Learn Neos

TypoScript 2
Pocket Reference

for TYPO3 Neos 1.1

Table of Contents

TypoScript 2 • Syntax

Comments

```
# Line comment
// Another line comment
/*
 * Multi-line comment
 */
```

Simple Values

```
book = "The guide"
answer = 42
enableImprobabilityDrive = true
```

Expressions

```
answer = ${Math.floor(85 / 2)}
title = ${q(node).property('title')}
```

Objects

```
output = TYPO3.TypoScript:Array {
    hello = 'Hello'
    world = 'world!'
}
```

Property Definitions

```
output.world = "world!"
output {
    hello = "Hello"
}
prototype(Acme.Demo:Book).book = ${book}
```

Prototype Extension

```
prototype(Acme.Demo:Book) < prototype(Template) {
    templatePath = 'Book.html'
}
```

Clear Property

```
output.world >
```

Namespaces

```
namespace d=Acme.Demo
output = d:Book
```

@process	array	**Apply processors after evaluation**

```
example.@process.someProcessor = ${String.toUpperCase(value)}
```

@override	array	**Set context variables by named key**

```
example.@override.someVariable = ${q(node).parent()}
```

@position	mixed	**Order definitions in Array, @process, Case**

<numerical-order> | start (<weight>) | end (<weight>)
before <key> (<weight>) | after <key> (<weight>)

```
page.body.javascripts.jQuery.@position = 'start 100'
```

@cache	array	**Define caching behavior for object**
mode	string	'embed', 'cached' or 'uncached'
entryIdentifier	array	Named values to build the entry identifier
entryTags	array	Tags when the entry should be flushed
context	array	Context variable names to store in mode 'uncached'
maximumLifetime	integer	Maximum lifetime for nearest cached entry null (default), 0 (unlimited) or x seconds

```
cachedExample.@cache {
  mode = 'cached'
  entryIdentifier.node = ${node}
  entryTags.1 = ${'Node_' + node.identifier}
}
```

@class	string	**Set implementation class for object**

```
@class = 'Acme\\Demo\\TypoScript\\MyObjectImplementation'
```

@exceptionHandler	string	**Set exception handler class for object**

TYPO3.TypoScript

Template — Render a Fluid template

templatePath	string	The template path and filename (required)
partialRootPath	string	Path to partials
layoutRootPath	string	Path to layouts
sectionName	string	The Fluid <f:section> to render (if given)
[key]	mixed	Define a variable with name **key** in Fluid

```
prototype(My.Site:Footer) < prototype(TYPO3.TypoScript:Template) {
   templatePath = 'resource://My.Site/Private/Templates/Footer.html'
   sectionName = 'main'
   node = ${node}
}
```

Array — Render multiple ordered definitions and concatenate the results

[key]	string	A nested definition that evaluates to a string
[key].@position	mixed	Define the ordering of the element

```
output = TYPO3.TypoScript:Array {
   world = 'world!'
   hello = TYPO3.TypoScript:Value {
      value = 'Hello'
      @position = 'start'
   }
}
```

Value — Evaluate any value as a TypoScript object

value	mixed	The value to evaluate (required)

```
output = TYPO3.TypoScript:Value {
   value = 'A static text'
   @process.h1Tag = ${'<h1>' + value + '</h1>'}
}
```

Collection Render each item in a collection and concatenate the results

collection	array	The collection to render (Iterable, required)
itemName	string	Context variable name for each item (required)
iterationName	string	Context variable name for iteration information with properties index, cycle, isFirst, isLast
itemRenderer	string	The item rendering definition: simple value, expression or object (required)

```
output = TYPO3.TypoScript:Collection {
    collection = ${q(node).children()}
    itemName = 'childNode'
    itemRenderer = TYPO3.TypoScript:Template {
        templatePath = 'resource://...'
        node = ${childNode}
    }
}
```

RawArray Evaluate nested definitions as array

[key]	mixed	A nested definition
[key].@position	mixed	Define the ordering of the element

```
output = TYPO3.TypoScript:Attributes {
    class = TYPO3.TypoScript:RawArray {
        alert = 'alert'
        alert.@position = 'start'
        alert-error = 'alert-error'
    }
}
```

TYPO3.TypoScript

ResourceUri Build a URI to a package resource (e.g. JS or CSS)

path	string	Path to resource, either a path relative to Public and package or a resource:// URI
package	string	The package key (e.g. 'My.Package')
resource	Resource	Resource object instead of path and package
localize	boolean	Whether resource localization should be used, defaults to true

```
scriptInclude = TYPO3.TypoScript:Tag {
  tagName = 'script'
  attributes {
    src = TYPO3.TypoScript:ResourceUri {
      path = 'resource://My.Package/Public/Scripts/App.js'
    }
  }
```

UriBuilder Build a URI to a controller action

package	string	The package key (e.g. 'My.Package')
subpackage	string	The subpackage, defaults to empty
controller	string	The controller name (e.g. 'Registration')
action	string	The action name (e.g. 'new')
arguments	array	Arguments to the action by named key
format	string	An optional request format (e.g. 'html')
section	string	An optional fragment (hash) for the URI
additionalParams	array	Additional URI query parameters
addQueryString	boolean	Whether to keep current query parameters
argumentsToBeExcludedFromQueryString	array	Query parameters to exclude for addQueryString
absolute	boolean	Whether to create an absolute URI

```
uri = TYPO3.TypoScript:UriBuilder {
   package = 'My.Package'
   controller = 'Registration'
   action = 'new'
   absolute = true
}
```

Case **Conditionally render one of multiple object definitions or paths**

[key]	Matcher	
[key] .condition	boolean	The first condition that evaluates to true will be used (required)
[key].type	string	Object type to render (as string) (required)
[key].element	array	Properties for the object instance
[key].renderPath	string	Path to render (overrules type)
[key].@position		Define the ordering of the condition

```
page.body.content.header = TYPO3.TypoScript:Case {
   landingPage {
      condition = ${q(node).property('landingPage')}
      type = 'My.Package:SimpleMenu'
   }
   default {
      condition = true
      type = 'My.Package:MainMenu'
   }
}
```

TYPO3.TypoScript

Tag

Render an HTML tag with attributes and body (if given)

tagName	string	Tag name, defaults to div
content	string	The inner content of the element
attributes	string	Tag attributes, the Tag prototype defines an Attributes object by default
omitClosingTag	boolean	Whether to leave out the closing tag, defaults to false
selfClosingTag	boolean	Whether the tag is self closing, will be guessed from the tag name by default

```
output = TYPO3.TypoScript:Tag {
  tagName = 'meta'
  attributes {
    charset = 'UTF-8'
  }
}
```

Attributes

Render HTML tag attributes in an extensible way

[key]	mixed	The attribute name [key] and value, array values will be joined with whitespace

```
output = TYPO3.TypoScript:Attributes {
  id = 'message'
  class = TYPO3.TypoScript:RawArray {
    alert = 'alert'
    alert-error = 'alert-error'
  }
}
```

Page — Render an HTML document with metadata for Neos

doctype	string	HTML doctype, defaults to <!DOCTYPE html>
htmlTag	ts:Tag	The opening <html> tag
attributes.[key]	mixed	Attributes for <html> tag
headTag	ts:Tag	The opening <head> tag
head	ts:Array	Content inside <head>
titleTag	ts:Tag	The <title> tag
javascripts	ts:Array	Head JS includes (script tags)
stylesheets	ts:Array	Head CSS includes (link tags)
bodyTag	ts:Tag	The opening <body> tag
attributes.[key]	mixed	Attributes for <body> tag
body	ts:Template	The main body content
templatePath	string	Path to the body Fluid template (required)
[key]	mixed	Variables for the body template
javascripts	ts:Array	Body footer JS includes (script tags)

```
# Render content from main ContentCollection
# Output in Fluid template with {content.main -> f:format.raw()}
page.body {
  content.main = PrimaryContent {
    nodePath = 'main'
  }
}

# Include a stylesheet from a <f:section> in the template
page.head.stylesheets.mySite = TYPO3.TypoScript:Template {
    templatePath = 'resource://My.Package/Private/Page.html'
    sectionName = 'headStylesheets'
}
```

TYPO3.Neos

ContentCollection — Render content from a ContentCollection

nodePath	string	The relative node path of the Content-Collection (e.g. 'main') (required)
@override.contentCollectionNode	Node	The content collection node, resolved from nodePath by default
@cache	array	Content cache properties, 'mode' defaults to 'embed'

```
page.body {
   content {
      main = PrimaryContent {
         nodePath = 'main'
      }
      footer = ContentCollection {
         nodePath = 'footer'
         # Static ContentCollections must enable cache
         @cache.mode = 'cached'
      }
   }
}
```

PrimaryContent — Render the primary ContentCollection, extends Case

nodePath	string	The relative node path of the Content-Collection (e.g. 'main') (required)
default	array	Default case that renders a Content-Collection
[key]	array	Additional conditions (e.g. to handle specific document type), see TYPO3.TypoScript:Case

Content — Base type to render content nodes, extends Template

templatePath	string	The template path and filename, defaults to 'resource://[packageKey]/Private/Templates/NodeTypes/[nodeType].html'
[key]	mixed	Variables for the node template, all node type properties are set by default
attributes	ts:Attributes	Extensible attributes for the template

```
prototype(My.Package:MyContent) < prototype(TYPO3.Neos:Content) {
    templatePath = 'resource://My.Package/Template/Other.html'
    # Auto-generated for all node type properties
    # title = ${node.properties.title}
}
```

Plugin — Base type to render plugin nodes

package	string	The package key (e.g. 'My.Package') (required)
subpackage	string	The subpackage, defaults to empty
controller	array	The controller name (e.g. 'Registration')
action	string	The action name (e.g. 'new')
argumentNamespace	string	Namespace for action arguments, will be resolved from node type by default
[key]	mixed	Pass an internal argument to the action (access with _key)

```
prototype(My.Package:FlickrPlugin) < prototype(TYPO3.Neos:Plugin) {
    package = 'My.Package'
     controller = 'Flickr'
     action = 'tagStream'
}
```

Menu

Render a menu with items for document nodes

templatePath	string	Override the template path
entryLevel	integer	Start the menu at the given depth
maximumLevels	integer	Restrict the maximum depth of items in the menu (relative to entryLevel)
startingPoint	Node	The parent node of the first menu level (defaults to node context variable)
lastLevel	integer	Restrict the menu depth by node depth (relative to site node)
filter	string	Filter items by node type, defaults to TYPO3.Neos:Document
renderHidden-InIndex	boolean	Whether nodes with hiddenInIndex should be rendered, defaults to false
itemCollection	array	Explicitly set the Node items for the menu (alternative to startingPoints and levels)
attributes	ts:Attributes	Extensible attributes for the whole menu
normal.attributes	ts:Attributes	Attributes for *normal* state
active.attributes	ts:Attributes	Attributes for *activc* state
current.attributes	ts:Attributes	Attributes for *currcnt* state

Menu item properties

node	Node	A node that should be used to link to the item (with resolved shortcuts)
originalNode	Node	The unresolved node for the menu item
state	string	'normal', 'active' (the current node) or 'current' (parent of current node)
label	string	Full label of the node

- Eel expressions are valid JavaScript syntax
- JavaScript expression syntax can be used for Eel

Literals	`42 | 1.5 | false | true | null`
	`'A string' | "Another \"quoted\" string"`

Variabels	`${myVariable}`
Traversal	`${node.contextPath}`

Method Calls	`${String.substr("Hello world!", 6, 5)}`
	`${q(node).property('title')}`

Offset Access	`${String.split("Hello world!", ' ')[1]}`
	`${items[index + 1]}`

Mathematical operators	`+ | - | * | / | %`
Comparisons	`== | != | > | >= | < | <=`
Boolean operators	`! | && | ||`
Conditional operator	`${a ? b : c}`

Array and Object Literals	`${Array.join(['first', 'second', 'last'], ', ')}`
	`${q(node).context({'invisibleContentShown': true})}`

String.[function]

charAt (string, index)	Get the character at a specific position
crop (string, length, *suffix*)	Crop a string, optionally append suffix
cropAtSentence (string, length, *suffix*)	Crop a string, take sentences into account
cropAtWord (string, length, *suffix*)	Crop a string, take words into account
endsWith (string, search, *position*)	Test if a string ends with search string
firstLetterToLowerCase (string)	Lowercase the first letter of a string
firstLetterToUpperCase (string)	Uppercase the first letter of a string
htmlSpecialChars (string, *preserve*)	Convert special chars to HTML entities
indexOf (string, search, *fromIndex*)	Find the position of a substring
isBlank (string)	Test if a string is empty or whitespace only
lastIndexOf (string, search, *toIndex*)	Find the last position of a substring
md5 (string)	Calculate the MD5 hash of a string
pregMatch (string, pattern)	Match a regular expression (PREG)
pregReplace (string, pattern, replace)	Replace by regular expression (PREG)
rawUrlDecode (string)	Decode the string from URL encoding
rawUrlEncode (string)	Encode the string to URL encoding
replace (string, search, replace)	Replace occurrences of search string
split (string, *separator*, *limit*)	Split a string by a separator
startsWith (string, search, *position*)	Test if a string starts with search string
stripTags (string)	Strip all HTML tags from a string
substr (string, start, *length*)	Substring from start to the given length
substring (string, start, end)	Substring from start index to end index
toBoolean (string)	Convert a string to boolean
toFloat (string)	Convert a string to float
toInteger (string)	Convert a string to integer
toLowerCase (string)	Lowercase a string
toUpperCase (string)	Uppercase a string
toString (value)	Convert the given value to a string
trim (string, *charlist*)	Trim whitespace at beginning and end

Array.[function]

concat (array1, array2, *array*)	Concatenate arrays or values
first (array)	Get the first element of an array
indexOf (array, search, *fromIndex*)	Find the item position in an array
isEmpty (array)	Check if an array is empty
last (array)	Get the last element of an array
length (array)	Get the length of an array
join (array, *separator*)	Join values of an array with a separator
reverse (array)	Returns an array in reverse order
keys (array)	Get the array keys
pop (array)	Removes the last element from an array
push (array, element)	Insert elements at the end of an array
random (array)	Picks a random element from the array
shift (array)	Remove the first element of an array
shuffle (array, *preserveKeys*)	Randomize an array
slice (array, begin, *end*)	Extract a portion of an indexed array
sort (array)	Sorts an array (numbers>characters)
splice (array, offset, *length*, *replace*)	Replaces an array range by a replacement
unshift (array, element)	Insert elements at the beginning

Configuration.[function]

setting (settingPath)	Return the specified settings

Date.[function]

add (date, interval)	Add an interval to a date
dayOfMonth (dateTime)	Get the day of month of a date
diff (dateA, dateB)	Get the difference as a DateInterval
format (date, format)	Format a date (or interval) to a string
parse (string, format)	Parse a date from string with a format
hour (dateTime)	Get the hour of a date (24 hour format)
minute (dateTime)	Get the minute of a date
month (dateTime)	Get the month of a date
now ()	Get the current date and time
second (dateTime)	Get the second of a date
subtract (date, interval)	Subtract an interval from a date
today ()	Get the current date
year (dateTime)	Get the year of a date

Math.[function]

ceil (x)	Smallest integer greater than or equal to x
floor (x)	Largest integer less than or equal to x
max (x, y, ...)	Largest of given numbers
min (x, y, ...)	Smallest of given numbers
random ()	Random float between 0 (incl) and 1 (excl)
randomInt (min, max)	Random int between min and max (incl)
round (subject, *prccision*)	Rounds the subject to the given precision

For more Math functions
see docs.typo3.org/neos/

Don't panic

learn-neos.com

networkteam creative webprojects.