Table of Contents

This documentation applies to v1.4 of Table of Contents.
For earlier versions, see
Table of Contents API.

Production Status: SNEAK PEEK

The Table of Contents component examines a page for HTML heading elements, and generates and formats a list of links to headings that can be used to navigate the page. It is used by the UI Enhancer component.

See Also
UI Enhancer

New in 1.4 (August, 2011)

- Code refactored

#Creator	`#fluid.tableOfContents(container, options)`

#Supported Events	`#onReady`
	Fires when the component has finished initializing.
	
	`#afterRender`
	Fires after the table of contents has been rendered.

#Methods	`#filterHeadings(headings)`
	Used to filter the headings that will appear in the table of contents.
	
	`#generateGUID()`
	Used in the process of creating anchors for the headings.
	
	`#hide()`
	Hides the table of contents.
	
	`#insertAnchor(name, element)`
	Used to insert anchors before each heading.
	
	`#show()`
	Shows the table of contents.

#Options	**`listeners`**
	See #Supported Events for information.
	
	`selectors`
	See below.

#Selectors	`#headings`
	Identifies the headings in the page
	
	`#tocContainer`
	Container for the generated table of contents

#Subcomponents	`#levels`
	Renders the generated table of contents.
	
	`#modelBuilder`
	Builds an internal model of the structure of the headings in the document.

## Creator

back to top

Use the following function to create a Table of Contents component:

fluid.tableOfContents(container, options)

Method	`fluid.tableOfContents(container, options);`
Description	Instantiates a Table of Contents component applied to the specified container.

Parameters	*`container`*
	A CSS-based selectors, single-element jQuery object, or DOM element that identifies the root DOM node where the Table of Contents component should search for headings.
	
	`options`
	An optional data structure that configures the Table of Contents component, as described below.

Returns	The Table of Contents component
Examples	`var toc = fluid.tableOfContents("#main");`
Notes	The Table of Contents only processes headings found inside the element provided by the `container` argument. This allows the component to be applied to only a section of a page, if desired. Keep in mind that the generated list of links will be placed inside the `container`, as specified by the `tocC ontainer` selector.

Supported Events

back to top

Listeners can be attached to any supported events through a component's `listeners` option. Values can be a function reference (not a string function name) or an anonymous function definition, as illustrated below:

```
var myComponent = component.name("#myContainerID", {
    listeners: {
        eventName1: functionName,
        eventName2: function (params) {
            ...
        }
    }
});
```

For information on the different types of events, see Infusion Event System.

onReady

Description	This event fires when the DOM is fully loaded. This event triggers the creation of the Table of Contents subcomponent.
Type	default
Parameters	*none*
Availability	Infusion 1.4 and later

afterRender

Description	This event fires Fires when any settings have changed.
Type	default
Parameters	*newModel* The overall model containing the user's current selections. *oldModel* A "snapshot" of the previous state of the model, before the change. *changeRequest* The original change request object.
Availability	Infusion 1.4 and later
See also	Firing a change using a ChangeApplier

Methods

back to top

This component has public methods that can be invoked by integrators as necessary. Some of these methods may be implemented as Invokers, which resolve their arguments from the environment at invocation time. These methods can be configured by the integrator if necessary: arguments can be changed, and in fact the entire implementation function can be replaced by a custom function (though it is likely rare that this would be necessary or desirable).

Configuration of invokers is carried out using Demands Specifications, using the following pattern:

```
fluid.demands("component.name", [<context>], {
    invokers: {
        <invokerName>: {
            funcName: <implementation function name>,
            args: [<array of argument specifications>]
        }
    }
});
```

filterHeadings(headings)

Description	Used by the component to filter a list of headings. The default implementation filters the list based on visibility.
Parameters	*headings* A list of headings, usually generated by `that.locate("headings")`
Configurable	*yes* Default implementation: `"fluid.tableOfContents.filterHeadings"`
Availability	Infusion v1.4

generateGUID()

Description	Used by the component in the creation of unique anchors for the headings. The default implementation uses fluid.allocateSimpleId.
Parameters	*none*
Configurable	*yes* Default implementation: `"fluid.tableOfContents.generateGUID"`
Availability	Infusion v1.4
See also	#insertAnchor

hide()

Description	Hides the generated table of contents.
Parameters	*none*
Configurable	*no*
Availability	Infusion v1.0
See also	#show()

insertAnchor(name, element)

Description	Used by the component to create and insert anchors before each element linked to by the table of contents. The default implementation creates an HTML <a> element using the `name` parameter as both the name and id of the element.
Parameters	*name* The name used in the creation of the anchor. *element* The element to insert the anchor before.
Configurable	*yes* Default implementation: `"fluid.tableOfContents.insertAnchor"`
Availability	Infusion v1.4
See also	#generateGUID

show()

Description	Shows the generated table of contents.
Parameters	*none*

Configurable	*no*
Availability	Infusion v1.0
See also	#hide()

Options

back to top

The second argument to the creator function is the options argument. This is a JavaScript object containing name/value pairs: The name is the name of the option and the value is the desired setting. Components define their own default values for options, but integrators can override these defaults by providing new values using the options argument. For technical information about how options are merged with defaults, see Options Merging.

`selectors`

Description	See below for details.

Selectors

back to top

One of the options that can be provided to the component is a set of CSS-based selectors identifying where in the DOM different elements can be found. The value for the option is itself a Javascript object containing name/value pairs:

```
selectors: {
    selector1Name: "selector 1 string",
    selector2Name: "selector 2 string",
      ...
}
```

The component defines defaults for these selectors: If you use those defaults in your markup, you do not need to specify the selectors option. If you do choose to override any of the selectors, you can specify your custom selector using this option.

The selectors supported by Table of Contents are described below.

`headings`

Description	This selector will be used to identify all of the headings in the page. These headings will be used in the table of contents (after filtering).
Default	`":header"`

Example	`fluid.tableOfContents(".myContainer", {`
	`    selectors: {`
	`        headings: "h1,h2,h3" // limit to only three levels deep`
	`    }`
	`});`

See also	`#filterHeadings`

`tocContainer`

Description	This selector identifies the container into which the generated table of contents will be inserted. This will typically be an empty `<div>` element.
Default	`".flc-toc-tocContainer"`

Example	`fluid.tableOfContents(".myContainer", { selectors: { tocContainer: "#tableOfContentsDiv" } });`

```
fluid.tableOfContents(".myContainer", {
    selectors: {
        tocContainer: "#tableOfContentsDiv"
    }
});
```

Subcomponents

Subcomponents can be configured though the parent component's `components` option using the following pattern:

```
parent.component.name(".myContainer", {
    components: {
        <subcomponentName>: {
            options: {
                <subcomponent options>
            }
        }
    }
});
```

levels

Description	The `levels` subcomponent renders the generated table of contents based on a template.
Default implementation	`fluid.tableOfContents.levels`
See also	Table of Contents Levels

modelBuilder

Description	The `modelBuilder` subcomponent builds an internal model of the structure of the headings in the document. This model is used by the Table of Contents component to...
Default implementation	`fluid.tableOfContents.modelBuilder`
See also	Table of Contents Model Builder