Renderer-bearing Components

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.

Overview

If you are creating a component that requires the use of the Renderer, use the new `fluid.initRendererComponent` function in your component creator function:

```
my.component = function (container, options) {
    var that = fluid.initRendererComponent("my.component", container, options);
    ...
    return that;
}
```

This new function automates the work of applying the Renderer, fetching templates, configuring cutpoints based on the DOM binder, as well as localisation via the string bundle.

This function will:

- create `that.model`, using `options.model` if available (creating an empty object if not)
- fetch any resources (such as HTML templates, etc.) specified in `options.resources`
- create a renderer function and attach it to your `that` object as `that.render(tree);`

Options for Renderer-bearing Components

While developers are free to define whatever options they like for their component, a component initialised with fluid.initRendererComponent will also understand certain options specific to the Renderer:

Name	Description	Values	Default
`model`	The "data model" to which value bindings expressed within the tree will be expressed	Object	none
`resources`	A list of resources (such as HTML files, CSS files, data files) that are required by the component.	Object as required by `fluid.fetchResources`	none
`resolverGetConfig`	Configuration functions to be applied to any data retrieved from the model	Array of functions	The raw value will be retrieved unchanged.
`resolverSetConfig`	Configuration functions to be applied to any data being saved in the model	Array of functions	The raw value will be saved unchanged.
`rendererOptions`	Options that will be included in the `rendererFnOptions` as `rendererOptions`	Object	
`rendererFnOptions`	Options that will be passed directly to the renderer creation function, `fluid.renderer.createRendererFunction`	Object	See the documentation for `fluid.renderer.createRendererFunction`
`selectors`	A set of named selectors that will be converted to cutpoints for use by the renderer	Object	none
`repeatingSelectors`	A list of any of the named `selectors` that reference elements that will be repeated when renderer (e.g. rows in a table)	Array of Strings	none
`selectorsToIgnore`	A list of any of the named `selectors` that should *not* be included in the renderer cutpoints	Array of Strings	none
`protoTree`	A data structure that represents the binding between the contents and data. Also see Renderer Component Trees for more detail.	Object	none
`produceTree`	A user-defined function that returns protoTree	a function	none
`renderOnInit`	A flag indicating whether or not the component should render itself automatically after initialization.	boolean	false

Events for Renderer-bearing Components

prepareModelForRender

Description	This event fires before the generation of protoTree. Whatever adjustment on the model, which is the protoTree is generated based on, is ideal to be performed at this event.

Parameters	*model* The internal Model Component that is used by this renderer component. *applier* The internal Change Applier Component that is used by this renderer component. *that* The reference to the current renderer component.
Returns	None
Note	The first event to be fired before events "onRenderTree" and "afterRender".
Availability	Infusion 1.4 and later

onRenderTree

Description	This event fires right before protoTree is rendered. This event is ideal for the final manipulation of the fully expanded protoTree.
Parameters	*that* The reference to the current renderer component. *tree* Expanded renderer tree.
Returns	None
Note	The event fired after "prepareModelForRender" and before "afterRender".
Availability	Infusion 1.4 and later

afterRender

Description	This event fires after protoTree is rendered.
Parameters	*that* The reference to the current renderer component.
Returns	None
Note	The event fired after "onRenderTree" and "afterRender".
Availability	Infusion 1.4 and later

Note: The 3 events are fired in the order of prepareModelForRender, onRenderTree, afterRender. They are only intended for use by experts.

Functions on `that`

render(tree)

```
that.render(tree);
```

Expands the provided tree, generates cutpoints, and renders the tree.

produceTree()

```
that.produceTree();
```

This function is only present if a protoTree has been provided in the options. This function can be overridden by providing a produceTree in the options.

refreshView()

```
that.refreshView();
```

This function simply calls `that.render(that.produceTree());` This function is only present if a protoTree has been provided in the options.

Example to render a drop down list box

```
(function ($, fluid) {
    fluid.examples.rederer = function (container, options) {
        var that = fluid.initRendererComponent("fluid.examples.rederer", container, options);
        that.renderer.refreshView();

        return that;
    };

    fluid.defaults("fluid.examples.rederer", {
        gradeNames: ["fluid.rendererComponent"],
        selectors: {
            textFont: ".flc-examples-text-font",
            notInProtoTree: ".flc-examples-not-in-protoTree"
        },
        // "selectorsToIgnore" is an array of all the selectors
        // that are defined in "selectors" but not used in
        // "protoTree". It tells renderer not to generate cutpoints
        // for these selectors.
        selectorsToIgnore: ["notInProtoTree"],
        model: {
            textFontNames: ["Serif", "Sans-Serif", "Arial"],
            textFontList: ["serif", "sansSerif", "arial"],
            textFontValue: ""
        },
        rendererOptions: {
            autoBind: true,
        },
        protoTree: {
            // "textFont" is an ID that is defined in "selectors"
            // option
            textFont: {
                // "textFontNames", "textFontList", "textFontValue"
                // must be defined in "model"
                optionnames: "${textFontNames}",
                optionlist: "${textFontList}",
                selection: "${textFontValue}"
            }
        }
        resources: {
            template: {
                forceCache: true,
                url: "examples-rederer.html"
            }
        }
    });
})
```

The template "examples-rederer.html" looks like,

```
<form id="options" action="">
    <label for="text-font" class="fl-label">Font style:</label>
    <select class="flc-examples-text-font" id="text-font">
    </select>
</form>
```

This example uses renderer component to generate a drop down list box. `fluid.initRendererComponent()`. The option "protoTree" is the key option that establishes the binding between the "selectors" and data presented in "model". See `Component Tree Expanders` for more usage of "protoTree".