fluid.render

This functionality is Sneak Peek status. This means that the **APIs may change**. We welcome your feedback, ideas, and code, but please use caution if you use this new functionality.

This page is still incomplete.

fluid.render()

A simple driver for single node self-templating. Parses the source into a template structure, renders it using the supplied component tree and options, replaces the markup in the target node with the rendered markup, and finally performs any required data binding.

```
fluid.render(source, target, tree, options);
```

File name: `Renderer.js`

Parameters

`source`	(Object or String) Either a structure {`node: node, armouring: armourstyle`} or a string holding a literal template. See the source section below for more information.
`target`	(Node) The node to receive the rendered markup
`tree`	(Object) The component tree to be rendered
`options`	(Object) An options structure to configure the rendering and binding process. See the options section below for more information.

Return Value

Object	A templates structure, suitable for a further call to `fluid.reRender` or `fluid.renderTemplates`

See Also

- fluid.initRendererComponent

Notes

The `source` parameter

The `source` parameter can either be a string holding a literal template or an object with the following structure:

```
{
    node: <DOM node>,
    armouring: <boolean>
}
```

The properties in the object are:

`node`	This is a either a pure DOM node or a jQuery object wrapping a DOM node.
`armouring`	This boolean flag controls whether or not the `node` is treated as ...? (default: `false`)

Options

The `options` parameter is an optional object containing key/value pairs that can configure how the Renderer works. Supported options are:

Field	Description	Type	In/Out
`model`	Perhaps the most important parameter, contains the "data model" to which value bindings expressed within the tree will be expressed.	free `Object`	Mostly In, but the supplied model may be written to later as a result of servicing user actions, especially if the parameter `autoBind` is supplied.
`applier`	a ChangeApplier object associated with the `model`	`ChangeApplier`	In
`autoBind`	If set, any user modification of fields with `valuebindings` s set will immediately be reflected in the current state of the supplied `model`	`boolean`	In
`document`	If set, will cause the rendered ids to be uniquified against the supplied document, rather than the current one	`document`	In
`debugMode`	If set, mismatches between template and component tree will be highlighted in an unreasonable garish pink colour	`boolean`	In
`idMap`	This map operates in conjunction with the `identify` decorator which may be attached to nodes in the component tree. Whilst rendering takes place, this map will fill up with a lookup from the supplied nickname to the finally rendered id	free `Object`	In/Out
`messageLocator`	Configures the lookup from (I18N) messages referenced in the component tree, to some source of a message bundle	`function (key, args)->message`	In
`messageSource`	Will construct a `messageLocator` from a raw bundle specification via a call to `fluid.resolveMessageSource`	`MessageSource` structure	In
`renderRaw`	Will XMLEncode the rendered markup before insertion into the document. Can be useful for debugging	`boolean`	In
`cutpoints`	This is properly a directive to the parser, rather than the renderer, but the options structure is shared. This contains a list of pairs of `id, selector` which will be used to impute an `rsf:id` structure onto a document, by means of matching the paired selector	Array of `Cutpoint` object	In

Example

```
example here
```

In this example, description here...