    head {  
        stylesheets.site = TYPO3.TypoScript:Template {  
            templatePath = 'resource://Lobacher.Demo/Private/Templates/Page/Default.html'  
            sectionName = 'stylesheets'  
        }  
  
        javascripts.site = TYPO3.TypoScript:Template {  
            templatePath = 'resource://Lobacher.Demo/Private/Templates/Page/Default.html'  
            sectionName = 'headScripts'  
        }  
    }  
}
```

TypoScript: Root.ts2 & Template der Site (2/3)

- Per Default wird dort folgender Code angelegt:

```
...
body {
    templatePath = 'resource://Lobacher.Demo/Private/Templates/Page/Default.html'
    sectionName = 'body'
    parts {
        menu = Menu
        breadcrumb = Breadcrumb
    }
    // These are your content areas, you can define as many as you want, just name them and the
    nodePath.
    content {
        // The default content section
        main = PrimaryContent {
            nodePath = 'main'
        }
    }
}

javascripts.site = TYPO3.TypoScript:Template {
    templatePath = 'resource://Lobacher.Demo/Private/Templates/Page/Default.html'
    sectionName = 'bodyScripts'
}
}
```

TypoScript: Root.ts2 & Template der Site (3/3)

- Das Default HTML-Template wird wie folgt angelegt:

```
<!DOCTYPE html>
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<html>
<head>
    <f:section name="stylesheets">
        <!-- Put your stylesheet inclusions here, they will be included in your website by
TypoScript -->
    </f:section>
    <f:section name="headScripts">
        <!-- Put your scripts inclusions for the head here, they will be included in your website
by TypoScript -->
    </f:section>
</head>
<body>
<f:section name="body">
    <nav class="menu">{parts.menu -> f:format.raw()}</nav>
    <nav class="breadcrumb">{parts.breadcrumb -> f:format.raw()}</nav>
    <div class="content">{content.main -> f:format.raw()}</div>
</f:section>
<f:section name="bodyScripts">
    <!-- Put your scripts inclusions for the end of the body here, they will be included in your
website by TypoScript -->
</f:section>
</body>
</html>
```

TYPO3 NEOS

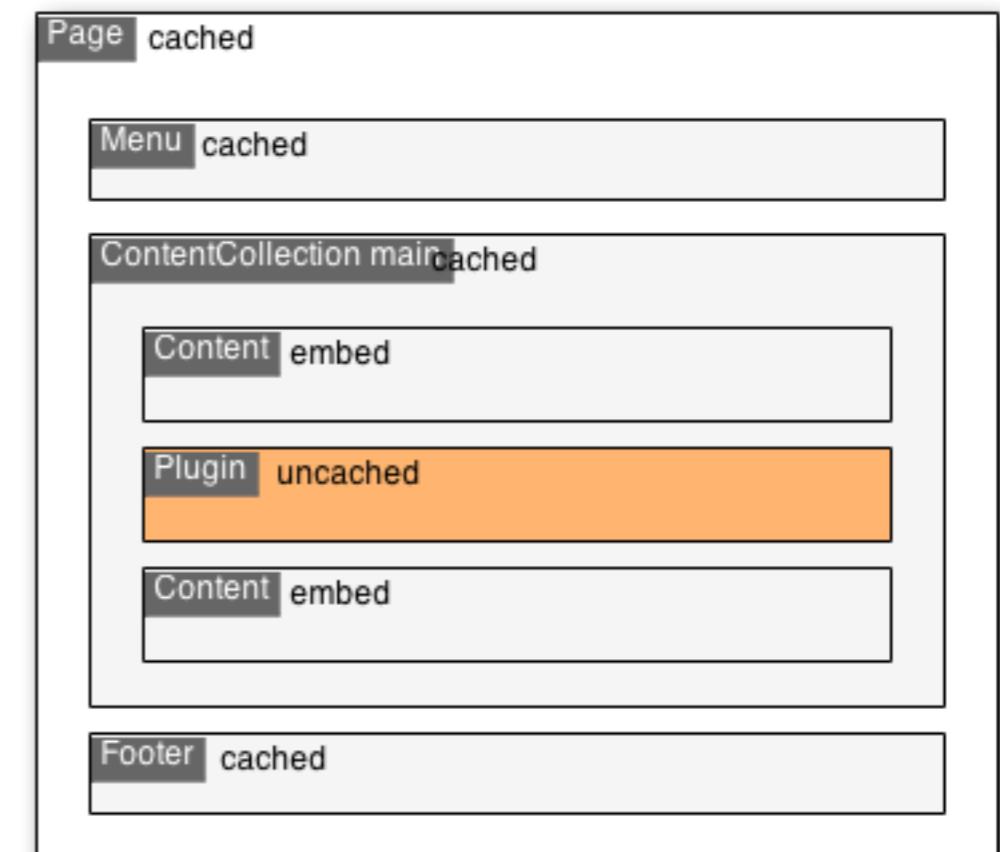
Content Cache

Content Cache

- Das Frontend-Rendering einer Document-Node in Neos involviert viele Abfragen und Operationen.
- Dies für jeden Request durchzuführen, wäre zu langsam, um eine akzeptable Response-Zeit zu erreichen.
- Der in TYPO3 NEOS 1.1 eingeführte „Content Cache“ ist ein TypoScript-Feature.
- Dieser unterstützt Konfiguration und einen verschachtelten Cache, der viele Anfragen direkt aus dem Cache beantworten kann.
- Der Cache basiert auf dem TYPO3 Flow Caching Framework, welches verschiedene Cache-Backends, Tagging und Invalidierung unterstützt.

Content Cache

- Jeder TypoScript-Pfad (der ein Objekt repräsentiert) kann eine eigene Cache-Konfiguration besitzen. Diese kann verschachtelt werden, um Teile des Contents wieder zu verwenden bzw. um mehrere Cache-Einträge mit unterschiedlichen Eigenschaften auf der selben Seite zu haben. Dies kann ein Menü oder eine Section sein, die auf vielen Seiten identisch ist. Das Konzept der Verschachtelung unterstützt auch ungecachten Inhalt innerhalb von gecachten.
- Der Content-Cache ist sogar im Editor-Modus aktiv. Cache-Einträge werden automatisch gelöscht, wenn sich die Daten (Code, Template oder Konfiguration) ändern.



Content Cache - Basis (1/5)

- Der Haupt-TypoScript-Pfad ist `root`, welcher in der Datei `TypoScript/DefaultTypoScript.ts2` im Package `TYPO3.Neos` package definiert ist.

```
root = TYPO3.TypoScript:Case {
    default {
        @position = 'end 9999'
        condition = TRUE
        renderPath = '/page'
    }
    @cache {
        mode = 'cached'
        maximumLifetime = '86400'
        entryIdentifier {
            node = ${node}
            editPreviewMode = ${editPreviewMode}
            format = ${request.format}
        }
        entryTags {
            # Whenever the node changes the matched condition could change
            1 = ${'Node_' + documentNode.identifier}
            # Whenever one of the parent nodes changes the layout could change
            2 = ${'DescendantOf_' + documentNode.identifier}
        }
    }
}
```

Content Cache - Basis (2/5)

- Die Default-Konfiguration schreibt den gesamten Seiten-Inhalt mit einem eindeutigen Identifier (entryIdentifier) in den Cache, der aus der aktuellen Node (Document Node), dem Preview-Mode und dem Format kalkuliert wird.
- In der @cache-Eigenschaft können folgende Untereigenschaften verwendet werden:
- mode**
 - Setzt den Caching-Modus für den aktuellen Pfad. Mögliche Werte sind **embed** (default), **cached** oder **uncached**. Es werden nur einfache String-Werte unterstützt
 - Der Default-Modus **embed** erstellt selbst keine neuen Cache-Einträge, sondern speichert den Inhalt im nächst möglichen äußeren Cache-Eintrag. Mit dem Modus **cached** wird ein eigener Cache-Eintrag für den Pfad erstellt. Der Modus **uncached** schließlich kann dafür verwendet werden, um einen Pfad innerhalb eines gedachten Pfades immer neu zu rendern. Die Eigenschaft **context** sollte gesetzt werden, um die TypoScript-Context-Variablen zu konfigurieren, die verfügbar sind, sobald ein ungewachsener Pfad evaluiert wird.

```
@cache {
    mode = 'uncached'
    context {
        1 = 'node'
    }
}
```

Content Cache - Basis (3/5)

- **maximumLifetime**
 - Setzt die maximale Gültigkeit für den nächsten gecachten Pfad. Mögliche Werte **null** (default), **0** (unbegrenzte Gültigkeit) oder die Anzahl Sekunden als Integer.
 - Wenn diese Eigenschaft auf einen Pfad angewendet wird, mit dem Caching-Modus **cached**, dann setzt es die Gültigkeit des Cache-Eintrages auf das Minimum aller verschachtelten **maximumLifetime** Konfigurationen (in Pfaden mit dem Modus **embed**) bzw. der **maximumLifetime** der aktuellen Konfiguration.
- **entryIdentifier**
 - Konfiguriert den Identifier für den Cache-Eintrag des Modus **cached** basierend auf einem Array von Werten. Wenn nichts angegeben wird, wird der Identifier aus allen TypoScript **context**-Werten , die einfache Werte sind oder das CacheAwareInterface implementiert, zusammengebaut.
 - Der Wert des Identifiers ist ein Hash, der über alle Array-Werte ermittelt wird, und deren Schlüssel enthält sowie nach diesem sortiert ist.

Content Cache - Basis (4/5)

- Es ist sehr wichtig, dass man alle Werte, die die Ausgabe des aktuellen Pfads zu beeinflussen, zum **entryIdentifier** Array hinzufügt, da Cache-Einträge über alle Seiten wiederverwendet werden, wenn der selbe Identifier angefordert wird. In der Cache-Hierarchie bestimmt der äußerste Cache-Eintrag die verschachtelten Einträge, daher ist es wichtig, Einträge hinzuzufügen, die das Rendering für jeden gedachten Pfad innerhalb der Hierarchie beeinflussen
- **entryTags**
 - Konfiguriert ein Set an Tags, die an den Cache-Eintrag (im Modus **cached**) als Array zugewiesen werden.
 - Die richtigen Entry-Tags sind wichtig, um ein automatisches Löschen der Cache-Einträge zu erwirken, wenn eine Node oder andere Daten in Neos geändert wurden.

Content Cache - Basis (5/5)

- **context**
 - Konfiguriert eine Liste von Variablen-Namen aus dem TypOScript Kontext, die später im Pfad mit dem Modus uncached gerendert werden. Lediglich die hier definierten Werte sind später im TypoScript verfügbar, wenn der Pfad ausgewertet wird.

```
prototype(TYPO3.Neos:Plugin) {
    @cache {
        mode = 'uncached'
        context {
            1 = 'node'
            2 = 'documentNode'
        }
    }
}
```

Content Cache - Cache Entry Tags

- Neos flusht ein Set an Tags automatisch, wann immer eine Node erstellt, geändert, publiziert oder verworfen wird. Das exakte Set an Tags hängt von der Node-Hierarchie und dem NodeType der geänderten Node ab. Daher sollte man Tags zuweisen, die eines der folgenden Pattern entsprechen. Zum Aufbau der Pattern kann man einen Eel-Ausdruck in Abhängigkeit von der Kontext-Variablen verwenden, der zudem den Node-Identifier und/oder den Typ enthält.
- **Die folgenden Tag-Patterns werden von Neos geflusht:**
- **Everything**
 - Flusht den Cache-Eintrag für jede geänderte Node

Content Cache - Cache Entry Tags

- **NodeType_[My.Package:NodeTypeName]**
 - Flusht den Cache-Eintrag, wenn eine Node des angegebenen Typs geändert wird. Vererbung wird durchgeführt, so wird z.B. für eine geänderte Node vom Typ TYPO3.Neos.NodeTypes:Page sowohl die Tags NodeType_TYPO3.Neos.NodeTypes:Page wie auch NodeType_TYPO3.Neos:Document (und ggf. mehr) geflusht.
- **Node_[Identifier]**
 - Flusht den Cache-Eintrag, wenn eine Node des angegebenen Node-Identifiers geändert wird.
- **DescendantOf_[Identifier]**
 - Flusht den Cache-Eintrag, wenn eine Kind-Node des angegebenen Node-Identifiers geändert wird.

Content Cache - Cache Entry Tags

- Beispiel:

```
prototype(TYPO3.Neos:ContentCollection) {  
    #...  
    @cache {  
        #...  
        entryTags {  
            1 = ${'DescendantOf_' + contentCollectionNode.identifier}  
        }  
    }  
}
```

- Die Cache-Konfiguration der ContentCollection benennt einen Tag, der den Cache für die Collection flusht, wenn einer seiner Nachkommen (direkte und indirekte) geändert wird. Damit wird die Collection geflusht, wenn sich eine Node innerhalb der collection ändert

Content Cache - Default Konfiguration

- **Die folgenden Objekte sind per Default gecached:**
 - TYPO3.Neos:Breadcrumb
 - TYPO3.Neos:Menu
 - TYPO3.Neos:Page
 - TYPO3.Neos:PrimaryContent
- **Die folgenden Objekte sind per Default NICHT gecached:**
 - TYPO3.Neos.NodeTypes:Form
 - TYPO3.Neos:Plugin
- **Warnung:**
 - Der TYPO3.Neos:ContentCollection Prototype wird per Default eingebettet, hat aber eine Cache Konfiguration mit einem Definition für Identifier, Tags und einer maximumLifetime. Für jede fixe ContentCollection auf einer Seite (z.B. einer Sidebar oder ein Footer) muss das Caching explizit eingeschaltet werden (`@cache.mode = 'cached'`) um ein korrektes Flush-Verhalten zu bekommen. Für NodeTypes wie multi-column besteht die Notwendigkeit nicht

Content Cache - Default Cache-Konfiguration überschreiben

- **Man kann per TypoScript die Default-Konfiguration auch überschreiben:**

```
prototype(TYPO3.Neos:PrimaryContent).@cache.mode = 'uncached'
```

- **Ebenfalls möglich ist das Überschreiben für einen spezifischen Pfad:**

```
page.content.main {  
    prototype(TYPO3.Neos:Plugin).@cache.mode = 'cached'  
}
```

- **Einschalten des Caches für eine zusätzliche, fixe ContentCollection:**

```
page {  
    content {  
        teaser = ContentCollection {  
            nodePath = 'teaser'  
            @cache {  
                mode = 'cached'  
            }  
        }  
    }  
}
```

Content Cache - Cache Backend ändern

- Normalerweise werden die Cache-Einträge im Filesystem gespeichert
- Dies kann aber in der Datei `Configuration/Caches.yaml` geändert werden

```
TYPO3_TypoScript_Content:  
backend: TYPO3\Flow\Cache\Backend\RedisBackend
```

Content Dimensionen

Content Dimensionen - Einführung

- „**Content dimensions**“ (Content Dimensionen) stellen das allgemeine Konzept dar, um mehrere **Varianten** einer Node zu haben. Eine Dimension ist dabei so etwas wie „language“, „country“ or "customer segment". Das Content Repository unterstützt eine beliebige Anzahl an Dimensionen.
- Node Varianten können mehrere Werte für jede Dimension besitzen und werden durch den selben **Identifier** verbunden. Dies ermöglicht einen Single-Tree-Ansatz für Lokalisierung, Personalisierung oder jeder anderen Variation an Content auf einer Seite
- Wenn Inhalt gerendert und dafür aus dem Content Repository geholt wird, geschieht dies immer innerhalb eines „**Kontext**“. Dieser Kontext enthält eine Liste an Werten für jede Dimension, die angibt, welche Werte sichtbar sind und welcher „**FallBack-Reihenfolge**“ diese angewendet werden sollen
- So können die selben Node-Varianten zu unterschiedlichen Ergebnissen - abhängig vom Kontext - führen
- „**Dimension presets**“ fügen einen Namen zur Liste der Dimension-Werte hinzu und werden verwendet um Dimensionen im User Interface oder im Routing anzuzeigen. Sie repräsentieren die erlaubten Kombinationen der Dimensions-Werte

Content Dimensionen - Konfiguration

- „Content dimensions“ (Content Dimensionen) - Settings.yaml

TYPO3:

TYPO3CR:

```
contentDimensions:  
  'language':  
    # Default Dimension, die beim Anlegen von Nodes verwendet wird  
    default: 'mul_ZZ'  
    # Default Setting wenn kein URL Segment übermittelt wird  
    defaultPreset: 'all'  
    label: 'Language'  
    icon: 'icon-language'  
    presets:  
      'all':  
        label: 'All languages'  
        values: ['mul_ZZ']  
        uriSegment: 'all'  
      'en_GB':  
        label: 'English (Great Britain)'  
        values: ['en_GB', 'en_ZZ', 'mul_ZZ']  
        uriSegment: 'gb'  
      'de':  
        label: 'German (Germany)'  
        values: ['de_DE', 'de_ZZ', 'mul_ZZ']  
        uriSegment: 'de'
```

Content Dimensionen - Routing & Limitationen

- TYPO3CR und Neos enthalten keinerlei Default Dimension Konfiguration
- Neos stellt einen „route- part handler“ zur Verfügung, der einen Prefix mit dem Wert des „uriSegment“ Settings des Dimension-Presets inkludiert. Die URI enthält erst dann einen Prefix, wenn eine Content-Dimension konfiguriert wurde. Multiple Dimensionen werden über _ (Underscore) definiert - dadurch folgt, dass das „uriSegment“ Settings keinen _ enthalten darf.
- In Neos 1.2 können Node-Varianten nur mit einem allgemeinen Fallback-Wert in ihren Presets erzeugt werden. Damit kann eine Node nur dann in eine andere Dimension „übersetzt“ werden, wenn der Fallback-Wert „durchscheint“
- In Neos 1.3 ist es möglich, Node-Varianten über die Dimension-Grenzen hinweg zu erzeugen. Damit kann man beispielsweise ein Englische Seite in Deutsch übersetzen, ohne einen Fallback von Deutsch nach Englisch (und umgekehrt) zu haben.

Neuaufbau einer TYPO3 Neos Website

Startpunkt

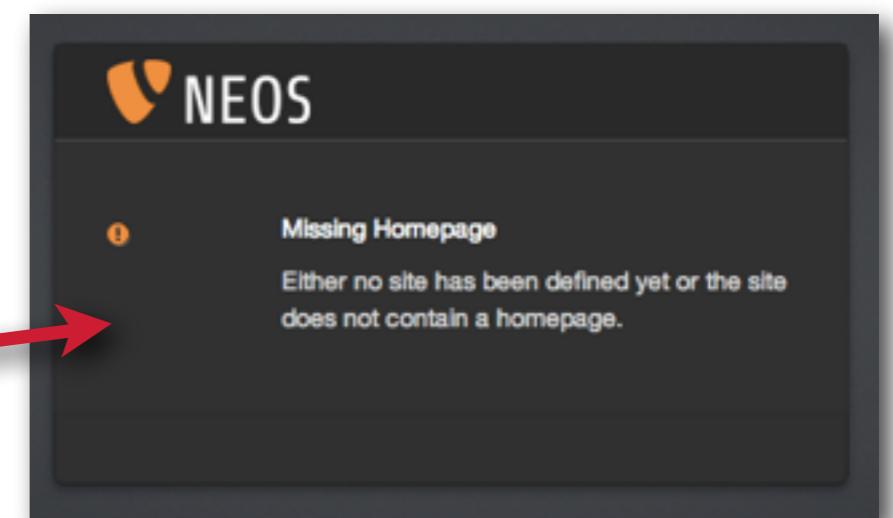
- Entweder installiert man TYPO3 Neos komplett neu
- Oder (wenn man ein Testsystem „zurücksetzen“ will): Man löscht die Datenbank und legt diese erneut an
- Anschließend führt man den folgenden Befehl aus, der alle notwendigen Datenbank-Tabellen (und Daten) wieder herstellt:

```
./flow doctrine:migrate
```

- Schließlich sollte man den Cache löschen

```
./flow flow:cache:flush
```

- Ein Aufruf im Frontend sollte folgendes Ergebnis bringen:



User anlegen

- Damit man sich später einloggen kann, brauchen wir einen Benutzer
- Diesen legen wir über die Kommandozeile an
- `./flow user:create --roles Administrator admin password firstname lastname`
- SYNTAX:
`./flow user:create [<options>] <username> <password> <first name> <last name>`

ARGUMENTE:

<code>--username</code>	Username des Users, der erstellt wird
<code>--password</code>	Passwort des Users, der erstellt wird
<code>--first-name</code>	Vorname des Users, der erstellt wird
<code>--last-name</code>	Nachname des Users, der erstellt wird

OPTIONEN:

<code>--roles</code>	Kommaseparierte Liste der Rollen, die der User bekommen soll
----------------------	--

Site anlegen

- Ein Website kann auf zwei Arten initialisiert werden
 - Über den Site-Manger innerhalb von Neos
 - Über die TYPO3 Flow Kommandozeile

```
cd /pfad/zur/Neos-Installation/
./flow kickstart:site Lobacher.Demo Website
./flow site:import --package-key Lobacher.Demo
```

- Syntax zum Anlegen

```
./flow kickstart:site <PackageKey> <SiteName>
```
- Syntax zum Importieren der Site in TYPO3 Neos

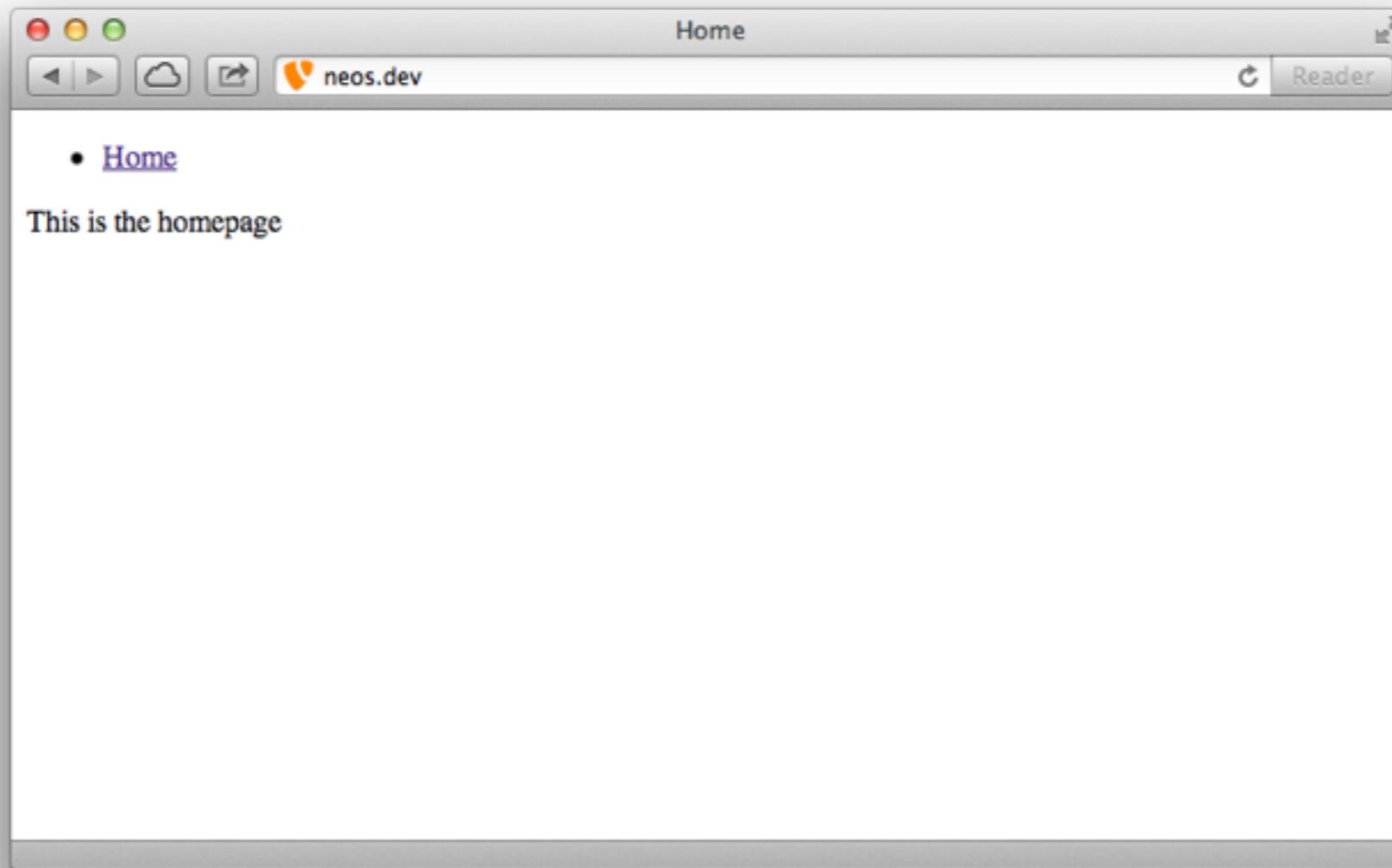
```
./flow site:import --package-key <PackageKey>
```

(Import führt zu einem Datensatz in der Tabelle `typo3_neos_domain_model_site`)
- Syntax zum Anzeigen aller Sites

```
./flow site:list
```

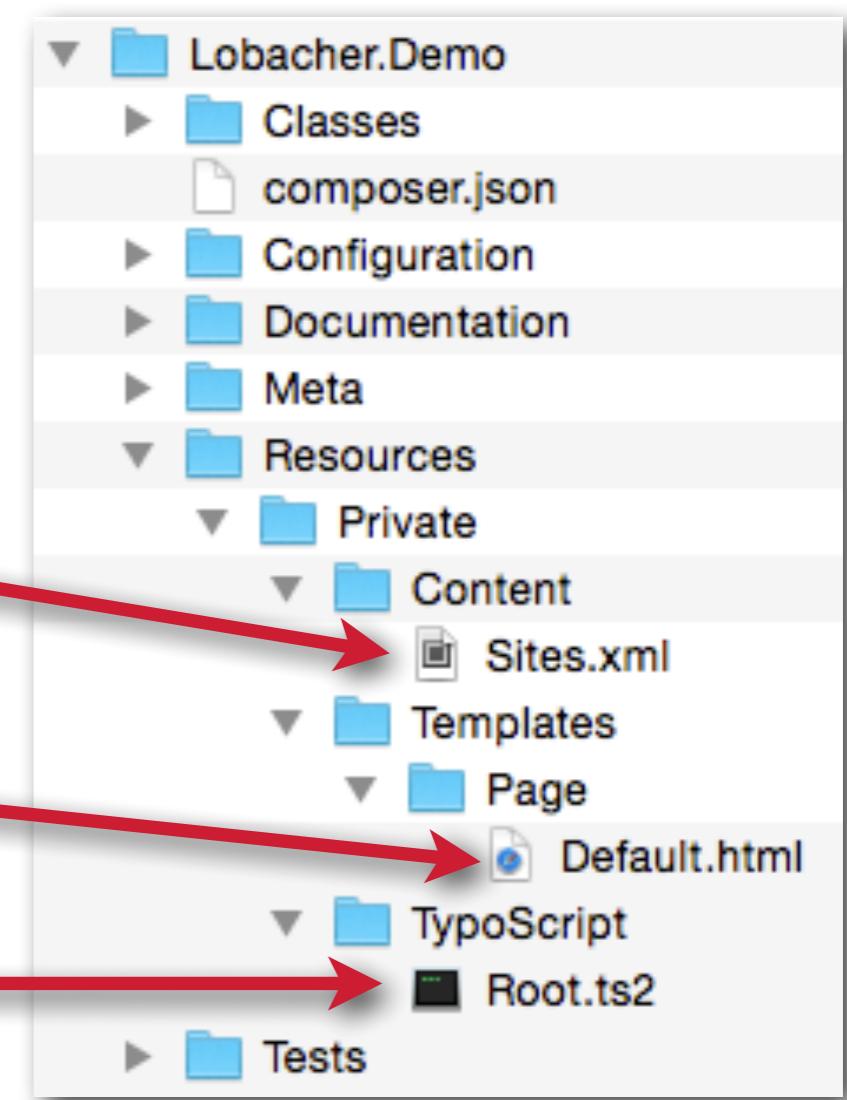
Site anlegen - Frontend

- Ein Aufruf der Website sollte dann folgenden Output liefern:



Site anlegen - angelegte Dateien

- Durch den Kickstarter wurde nun im Verzeichnis Packages/Sites folgende Struktur im Verzeichnis Lobacher.Demo inkl. der Dateien angelegt
- Sites.xml
Enthält die Root-Node
Definition der Website
- Default.html
Standard-Template
- Root.ts2
Default TypoScript Code



Site anlegen - angelegte Dateien - Sites.xml

- **Teil 1:** Die Sites.xml enthält die Root-Node Definition

```
<node nodeName="subpage">
    <variant workspace="live" nodeType="TYPO3.Neos.NodeTypes:Page" sortingIndex="100" version="1" removed="" hidden=""
hiddenInIndex="">
        <dimensions/>
        <accessRoles __type="array">[]</accessRoles>
        <properties>
            <title __type="string">A sub page</title>
        </properties>
    </variant>
    <node nodeName="main">
        <variant workspace="live" nodeType="TYPO3.Neos:ContentCollection" sortingIndex="100" version="1" removed="" hidden=""
hiddenInIndex="">
            <dimensions/>
            <accessRoles __type="array">[]</accessRoles>
        </variant>
        <node nodeName="text1">
            <variant workspace="live" nodeType="TYPO3.Neos.NodeTypes:Text" sortingIndex="100" version="1" removed="" hidden=""
hiddenInIndex="">
                <dimensions/>
                <accessRoles __type="array">[]</accessRoles>
                <properties>
                    <text __type="string">&lt;p&gt;This is the first sub page&lt;/p&gt;</text>
                </properties>
            </variant>
        </node>
    </node>
</node>
</nodes>
</site>
```

Site anlegen - angelegte Dateien - Sites.xml

- **Teil 2:** Die Sites.xml enthält die Root-Node Definition

```
<?xml version="1.0" encoding="UTF-8"?>
<root>
  <site name="Website" state="1" siteResourcesPackageKey="Lobacher.Demo" sitenodeName="demo">
    <nodes formatVersion="2.0">
      <node nodeName="demo">
        <variant workspace="live" nodeType="TYPO3.Neos.NodeTypes:Page" sortingIndex="100" version="1" removed="" hidden=""
hiddenInIndex="">
          <dimensions/>
          <accessRoles __type="array">[]</accessRoles>
          <properties>
            <title __type="string">Home</title>
          </properties>
        </variant>
        <node nodeName="main">
          <variant workspace="live" nodeType="TYPO3.Neos:ContentCollection" sortingIndex="100" version="1" removed="" hidden="" hiddenInIndex="">
            <dimensions/>
            <accessRoles __type="array">[]</accessRoles>
          </variant>
          <node nodeName="text1">
            <variant workspace="live" nodeType="TYPO3.Neos.NodeTypes:Text" sortingIndex="100" version="1" removed="" hidden="" hiddenInIndex="">
              <dimensions/>
              <accessRoles __type="array">[]</accessRoles>
              <properties>
                <text __type="string">&lt;p&gt;This is the homepage&lt;/p&gt;</text>
              </properties>
            </variant>
          </node>
        </node>
      </nodes>
    </site>
  </root>
```

Site anlegen - MultiSite

- Wenn das System nun lediglich eine Site enthält, kann man diese bereits über die URL aufrufen
- Sind jedoch mehrere Sites enthalten, so müssen diese mit Hilfe von Domain-Einträgen konfiguriert werden
- Zunächst wird mittels `./flow site:list` die Liste aller Sites ermittelt

```
pats-macbook-air-2:TYPO3-Neos patricklobacher$ ./flow site:list
```

Name	Node name	Resources package
<hr/>		
TYPO3 Neos Demo Site	neosdemotypo3org	TYPO3.NeosDemoTypo3Org
Demo	demo	Lobacher.Demo
LobacherTest	test	Lobacher.Test

Site anlegen - MultiSite

- Nun fügen Sie für jeden NodeName (deren Site Sie später ansprechen wollen) einen Domain-Eintrag hinzu

```
./flow domain:add test neos.dev
```

- Syntax
 - ./flow domain:add <node name> <domain>
- Auflisten kann man die Domains mittels ./flow domain:list
- Löschen Sie anschließend den Cache
 - ./flow flow:cache:flush --force

Minimales TypoScript

- Minimal benötigt wird folgendes TypoScript - dieses sorgt dafür, dass die Template-Datei eingelesen und als (leere) HTML-Website ausgegeben wird

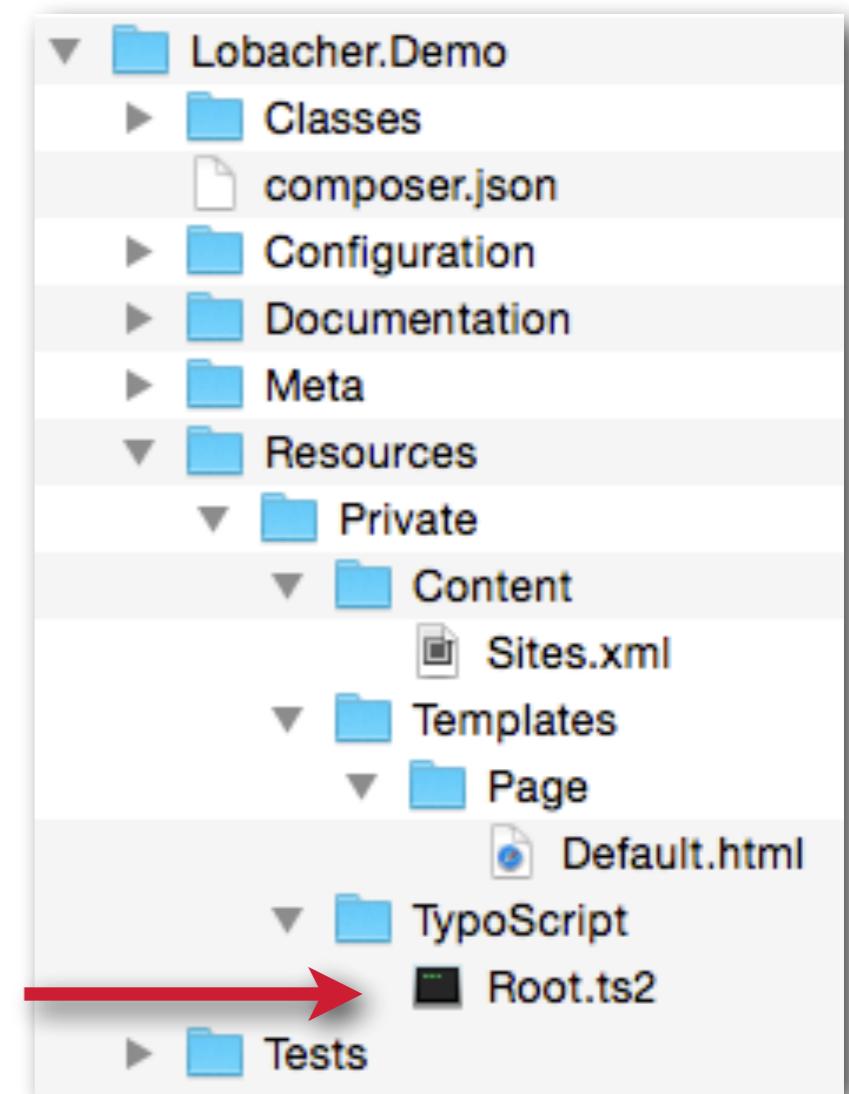
```
page = Page
page.body.templatePath =
    'resource://Lobacher.Demo/Private/Templates/Page/Default.html'
```

- Oder man verwendet den ViewHelper {parts.menu -> f:format.raw() }, welcher dafür sorgt, dass der TypoScript-Pfad page.body.parts.menu gerendert wird.

Analyse Root.ts2

- Die Datei Root.ts2 (TypoScript 2.0) wird nun als erstes geparsst und ausgeführt
- Zuerst wird ein Objekt mit dem Namen „page“ erstellt, welchem das Page-Objekt zugewiesen wird

```
page = Page
```
- Das Objekt Page hat unter anderem zwei Eigenschaften:
 - head (vom Typ TYPO3.TypoScript:Array)
 - body (vom Typ TYPO3.Neos:Template)
- Diese wurden in
TYPO3.Neos/Resources/Private/
TypoScript/DefaultTypoScript.ts2
festgelegt



Analyse Root.ts2 - Header-Bereich

- Als erstes wird der Header-Bereich der Website gefüllt, einmal mit den Stylesheets und einmal mit den benötigten JavaScripts.
- Dazu wird in beiden Fällen das Template geladen und die entsprechende Sektion angesprungen

```
page = Page {
    head {
        stylesheets.site = TYPO3.TypoScript:Template {
            templatePath = 'resource://Lobacher.Demo/Private/Templates/Page/
Default.html'
            sectionName = 'stylesheets'
        }
        javascripts.site = TYPO3.TypoScript:Template {
            templatePath = 'resource://Lobacher.Demo/Private/Templates/Page/
Default.html'
            sectionName = 'headScripts'
        }
    }
}
```

Analyse Root.ts2 - Header-Bereich - Template

- Im Template (`resource://Lobacher.Demo/Private/Templates/Page/Default.html`) sieht das dann wie folgt aus:

```
<head>
    <f:section name="stylesheets">
        <!-- Put your stylesheet inclusions here, they will be included in your website by TypoScript -->
    </f:section>
    <f:section name="headScripts">
        <!-- Put your scripts inclusions for the head here, they will be included in your website by TypoScript -->
    </f:section>
</head>
```

- Ausgegeben wird dann letztendlich nur der Code, der zwischen den beiden `<f:section>`-ViewHelpern steht. Das `<head>`-Tag wird von TYPO3 Neos selbst ausgegeben (aufgrund von `page.head.stylesheets.sectionName` und `page.head.scripts.sectionName`)

Analyse Root.ts2 - Body-Bereich & Template

- Weiter geht es mit der Befüllung des <body>-Bereichs - dafür wird ein Template geladen und dort die Section „body“ aufgerufen:

```
body {  
    templatePath = 'resource://Lobacher.Demo/Private/Templates/Page/  
Default.html'  
    sectionName = 'body'
```

- Das zugehörige Template an dieser Stelle sieht wie folgt aus:

```
<body>  
<f:section name="body">  
    ...  
</f:section>  
</body>
```

- Auch hier wird nur das gerendert, was sich zwischen dem <f:section>-ViewHelper befindet

Analyse Root.ts2 - Body-Bereich & Template

- Innerhalb von `page.body` sind nun zwei Objekte definiert - einerseits `menu` vom Objekttyp `Menu` und andererseits `breadcrumb` vom Objekttyp `BreadcrumbMenu`: (ACHTUNG: In der aktuellen Version 1.1 steht hier noch `Breadcrumb`)

```
parts {
    menu = Menu
    breadcrumb = BreadcrumbMenu
}
```

- Im Template greift man dann über die Pfade `parts.menu` bzw. `parts.breadcrumb` ZU:

```
<nav class="menu">
    {parts.menu -> f:format.raw() }
</nav>
<nav class="breadcrumb">
    {parts.breadcrumb -> f:format.raw() }
</nav>
```

Analyse Root.ts2 - Body-Bereich & Template - **Menu**

- Für das Rendern des Menüs via `menu = Menu` ist letztlich das TypoScript unter `TYPO3.Neos/Private/Templates/TypoScript/DefaultTypoScript.ts2` und infolgedessen das Template `templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/Menu.html'` zuständig:

```
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<ul{attributes -> f:format.raw()}>
    <f:render section="itemsList" arguments="{item: item}" />
</ul>

<f:section name="itemsList">
    <f:for each="{items}" as="item">
        <li{ts:render(path:'{item.state}.attributes') -> f:format.raw()}>
            <neos:link.node node="{item.node}">{item.label}</neos:link.node>
            <f:if condition="{item.subItems}">
                <ul>
                    <f:render section="itemsList" arguments="{items: item.subItems}" />
                </ul>
            </f:if>
        </li>
    </f:for>
</f:section>
```

Analyse Root.ts2 - Body-Bereich & Template - Menu

- Das TypoScript für den Prototyp Menu sieht wie folgt aus:

```
prototype(TYPO3.Neos:Menu) < prototype(TYPO3.TypoScript:Template) {
    @class = 'TYPO3\\Neos\\TypoScript\\MenuImplementation'
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/Menu.html'
    node = ${node}
    items = ${this.items}
    filter = 'TYPO3.Neos:Document'
    attributes = TYPO3.TypoScript:Attributes
    active.attributes = TYPO3.TypoScript:Attributes {
        class = 'active'
    }
    current.attributes = TYPO3.TypoScript:Attributes {
        class = 'current'
    }
    normal.attributes = TYPO3.TypoScript:Attributes {
        class = 'normal'
    }
    @exceptionHandler = 'TYPO3\\Neos\\TypoScript\\ExceptionHandlers\\NodeWrappingHandler'
    @cache {
        mode = 'cached'
        entryIdentifier {
            documentNode = ${documentNode}
            domain = ${site.context.currentDomain}
        }
        entryTags {
            1 = 'NodeType_TYPO3.Neos:Document'
        }
    }
}
```

Analyse Root.ts2 - Body-Bereich & Template - **Menu**

- Die Menu Klasse sieht (auszugsweise) wie folgt aus:

```
class MenuImplementation extends \TYPO3\TypoScript\TypoScriptObjects
\TemplateImplementation {
    const MAXIMUM_LEVELS_LIMIT = 100;
    const STATE_NORMAL = 'normal';
    const STATE_CURRENT = 'current';
    const STATE_ACTIVE = 'active';
    protected $items;
    protected $currentNode;
    public function getEntryLevel() {
        return $this->tsValue('entryLevel');
    }
    public function getMaximumLevels() {
        $maximumLevels = $this->tsValue('maximumLevels');
        if ($maximumLevels > self::MAXIMUM_LEVELS_LIMIT) {
            $maximumLevels = self::MAXIMUM_LEVELS_LIMIT;
        }
        return $maximumLevels;
    }
    ...
}
```

Analyse Root.ts2 - Body-Bereich & Template - Breadcrumb

- Für das Rendern des Menüs via `breadcrumb = BreadcrumbMenu` ist letztlich das TypoScript unter `TYPO3.Neos/Private/Templates/TypoScript/DefaultTypoScript.ts2` und infolgedessen das Template `templatePath = 'resource://TYPO3.Neos/Private/Templates/TypoScriptObjects/BreadcrumbMenu.html'` zuständig:

```
{namespace neos=TYPO3\Neos\ViewHelpers}
{namespace ts=TYPO3\TypoScript\ViewHelpers}
<f:if condition="{items}">
    <ul{attributes -> f:format.raw()}>
        <f:for each="{items}" as="item" reverse="TRUE">
            <li{ts:render(path:'{item.state}.attributes') -> f:format.raw()}>
                <f:if condition="{item.state} == 'current'">
                    <f:then>{item.label}</f:then>
                    <f:else>
                        <neos:link.node node="{item.node}">{item.label}</neos:link.node>
                    </f:else>
                </f:if>
            </li>
        </f:for>
    </ul>
</f:if>
```

Analyse Root.ts2 - Body-Bereich & Template - **Breadcrumb**

- Das TypoScript für den Prototyp Breadcrumb sieht wie folgt aus:

```
# TYPO3.Neos:BreadcrumbMenu provides a breadcrumb navigation based on menu items.  
#  
prototype(TYPO3.Neos:BreadcrumbMenu) < prototype(TYPO3.Neos:Menu) {  
    templatePath = 'resource://TYPO3.Neos/Private/Templates/TyposcriptObjects/  
BreadcrumbMenu.html'  
    itemCollection = ${q(node).add(q(node).parents(' [instanceof  
TYPO3.Neos:Document]')).get()}  
    attributes.class = 'breadcrumb'  
}
```

- Hier gibt es keine eigene Menü-Klasse in PHP

Analyse Root.ts2 - Body-Bereich & Template

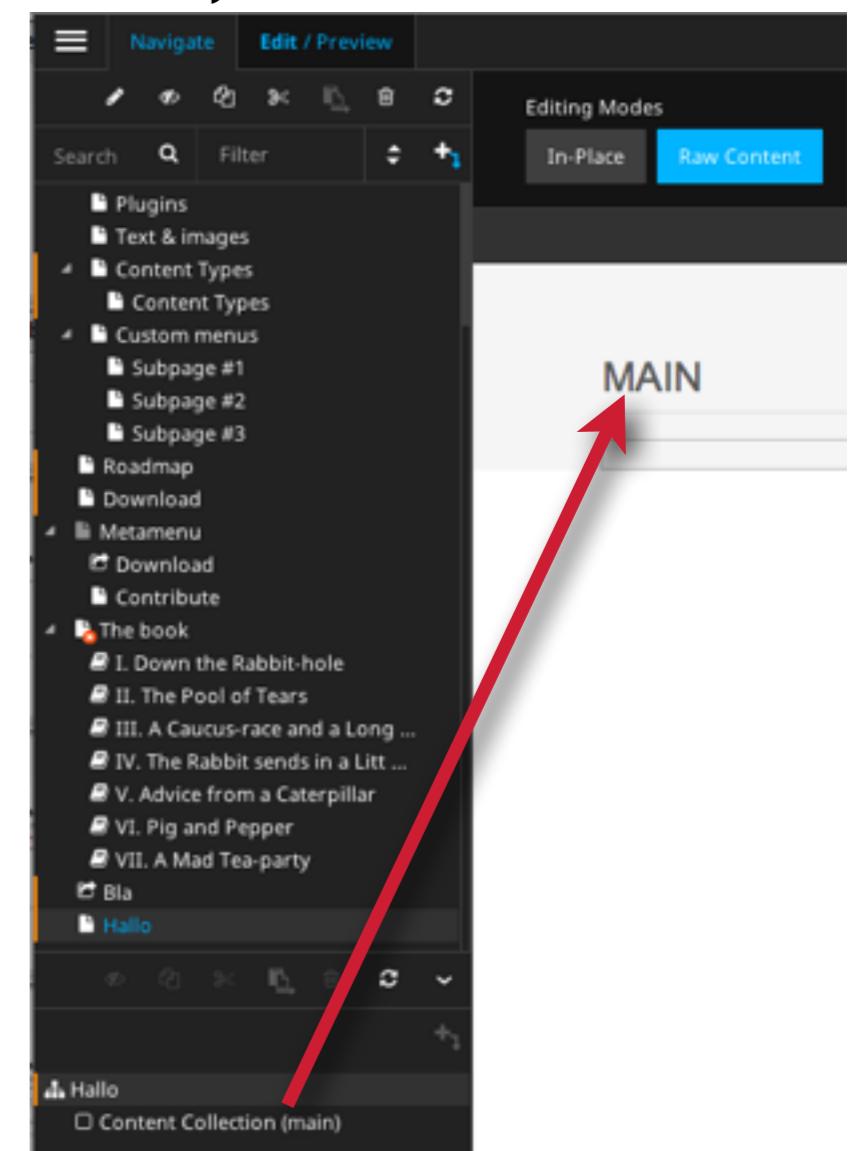
- Innerhalb von `page.body` wird nun auch noch das Content-Objekt definiert:

```
page.body {  
    content {  
        // The default content section  
        main = PrimaryContent {  
            nodePath = 'main'  
        }  
    }  
}
```

- Im Template greift man dann über den Pfad `content.main` zu:

```
<div class="content">  
{content.main -> f:format.raw() }</div>
```

- Die erste Content Collection wird automatisch mit **main** benannt - alle weitere kann man selbst festlegen.



Erweiterung der Site

Einfaches Plugin 1

Plugin: Meta-Tag zur Angabe einer Canonical URL

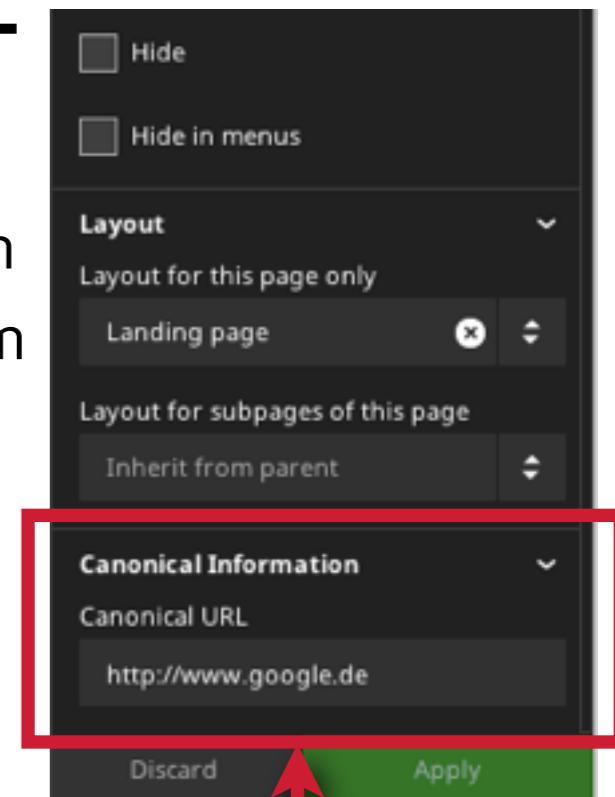
- **Ziel:** TYPO3 Neos soll so erweitert werden, dass es möglich ist, im Backend eine Canonical URL anzugeben, welche dann wie folgt im Frontend gerendert wird:

```
<link rel="canonical" href="http://www.lobacher.de/canonical" />
```

- Dafür benötigen wir ein Flow Package, welches wir als Neos Plugin verwenden

- **Schritt 1: Anlegen eines Flow Package** - der Vendor ist hier „Lobacher“, der Name des Packages ist „Canonical“

./flow kickstart:package Lobacher.Canonical



Plugin: Meta-Tag zur Angabe einer Canonical URL

- **Schritt 2: Zufügen einer Datei „NodeTypes.yaml“ in Packages/Application/**

Lobacher Canonical/Configuration/

```
# Hinzufuegen des Feldes zu den Seiteneigenschaften
'TYPO3.Neos.NodeTypes:Page':
    superTypes: ['TYPO3.Neos:Document']
    properties:
        canonicalUrl:
            type: string
            ui:
                label: 'Canonical URL'
                reloadIfChanged: TRUE
                inspector:
                    group: 'metaOptions'
            ui:
                inspector:
                groups:
                    metaOptions:
                        label: 'Canonical Information'
                        position: 180
```

WICHTIG:

Die Einrückungen
werden mit je
2 Leerzeichen
durchgeführt!

Plugin: Meta-Tag zur Angabe einer Canonical URL

- Schritt 3: Anlegen des TypoScript Verzeichnisses unterhalb von Resources

```
mkdir -p Packages/Application/Lobacher.Canonical/Resources/Private/TypoScripts
```

- Schritt 4: Dort wird die Datei Root.ts2 angelegt, welche den Head-Bereich der Website mit einem Template erweitert, in welchem die CanonicalUrl angezeigt wird

```
page.head.metadata {
    canonicalTag = TYPO3.TypoScript:Template
    canonicalTag {
        templatePath = 'resource://Lobacher.Canonical/Private/Templates/
TypoScript/CanonicalTag.html'
        canonicalUrl = TYPO3.TypoScript:Value
        canonicalUrl.value = ${q(node).property('canonicalUrl')}
    }
}
```

Plugin: Meta-Tag zur Angabe einer Canonical URL

- Schritt 5: Anlegen des TypoScript Verzeichnisses unterhalb von Templates

```
mkdir -p Packages/Application/Lobacher.Canonical/Resources/Private/Templates/  
TypoScript
```

- Schritt 6: Dort wird die Datei CanonicalTag.html angelegt

```
{namespace ts = TYPO3\TypoScript\ViewHelpers}  
<link rel="canonical" href="{ts:render(path: 'canonicalUrl')}" />
```

- Übung für den Leser: Das Tag nur dann ausgeben lassen, wenn auch eine Canonical URL eingegeben wurde

Plugin: Meta-Tag zur Angabe einer Canonical URL

- Schritt 7: Einfügen des TypoScripts in der Datei (möglichst am Anfang)
`Packages/Sites/Lobacher.Demo/Resources/Private/TypoScript/Root.ts2`

```
include: resource://Lobacher.Canonical/Private/TypoScript/Root.ts2
```

- Und weiter unten in der selben Datei (rote Zeilen hinzufügen)

```
page = Page {
  head {
    metadata = TypoScript:Template {
      templatePath = 'resource://Lobacher.Demo/Private/Templates/Page/
Default.html'
      sectionName = 'metadata'
    }
    stylesheets = TypoScript:Template
    ...
  }
}
```

Plugin: Meta-Tag zur Angabe einer Canonical URL

- **Hintergrundwissen:**

Die Seiteneigenschaften werden in der Tabelle

`typo3_typo3cr_domain_model_nodedata`

im Feld

`properties`

gespeichert:

```
a:7:{s:4:"name";s:20:"TYPO3 Neos Demo Site";s:5:"title";s:4:"Home";s:6:"layout";s:11:"landingPage";s:13:"subpageLayout";s:0:"";s:5:"state";s:1:"1";s:23:"siteResourcesPackageKey";s:22:"Lobacher.Demo";s:12:"canonicalUrl";s:20:"http://www.google.de";}
```

Plugin: Meta-Tag zur Angabe einer Canonical URL

- Schritt 8: Einfügen eines Markers ins Seitentemplate

Packages/Sites/Lobacher.Demo/Resources/Private/Templates/Page/Default.html

```
...
<head>
    <f:section name="metadata">
        ...
        {canonicalTag -> f:format.raw()}
    </f:section>
    ...
</head>
....
```

Erweiterung der Site

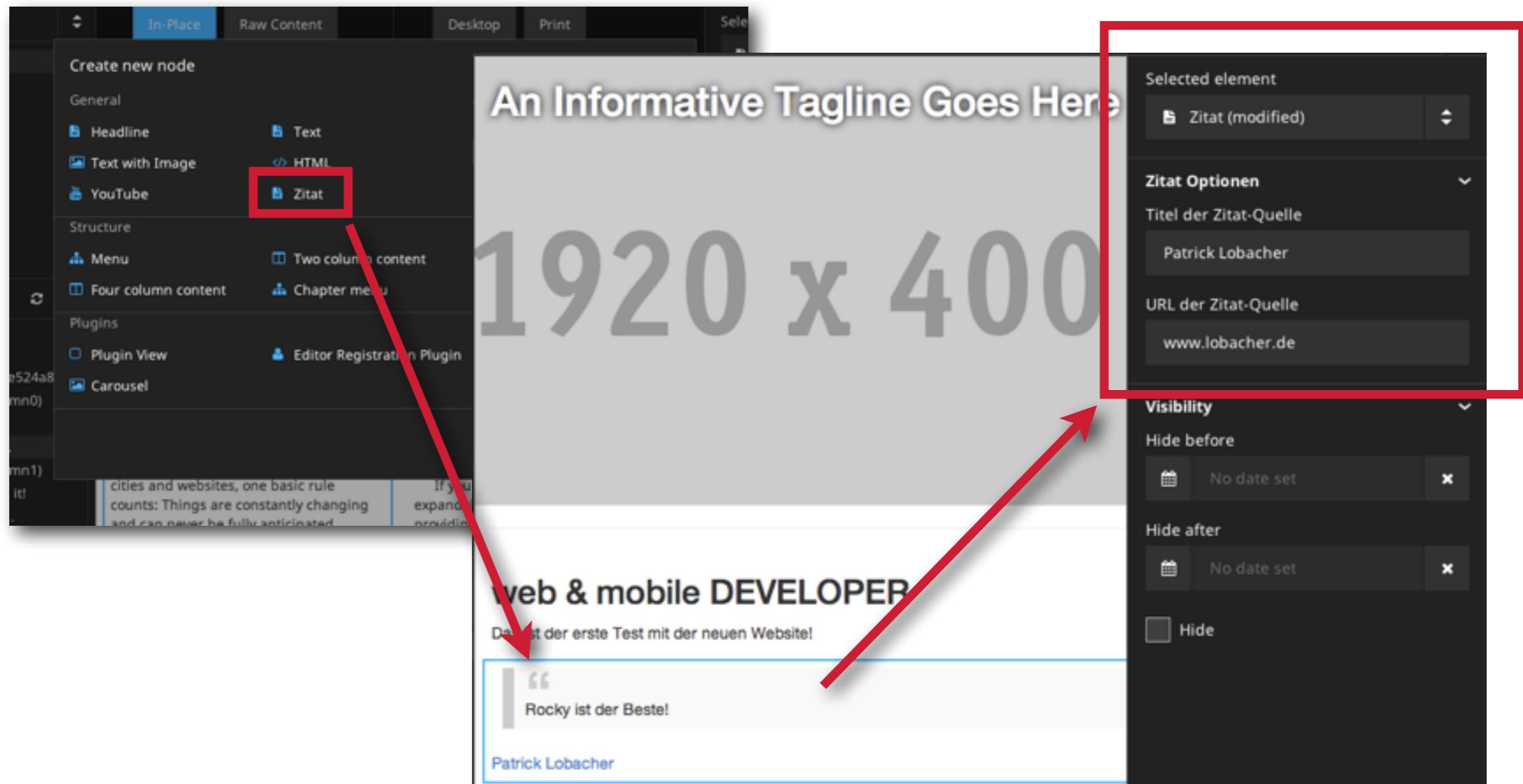
Einfaches Plugin 2

Plugin: Quote-FCE (Flexible Content Element)

- **Ziel:** TYPO3 Neos soll so erweitert werden, dass es möglich ist, ein Zitat-Content-Element einzufügen - bestehend aus einem Zitat-Text (Inline editierbar), einer Quelle und einer URL (beide über den Inspektor pflegbar)
- Dafür verwenden wir ein Flow Package, welches wir als Neos Plugin verwenden
- **Schritt 1: Anlegen eines Flow Package** - der Vendor ist hier „Lobacher“, der Name des Packages ist „Quote“

`./flow kickstart:package Lobacher.Quote`

Plugin: Quote-FCE (Flexible Content Element)



Plugin: Quote-FCE (Flexible Content Element)

- **Schritt 2: Hinzufügen einer Datei „`NodeTypes.yaml`“ in Packages/Application/Lobacher.Quote/Configuration/**

```
'Lobacher.Quote:Quote':  
    superTypes: ['TYPO3.Neos:Content']  
    ui:  
        label: 'Zitat'  
        inspector:  
            groups:  
                quoteproperties:  
                    label: 'Zitat Optionen'  
                    position: 5  
                icon: 'icon-file-text'
```

- **Weiter auf nächster Seite...**

WICHTIG:

Die Einrückungen werden mit je 2 Leerzeichen durchgeführt!

Plugin: Quote-FCE (Flexible Content Element)

- Schritt 2: ...weiterer Inhalt („properties“ ist 2 Leerzeichen eingerückt):

```
properties:  
  blockquote:  
    type: string  
    ui:  
      label: 'Zitat'  
      inlineEditable: TRUE  
      reloadIfChanged: TRUE  
  sourcetitle:  
    type: string  
    defaultValue: 'Titel der Zitat-Quelle'  
    ui:  
      label: 'Titel der Zitat-Quelle'  
  inspector:  
    group: 'quoteproperties'  
    reloadIfChanged: TRUE
```

WICHTIG:

Die Einrückungen werden mit je 2 Leerzeichen durchgeführt!

Plugin: Quote-FCE (Flexible Content Element)

- Schritt 2: ...weiterer Inhalt („sourceurl“ ist 4 Leerzeichen eingerückt):

```
sourceurl:  
    type: string  
    ui:  
        label: 'URL der Zitat-Quelle'  
        inspector:  
            group: 'quoteproperties'  
            reloadIfChanged: TRUE  
groups:  
    quoteproperties:  
    ui:  
        label: 'Zitat Optionen'  
        priority: 10
```

WICHTIG:

Die Einrückungen werden mit je 2 Leerzeichen durchgeführt!

Plugin: Quote-FCE (Flexible Content Element)

- Schritt 3: Anlegen des TypoScript Verzeichnisses unterhalb von Resources

```
mkdir -p Packages/Application/Lobacher.Quote/Resources/Private/TypoScripts
```

- Schritt 4: Dort wird die Datei Root.ts2 angelegt, welche das Quote-TS-Objekt definiert:

```
# Quote TypoScript Object
prototype(Lobacher.Quote:Quote) < prototype(TYPO3.Neos:Content) {
    templatePath = 'resource://Lobacher.Quote/Private/Templates/
TypoScriptObjects/Quote.html'
   blockquote = ${q(node).property('blockquote')}
    sourceurl = ${q(node).property('sourceurl')}
    sourcetitle = ${q(node).property('sourcetitle')}
}
```

Plugin: Quote-FCE (Flexible Content Element)

- Schritt 5: Anlegen des TypoScriptObjects Verzeichnisses unterhalb von Templates

```
mkdir -p Packages/Application/Lobacher.Quote/Resources/Private/Templates/  
TypoScriptObjects
```

- Schritt 6: Dort wird die Datei Quote.html angelegt

```
{namespace t=TYPO3\Neos\ViewHelpers}  
<t:contentElement node="{node}">  
    <blockquote cite="{f:uri.external(uri: '{sourceurl}', defaultScheme: 'http')}">  
        <t:contentElement.editable property="blockquote">  
            {blockquote}  
        </t:contentElement.editable>  
    </blockquote>  
    <f:if condition="{sourceurl}">  
        <f:then>  
            <f:link.external uri="{sourceurl}" defaultScheme="http">{sourcetitle}</  
f:link.external>  
        </f:then>  
        <f:else>  
            {sourcetitle}  
        </f:else>  
    </f:if>  
</t:contentElement>
```

Plugin: Quote-FCE (Flexible Content Element)

- Schritt 7: Einfügen des TypoScripts in der Datei (möglichst weit oben)
`Packages/Sites/Lobacher.Demo/Resources/Private/TypoScript/Root.ts2`

```
include: resource://Lobacher.Quote/Private/TypoScript/Root.ts2
```

- Schritt 8: Anlegen von zwei Verzeichnissen in der Demo-Site

```
mkdir Packages/Application/Lobacher.Quote/Resources/Public  
mkdir Packages/Application/Lobacher.Quote/Resources/Public/Stylesheets
```

In einem Schritt:

```
mkdir -p Packages/Application/Lobacher.Quote/Resources/Public/Stylesheets
```

Plugin: Quote-FCE (Flexible Content Element)

- Schritt 9: Anlegen einer Datei `Quote.css` im Verzeichnis:

`Packages/Application/Lobacher.Quote/Resources/Public/Stylesheets`

```
blockquote {  
    background:#f9f9f9;  
    border-left:10px solid #ccc;  
    margin:1.5em 10px;  
    padding:.5em 10px;  
    quotes:"\201C""\201D""\2018""\2019";  
}  
blockquote:before {  
    color:#ccc;  
    content:open-quote;  
    font-size:4em;  
    line-height:.1em;  
    margin-right:.25em;  
    vertical-align:-.4em;  
}  
blockquote p {  
    display:inline;  
}
```

Plugin: Quote-FCE (Flexible Content Element)

- Schritt 10: Referenz auf das Stylesheet in der Datei:

Packages/Sites/Lobacher.Demo/Resources/Private/Templates/Page/
Default.html

```
<f:section name="stylesheets">
    <!-- put your stylesheet inclusions here, they will be included in your website
by TypoScript -->

    <link rel="stylesheet" href="{f:uri.resource(path: 'Stylesheets/Quote.css',
package: 'Lobacher.Quote')} " media="all" />

</f:section>
```

Neos

Kommandozeile

Neos Kommandozeile - Domain

- Die CLI (Command line interface) Kommandos werden immer zusammen mit ./f1ow ausgeführt - also z.B. ./f1ow typo3.neos:domain:add ...
- **typo3.neos:domain:add** (**Fügt eine Domain hinzu**)
 - **--site-node-name**
Node-Name der Root-Node z.B. neostypo3org
 - **--host-pattern**
Host-Pattern z.B. neos.typo3.org
- **typo3.neos:domain:delete** (**Löscht eine Domain**)
 - **--host-pattern**
Host-Pattern, welcher entfernt werden soll z.B. neos.typo3.org
- **typo3.neos:domain:list** (**Listet alle Domains auf**)
 - **--host-pattern**
Optionales Host-Pattern für die Suche z.B. neos.typo3.org

Neos Kommandozeile - Site

- Die CLI (Command line interface) Kommandos werden immer zusammen mit ./f1ow ausgeführt - also z.B. ./f1ow typo3.neos:domain:add ...
- **typo3.neos:site:export** (**E**xportieren einer Site in ein XML Format)
- Argumente
 - **--site-name**
Name der Site, die exportiert werden soll - gibt man nichts an, werden alle Sites exportiert
- **typo3.neos:site:import**
- Argumente
 - **--package-key**
Package-Key welcher den Seiten-Inhalt importiert bekommen soll
 - **--file-name**
Dateinamen der XML-Datei, die den Inhalt hat

Neos Kommandozeile - Site

- Die CLI (Command line interface) Kommandos werden immer zusammen mit ./f1ow ausgeführt - also z.B. ./f1ow typo3.neos:domain:add ...
- **typo3.neos:site:list** (Auflisten aller Sites)
- Argumente
 - **keine**
- **typo3.neos:site:prune** (Löschen des Inhalts einer Site)
- Optionen
 - **--confirmation**
Fragt vor dem Löschen, ob man dies wirklich durchführen will

Neos Kommandozeile - User/Role

- Die CLI (Command line interface) Kommandos werden immer zusammen mit ./f1ow ausgeführt - also z.B. ./f1ow typo3.neos:domain:add ...
- **typo3.neos:user:addrole** (**Einem User eine Rolle hinzufügen**)
 - Argumente
 - **--username**
Username zu dem man die Rolle zufügen will
 - **--role**
Rolle, die man zum User zufügen will: "TYPO3.Neos:Editor" oder "TYPO3.Neos:Administrator"
- **typo3.neos:user:removerole** (**Eine User eine Rolle entfernen**)
 - Argumente
 - **--username**
Username, dessen Rolle man entfernen will
 - **--role**
Rolle, die man entfernen will: "TYPO3.Neos:Editor" oder "TYPO3.Neos:Administrator"

Neos Kommandozeile - User

- Die CLI (Command line interface) Kommandos werden immer zusammen mit ./f1ow ausgeführt - also z.B. ./f1ow typo3.neos:domain:add ...
- **typo3.neos:user:create** (Einen User anlegen)
- Argumente
 - **--username**
Username
 - **--password**
Password
 - **--first-name**
Vorname
 - **--last-name**
Nachname
- Optionen
 - **--roles**
Kommaseparierte Liste von Rollen, die der User bekommen soll

Neos Kommandozeile - User

- Die CLI (Command line interface) Kommandos werden immer zusammen mit ./f1ow ausgeführt - also z.B. ./f1ow typo3.neos:domain:add ...
- **typo3.neos:user:setpassword** (Passwort für einen User festlegen)
- Argumente
 - **--username**
Username, für den man das Passwort setzen will
 - **--password**
Das neue Passwort

Mitarbeit bei TYPO3 Neos

We need you!!

- Komme ins TYPO3 Neos und Flow Team!
- Jeder wird gebraucht: Programmierer, Architekten, UX-Experten, Designer, Doku, Tester, JS-Spezialisten, ...
- Meldet Euch bei robert [at] typo3.org



Kunden gesucht!

- **Direkte Mitarbeit**
 - Gebe frühes Feedback zu neuen Funktionen, User Interfaces, ...
 - Direkter Kontakt und Diskussion mit den Entwicklern
 - Sofern sinnvoll, wird das Feedback unmittelbar umgesetzt
- **Vorteile**
 - Gestalte das WCMS der Zukunft direkt mit
 - Nimm Einfluss auf die Entwicklung
 - Zugang zu Entwicklern, Architekten und UX'lern
 - Networking mit anderen Neos-Kunden
- **Kontakt**
 - [rasmus \[at\] typo3.org](mailto:rasmus@typo3.org)

Neos

Literatur

The Neos Book

- Im Februar 2014 wurde von Dan Frost eine Initiative gestartet, die sich „The Neos Book“ nennt
- Ziel soll es sein, ein (englischsprachiges) Buch zu TYPO3 Neos im (Spät)Sommer 2014 auf den Markt zu bringen
- Dafür werden mehr als 30 Core Entwickler und Autoren Inhalte beisteuern
- Die Finanzierung wird über [kickstarter.com](#) erfolgen
- Website: <http://www.theneosbook.com/>

Artikel

- **web & mobile DEVELOPER (Autor Patrick Lobacher)**
 - Ausgabe 11 / 2013: TypoScript, FlowQuery und Eel in TYPO3 Neos
 - Ausgabe 03 / 2014: TYPO3 Neos in Praxiseinsatz
 - Ausgabe 04 / 2014: Erweiterung von TYPO3 Neos
 - Ausgabe 05 / 2014: Plugin-Erstellung für TYPO3 Neos
- **Screenguide (Autor Patrick Lobacher)**
 - Ausgabe 20 / 2014: TYPO3 Neos - ein Überblick

Solltet Ihr weitere Artikel kennen - bitte mich mit Details kontaktieren!

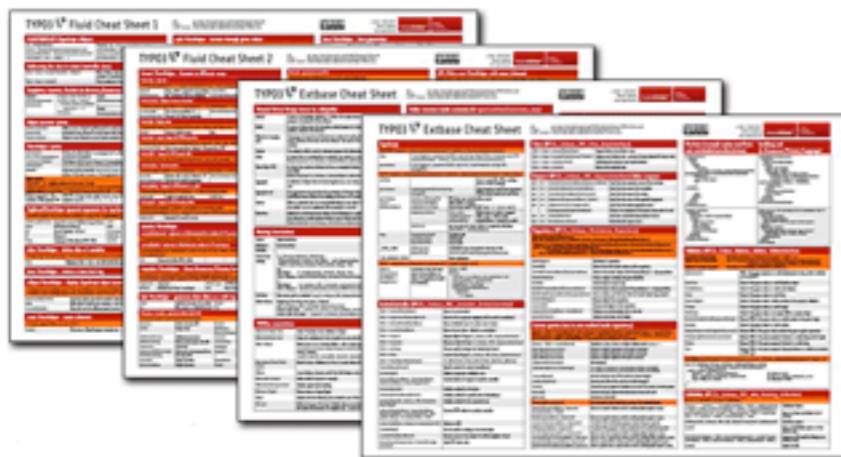
Quellen und Informationen

Quellen und Informationen

- **TYPO3 Neos Website**
<http://neos.typo3.org/>
- **TYPO3 Neos Download**
<http://neos.typo3.org/download.html>
- **TYPO3 Flow Website**
<http://flow.typo3.org/>
- **TYPO3 Neos Dokumentation**
<http://docs.typo3.org/neos/TYPO3NeosDocumentation/Index.html>
- **TYPO3 Flow Dokumentation**
<http://flow.typo3.org/documentation.html>

Quellen und Informationen

- TYPO3 Neos Projekt bei JIRA
<https://jira.typo3.org/browse/NEOS/>
- TYPO3 Flow Projekt bei JIRA
<https://jira.typo3.org/browse/FLOW/>
- Cheatsheet für Fluid (und demnächst für Flow)
http://www.lobacher.de/files/cs/FluidCheatSheet_3.01_Lobacher.pdf



Quellen und Informationen

- Learn Neos
<http://www.learn-neos.com>
- TypoScript 2 Pocket Reference
<http://learn-neos.com/reference/pocket-reference-typoscript2.html>
- GitHub Account von Lelesys (Pankaj Lele / <http://www.lelesys.com/>)
<https://github.com/lelesys>
- IRC Log vom Channel #typo3-neos bei Freenode
<http://riesvantwisk.com/cms/home/irc-logs/typo3-neos-irc-log.html>

Fragen?

Oder komplett verwirrt?



Patrick Lobacher

Geschäftsführer +Pluswerk GmbH

- 44 Jahre, glücklich verheiratet, wohnhaft in München
- Selbständige im Bereich Webentwicklung seit 1994
- Autor von 10 Fachbüchern und > 50 Fachartikeln zum Thema TYPO3 und Webentwicklung
- Mitglied im TYPO3 Education Committee (Co-Lead)
- Speaker, Trainer, Consultant, Coach, Nerd

[+]pluswerk
Digitale Leidenschaft



Veröffentlichungen:



Pluswerk

10 Standorte in Deutschland

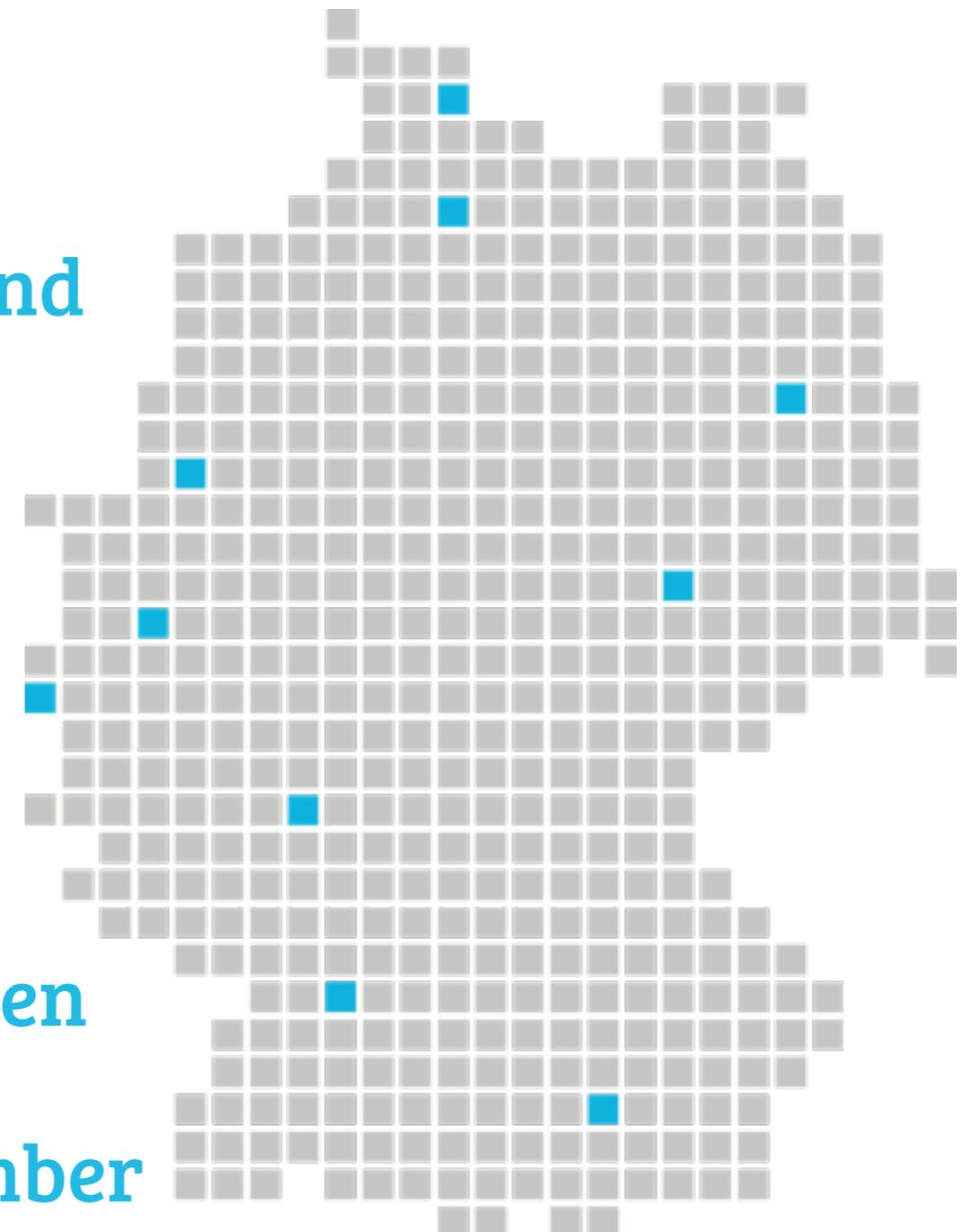
128 eigene Extensions

130 Mitarbeiter

5 Neos Websites

60 TYPO3 Experten

TYPO3 Association Platinum Member



Portfolio - Lösungen

Strategie & Consulting

- Digital Strategy
- E-Commerce Strategy
- Cross Channel Management
- Agile Consulting / Coaching
- Project Management

E-Commerce

- Online Shops
- Mobile Commerce
- B2C / B2B
- Social Commerce
- Commerce Search

Websites & Portale

- Corporate Websites
- Brand Sites
- Micro Sites
- Intranets & Extranets
- Mobile Applications

Performance Marketing

- Conversion Optimization
- Search Engine Marketing (SEM)
- Search Engine Optimization (SEO)
- Social Media Marketing (SMM)
- Web Analytics

Technologie & Systementwicklung

- CMS Development
- Enterprise Search
- CRM Development
- Systemintegration
- Software Development

Operations & Kreation

- Content Management
- Content Localization
- Shop Management
- Kreation / UI / UX
- Support & Hosting

Portfolio - Produkte



TYPO3



NEOS



FLOW



Magento®

eCommerce Platform for Growth



shopware®



OROCRM



akeneo



SUGARCRM

Referenzen

(Auszug: +Pluswerk/net-o-graphic/typofaktum/typovision)



Kontakt

Klassisch:

+Pluswerk GmbH

Patrick Lobacher
Solmstr. 6A
60486 Frankfurt am Main

Fon: +49 89 130 145 20

Email: lobacher@plus-werk.com
Web: www.plus-werk.com

Twitter:

www.twitter.com/PatrickLobacher

Facebook:

www.facebook.com/patrick.lobacher

Blog:

blog.lobacher.de

Schulungen:

www.lobacher.de

Google+:

plus.google.com/105500420878314068694

XING:

www.xing.com/profile/Patrick_Lobacher

LinkedIn

www.linkedin.com/pub/patrick-lobacher/4/881/171

Slideshare:

www.slideshare.net/plobacher

Amazon:

www.amazon.de/Patrick-Lobacher/e/B0045AQVEA



Digitale Leidenschaft

Vielen Dank für Eure
Aufmerksamkeit!