TYPO3.TypoScript - Feature # 39658

Status:	Resolved	**Priority:**	Should have
Author:	Sebastian Kurfuerst	**Category:**	
Created:	2012-08-09	**Assigned To:**	Sebastian Kurfuerst
Updated:	2012-08-21	**Due date:**	
Subject:	TypoScript Array and Case should support "Registry" idea for defining the order		

Description

Currently, we have two TypoScript objects which work with some kind of "array": the "Case" TypoScript object and the "TypoScriptArray" TS object which renders a list of something.

For both of them, it would be good if we could get rid of the numeric arrays as much as possible, instead using the old "Registry" concept, so one could say: "Let's add that after XY" or "Before Z" or "at the end" or "at the beginning".

Old JS Registry class:

http://git.typo3.org/FLOW3/Packages/TYPO3.TYPO3.git?a=blob;f=Resources/Public/JavaScript/Core/Registry.js;h=7a9eb8b398c99872e639231befd15d1930fb245;hb=34b7bec14db9ddabf8a2e8a1e880895ab2607112

Old JS Test Class:

http://git.typo3.org/FLOW3/Packages/TYPO3.TYPO3.git?a=blob;f=Tests/JavaScript/Tests/Core/RegistryTest.js;h=749b2e8105ad31ad0b76b0e4336441435f0a3bc;hb=34b7bec14db9ddabf8a2e8a1e880895ab2607112

The exact concept still needs to be defined.

Associated revisions

Revision 9f554669 - 2012-08-21 00:14 - Sebastian Kurfuerst

[FEATURE] Explicit positions inside TypoScript Arrays

Before, positions inside a TypoScript array were specified solely using
the array keys. Example:

200 = Bar
100 = Foo
 -> Foo is rendered before Bar

With this change, it is possible to specify a meta-property "@position"
inside the TypoScript object; so the above example can be re-written as:

something = Bar
something.@position = 200
anotherthing = Foo
anotherthing.@position = 100

Note: if two elements have the same @position, they do NOT override each
    other. Instead, they are both rendered.

Position Syntax===============

Besides simple numeric positions, we also support the following position syntaxes:

- "start" - add at beginning; same as "start 0"
- "start 100" - add at beginning with priority 100, the higher the priority, the more at the beginning
- "end" - add at end; same as "end 0"
- "end 100" - add at end with priority 100, the higher the priority, the more at the end
- "100" - placed in the middle with position 100, the lower the position, the more at the beginning (this is the old behavior)
- "before someKey" - add before an element with key "someKey", if not matched, it will be added after the start
- "before someKey 100" - add before an element with key "someKey", the higher the priority, the more it goes before "someKey"
- "after someKey" - add after an element with key "someKey", if not matched, it will be added before the end
- "after someKey 100" - add after an element with key "someKey", the higher the priority, the more it goes after "someKey"

Backwards Compatibility==========

This change is fully backwards compatible: If no @position is given, the system falls back to numeric array indices.

Related: #39658
Change-Id: I3d2947e998dfc61e2503f433ef12588d5a15f0f3

Revision 305f3d84 - 2012-08-21 00:23 - Sebastian Kurfuerst

[!!!][FEATURE] Add @position support to Case TypoScript object and introduce explicit Matcher

With I3d2947e998dfc61e2503f433ef12588d5a15f0f3, we introduced an explicit @position for TypoScript Arrays. With this change, the same functionality is possible for the Case TypoScript object.

Additionally, matchers are now directly added as sub-objects of the Case TypoScript object, not using the "matchers" sub-key anymore.

Example before:

```
someCase = Case {
    matchers.10.condition = ${/* condition here /}
    matchers.10.type = 'MyType'
    matchers.20.condition = ${/ another condition */}
    matchers.20.type = 'MyOtherType'
}
```

Example now:

```
someCase = Case {
    myTypeCondition.condition = ${/* condition here /}
    myTypeCondition.type = 'MyType'
    myTypeCondition.@position = 10
```

```
        myOtherTypeCondition.condition = ${/ another condition */}
        myOtherTypeCondition.type = 'MyOtherType'
        myOtherTypeCondition.@position = 20
    }
```

Of course, the old numerically-indexed variant still works as fallback if @position is not specified.
You can also specify @position with "start", "end" and priorities, as in TypoScript arrays.

Furthermore, an extra TypoScript object for the Matcher has been implemented, which is used internally.
This enables functionality such as modifying the TypoScript context inside a certain matcher, or
using TypoScript prototypes for this:

```
    someCase = Case {
        prototype(TYPO3.TypoScript:Matcher) {
            @override.myOtherVariable = 'test2'
        }
    myTypeCondition {
            condition = ${/* condition here */}
            type = 'MyType'
            @override.myNewVariable = 'test'
        }
    }
```

In the example above, both myOtherVariable and myNewVariable are available when rendering MyType
through the condition.

Backwards Compatibility==========

We still support the old "matchers.*" syntax for a limited time; but you should take time and re-write
it to the new syntax.

Breaking Change==========

This change is still breaking because it needs a new TypoScript object "TYPO3.TypoScript:Matcher"
registered. Thus, your TypoScript needs at least to include the following line:

```
prototype(TYPO3.TypoScript:Matcher).implementationClassName = 'TYPO3\\TypoScript\\TypoScriptObjects\\Matcher'
```

For Phoenix, we can include the line in the default TypoScript, effectively making the change non-breaking for
our users.

Related: #39658
Change-Id: Ie70394fe7149e63deb661a22eac6445c0d66ac56

History

#1 - 2012-08-09 17:33 - Sebastian Kurfuerst

```
    @position = 'start'            # add somewhere at start of array
    @position = 'start 100'        # add at the start with priority 100 (the higher the prio, the more to the front it moves)
    @position = 'end'              # add at end of array
    @position = 'end 100'          # add at end of array (high prio -> end of array)
    @position = 'after [name]'     # add after named element "name". If it does not exist, TODO
    @position = 'before [name]'    # add before named element "name". If it does not exist, TODO
```

```
@position = 'after [name] 100' # the higher the priority, the closer it is after [name]
@position = 10                 # add at position "10" in middle. Note: any other element with position "10" will not be overridden
# if no position is given, element is placed somewhere in the middle
```

#2 - 2012-08-09 18:04 - Bastian Waidelich

I really like this feature and I prefer "start" and "end" to zero and a very high number!
But I'm not so happy about the proposed syntax as it's really hard to distinguish "keywords" and values.

So instead of using magic strings like

```
@position = 'start'
```

something like

```
@position = START
```

would be more readable IMO (we don't have TypoScript Constants, do we?)
The same for the "references"
Imagine:

```
foo = Array
foo.before = Text
foo.before.@position = 'start'
foo.someButton = Button
foo.someButton.@position = 'after before'
```

..ok that example is a bit far-fetched, but you get the point ;)

I don't know how we could improve that, but it would be cool if you could reference the position of the other TS object somehow

```
foo.someButton.@position = foo.before.@position + 1
```

(Just a shot in the dark, I'm not happy with that either)

Apart from that I don't think that we need the **priority modifier**. It only defers the issue: if two packages set the position and both use the same priority, we have the same situation as before. Besides the priority makes it more complex and harder to comprehend. I'd say: If two packages both add an element to the start, the order between those two doesn't matter. And if it does, one could overwrite the position of one element for the application.

#3 - 2012-08-16 11:12 - Sebastian Kurfuerst

- Debug Messages wenn Key bei "before" und "after" nicht existieren
- Format ist gut so wie es ist

- Wenn key bei "before" / "end" nicht da ist, schieben wir das element an "start" oder "end" (ohne Prio)

#4 - 2012-08-17 08:59 - Sebastian Kurfuerst

- Status changed from New to Under Review

- Assigned To set to Sebastian Kurfuerst

#5 - 2012-08-21 07:36 - Sebastian Kurfuerst

- Status changed from Under Review to Resolved

- % Done changed from 0 to 100