TYPO3.TypoScript - Feature # 38638

Feature # 38629 (Resolved): Make TypoScript usable outside the context of TYPO3 / Nodes, but also for building extensible FLOW3 applications

Status:	On Hold	**Priority:**	Should have
Author:	Sebastian Kurfuerst	**Category:**	
Created:	2012-07-04	**Assigned To:**	Sebastian Kurfuerst
Updated:	2012-11-28	**Due date:**	
Subject:	Introduce PresentationModel Wrapper in eel expressions in order to apply processors		

Description

Imagine you want to render two objects (like a blog and post) inside a FluidTemplate TypoScript object. Currently, you need to do the following:

```
foo = FluidTemplate
foo {
  blogTitle = ${context.property('blog.title')}
  blogDescription = ${context.property('blog.description')}
  ...
  postTitle = ${context.property('post.title')}
  ...
}
```

The above is quite cumbersome to write, and also does not feel nice because the Fluid templates differ then very much if using Fluid in combination with TypoScript and without.

More nice would be to use the following:

```
foo.blog = ${context.property('blog')}
```

However, often, we still want to run processors on properties of these objects, like specifying:

```
foo.blog.title << 1.wrap(before: '<h1>', after: '</h1>')
```

Now, in order to make this possible, I'd suggest to implement a new function presentationModel which takes an object like the post above and wraps it such that it can be used with processors and all the other TypoScript features. As an example:

```
foo.blog = ${presentationModel(context.property('blog'))}
foo.blog.title << 1.wrap(before: '<h1>', after: '</h1>')
# the following line would suppress the author
foo.blog.author >
# the following line would override the "description" with some fixed string (not in the model, but in the presentation layer)
foo.blog.description = "Hello"
```

It seems we cannot do the conversion into a presentationModel automatically as we would run into problems with object equality etc; and we cannot decide if the user needs the object itself in the view, or just wants to output its values. Basically we don't even know what a TypoScript object does with such an object -- so we cannot wrap anything automatically.

Note: This also plays nicely together e.g. with #38631.

What do you think about this feature?

Related issues:

blocks Foo.ContentManagement - Major Feature # 37293: Refactor Admin to work ...	**New**	**2012-05-18**

History

#1 - 2012-07-05 22:11 - Gerrit Code Review

Patch set 1 for branch **master** has been pushed to the review server.
It is available at http://review.typo3.org/12654

#2 - 2012-07-05 22:13 - Sebastian Kurfuerst

- Status changed from New to Accepted

- Assigned To set to Sebastian Kurfuerst

#3 - 2012-07-09 15:11 - Christopher Hlubek

I think this makes sense but I'm unsure about the naming. presentationModel is too awkward to type and not concise for me.

Considering #38631 the context itself will be the plain object anyway, correct? So the example would become:

```
foo = FluidRenderer
foo {
  blog = ${presentationModel(context.getProperty('blog'))}
  blog.title << 1.wrap(before: '<h1>', after: '</h1>')
}
```

I don't think TypoScript authors should care about ${context.getProperty('blog')} vs. ${q(context).property('blog')} and also explicit decoration with presentationModel will be hard to explain.

What about implementing this transparently and "unpacking" the FlowQuery object in the FluidRenderer TS object and also decorating it by default with something like the PresentationModel? A FlowQuery object does not make much sense in a Fluid template since it cannot call the methods on it. Then the example would become (without #38631):

```
foo = FluidRenderer
foo {
  blog = ${context.property('blog'))}
  blog.title << 1.wrap(before: '<h1>', after: '</h1>')
}
```

If you want to access the raw element you could write ${context.get()} anyway.

For this to work we have to extend Eel a little bit to enable the usage of a custom context class by the means of a factory. Right now we use the TYPO3\Eel\Context class to wrap every result of an eel expression. We could think about using the PresentationModel class in combination with the functionality from Context to automatically evaluate TypoScript processors. But I'm not sure how to do this with nested objects.

So for me the question really is how to support something like that:

```
foo = FluidRenderer
foo {
  blog = ${context.property('blog'))}
  blog.author.name << 1.wrap(before: '<span class="author">', after: '</span>')
}
```

#4 - 2012-11-28 15:33 - Karsten Dambekalns

- Status changed from Accepted to On Hold

Abandoning the change, feel free to reopen.