Tutorial - UI Options and UI Enhancer

This tutorial applies to v1.3 and earlier. For tutorials explaining how to use UIOptions v1.4 and later, please see the How-to Guides in the new documentation space.

This page will walk you through an example of adding the Fluid UI Options and UI Enhancer to a website.

This tutorial assumes that:

- you are already familiar with HTML, Javascript and CSS
- you are familiar with what the UI Options and UI Enhancer are and do
- now you just want to know how to add them to your application.

UI Options requires UI Enhancer in order to apply the settings a user selects. UI Enhancer could be used alone in order to apply some effects to a page however its real power is when used in conjunction with UI Options. For more general information about UI Options see User Interface Options Version A Design Overview. For technical API documentation, see UI Options API and UI Enhancer API.

Tutorial: How to Use UI Options and UI Enhancer

Scenario

You'd like to allow visitors to customize the presentation of your website to their individual needs, such as enlarging the text, or increasing the visual contrast. This tutorial will show you how to use the Fluid UI Options in a dialog to do this.

The basic steps to add UI Options and UI Enhancer to your application are:

- fluid:Setup
- Step 1: Prepare your markup
- Step 2: Add dependencies to the markup
- Step 3: Write the script to create a UI Enhancer
- Step 4: Write the script to create a UI Options
- Step 5: Create the preview
- Step 6 (Optional): Use an accordian to simplify the interface
- Step 7 (Optional): Turn UI Options into a dialog
- Step 8 (Optional): Add UI Enhancer to other pages on your site

The rest of this tutorial will explain each of these steps in detail.

Status
This component is in Preview status

On This Page

- Scenario
- Setup: Download and install the Fluid Infusion library
- Step 1: Prepare your markup
- Step 2: Add dependencies to the markup
- Step 3: Write the script to create a UI Enhancer
- Step 4: Write the script to create a UI Options
- Step 5: Create the preview
  - Using the default preview
  - Guidelines for building a custom preview
- Step 6 (Optional): Use an accordian to simplify the interface
- Step 7 (Optional): Turn UI Options into a dialog
- Step 8 (Optional): Add UI Enhancer to other pages on your site

See Also

- (Floe) UI Options (2008-2009)
- UI Options API
- UI Enhancer API

Still need help?

Join the infusion-users mailing list and ask your questions there.

Setup: Download and install the Fluid Infusion library

1. Download a copy of the Fluid Infusion component library from:
    - http://fluidproject.org/products/fluid-infusion/download-infusion/
      You only really need the "Minified deployment package," but if you want to actually look at the code, you should download the "Source package."
2. Unpack the zip file you just downloaded, and place the resulting folder somewhere convenient for your development purposes.
   The folder will have the release number in its name (e.g. infusion-1.4/). The rest of this tutorial will use infusion-1.4 in its examples, but if you downloaded a different version, you'll have to adjust.

Step 1: Prepare your markup

Let's assume that you're starting with an HTML file, called `home.html` with the following content:

```
<body>
    <h1>Some Sample Title</h1>
    <p>Some content with <a id="myLink">a link</a> and<button id="myButton">a button</button>. </p>

    <!-- div that will contain the UI Options component -->
    <div id="myUIOptions">
    </div>
</body>
```

The UI Options component needs to know about the 'container' for the preference setting user interface. We will use an ID based selector of "myUIOptions" when creating the UI Options component below.

Step 2: Add dependencies to the markup

UI Options and UI Enhancer depend upon Fluid Skinning System (FSS) so the CSS files will need to be included. You'll also need to add the Fluid Infusion JavaScript to your HTML file. In the header of the file, link to the CSS files with `<link>` tags and the JavaScript files with `<script>` tags. Let's also link to a JavaScript file that we will create in fluid:step 3 (note the CSS files go first):

```
<link rel="stylesheet" type="text/css" href="framework/fss/css/fss-layout.css" />
<link rel="stylesheet" type="text/css" href="framework/fss/css/fss-text.css" />
<link rel="stylesheet" type="text/css" href="framework/fss/css/fss-theme-hc.css" />
<link rel="stylesheet" type="text/css" href="framework/fss/css/fss-theme-hci.css" />
<link rel="stylesheet" type="text/css" href="framework/fss/css/fss-theme-mist.css" />
<link rel="stylesheet" type="text/css" href="framework/fss/css/fss-theme-rust.css" />
<link rel="stylesheet" type="text/css" href="framework/fss/css/fss-theme-coal.css" />
<link rel="stylesheet" type="text/css" href="framework/fss/css/fss-theme-slate.css" />
<link rel="stylesheet" type="text/css" href="components/uiOptions/css/UIOptions.css" />
<link rel="stylesheet" type="text/css" href="components/uiOptions/css/Slider.css" />

<script type="text/javascript" src="InfusionAll.js"></script>
<script type="text/javascript" src="myInit.js"></script>
```

NOTE that the `InfusionAll.js` file is minified - all of the whitespace has been removed, so it isn't really human-readable. If you're using the source distribution and you want to be able to debug the code, you'll want to include each of the required files individually. Instead of including InfusionAll.js you would include this:

```
<!-- jQuery files -->
<script type="text/javascript" src="lib/jquery/core/js/jquery.js"></script>
<script type="text/javascript" src="lib/jquery/ui/js/jquery.ui.core.js"></script>
<script type="text/javascript" src="lib/jquery/ui/js/jquery.ui.widget.js"></script>
<script type="text/javascript" src="lib/jquery/ui/js/jquery.ui.mouse.js"></script>
<script type="text/javascript" src="lib/jquery/ui/js/jquery.ui.accordion.js"></script>
<script type="text/javascript" src="lib/jquery/ui/js/jquery.ui.slider.js"></script>

<script type="text/javascript" src="lib/json/js/json2.js"></script>

<!-- Infusion files -->
<script type="text/javascript" src="framework/core/js/jquery.keyboard-a11y.js"></script>
<script type="text/javascript" src="framework/core/js/Fluid.js"></script>
<script type="text/javascript" src="framework/core/js/FluidRequests.js"></script>  <!-- New in v1.3 -->
<script type="text/javascript" src="framework/core/js/FluidDOMUtilities.js"></script>
<script type="text/javascript" src="framework/core/js/DataBinding.js"></script>
<script type="text/javascript" src="framework/core/js/FluidIoC.js"></script>  <!-- New in v1.4 -->
<script type="text/javascript" src="lib/fastXmlPull/js/fastXmlPull.js"></script>
<script type="text/javascript" src="framework/renderer/js/fluidParser.js"></script>
<script type="text/javascript" src="framework/renderer/js/fluidRenderer.js"></script>
<script type="text/javascript" src="framework/renderer/js/RendererUtilities.js"></script>
<script type="text/javascript" src="components/uiOptions/js/Store.js"></script>  <!-- New in v1.4 -->
<script type="text/javascript" src="components/uiOptions/js/UIEnhancer.js"></script>
<script type="text/javascript" src="components/uiOptions/js/UIOptions.js"></script>
<script type="text/javascript" src="components/tableOfContents/js/TableOfContents.js"></script>

<!-- for Full-Page UI Options only: -->
<script type="text/javascript" src="components/uiOptions/js/FullNoPreviewUIOptions.js"></script>  <!-- New in
v1.4 -->

<!-- for Full-Page With Preview UI Options only: -->
<script type="text/javascript" src="components/uiOptions/js/FullPreviewUIOptions.js"></script>  <!-- New in v1.
4 -->

<!-- for Fat-Panel UI Options only: -->
<script type="text/javascript" src="components/uiOptions/js/URLUtilities.js"></script>  <!-- New in v1.4 -->
<script type="text/javascript" src="components/uiOptions/js/FatPanelUIOptions.js"></script>  <!-- New in v1.4 --
>
<script type="text/javascript" src="components/uiOptions/js/SlidingPanel.js"></script>  <!-- New in v1.4 -->
```

But all of these individual files are not necessary to make it work - the `InfusionAll.js` file has everything you need.

If you now load your `home.html` file in a browser you will see the sample content.

Step 3: Write the script to create a UI Enhancer

You'll need to create a file, say `myInit.js`, to contain your initialization script.

UI Options depends upon UI Enhancer and requires that a UI Enhancer is already on the page. So the first thing we will do is write a function to initialize the UI Enhancer.

When initializing the UI Enhancer it is important that we set the defaults that we want our site to display with, otherwise the default appearance of our site may be altered by the UI Enhancer. The defaults are set using the FSS and numeric and boolean values. See the UI Enhancer API for details about the defaultSiteSettings options. In this example, our site is based on the FSS theme 'mist', and uses bold, underlined links as the defaults.

The UI Enhancer has a Table of Contents subcomponent that has its own options. One of its options is an URL from which the template for a table of contents will be loaded. You will likely need to override the default URL as is done below.

```
(function ($, fluid) {

    //initialize the UI Enhancer
    var initUIEnhancer = function () {

        var enhancerOpts = {
            defaultSiteSettings: {
                theme: "mist",
                linksBold: true,
                linksUnderline: true
            },
            tableOfContents: {
                options: {
                    templateUrl: "components/tableOfContents/html/TableOfContents.html"
                }

            }
        };

        return fluid.uiEnhancer(document, enhancerOpts);
    };
})(jQuery, fluid);
```

We still need to call our initialization function but we must ensure the document is ready. This is accomplished in the document ready block below.

```
(function ($, fluid) {

    //initialize the UI Enhancer
    var initUIEnhancer = function () {

        var enhancerOpts = {
            defaultSiteSettings: {
                theme: "mist",
                linksBold: true,
                linksUnderline: true
            },
            tableOfContents: {
                options: {
                    templateUrl: "components/tableOfContents/html/TableOfContents.html"
                }

            }
        };

        return fluid.uiEnhancer(document, enhancerOpts);
    };

    //as soon as web document is loaded, initialize the UI Enhancer
    $(document).ready( function () {
        initUIEnhancer();
    });
})(jQuery, fluid);
```

If you reload the `home.html` page, you will see that the default styles have been applied.

Step 4: Write the script to create a UI Options

Now create an initialization function for the UI Options component. The first parameter to the `fluid.uiOptions` is the container DOM element `uiOption`, which can be found on the UI Options template. The second parameter is our default options for the UIOptions component - we need to specify the URL to content for the preview pane.

```
//initialize UI Options component
var initUIOptions = function () {
    return fluid.uiOptions(".uiOptions");
};
```

Currently you need to load the UI Options template yourself although this will be fixed in FLUID-2616. Use jQuery to load the section of the template that is required - the form with the `uiOptions` class on it. The template is loaded into the DOM node identified by #myUIOptions in `home.html`. Use jQuery to find this node.

```
//load the UI Options component
var loadUIOptions = function () {
    var urlSelector = "components/uiOptions/html/UIOptions.html .uiOptions";
    uiOptionsNode.load(urlSelector, initUIOptions);
};
```

Don't forget to define `uiOptionsNode` and call `loadUIOptions` in your document ready block. You should also declare `uiOptionsNode` at the beginning so its available to other functions. When you are finished your code will look something like this:

```
(function ($, fluid) {

    var uiOptionsNode;

    //initialize the UI Enhancer
    var initUIEnhancer = function () {

        var enhancerOpts = {
            defaultSiteSettings: {
                theme: "mist",
                linksBold: true,
                linksUnderline: true
            },
            tableOfContents: {
                options: {
                    templateUrl: "components/tableOfContents/html/TableOfContents.html"
                }

            }
        };

        return fluid.uiEnhancer(document, enhancerOpts);
    };

    //initialize UI Options component
    var initUIOptions = function () {
        return fluid.uiOptions(".uiOptions");
    };

    //load the UI Options component
    var loadUIOptions = function () {
        var urlSelector = "components/uiOptions/html/UIOptions.html .uiOptions";
        uiOptionsNode.load(urlSelector, initUIOptions);
    };

    //as soon as web document is loaded, initialize the UI Enhancer
    $(document).ready(function () {
        //use JQuery to find the dom node represented by myUIOptionsSelector
        uiOptionsNode = $("#myUIOptions"); //selector in which to load ui options
        initUIEnhancer();
        loadUIOptions();
    });

})(jQuery, fluid);
```

Reloading the `home.html` file will show the UI Options component on the page. Note that the preview window has not loaded.

Step 5: Create the preview

Using the default preview

The preview for UI Options is displayed within an IFrame. There is a default preview file that ships with Fluid Infusion - if you want to use this preview, simply copy `components/uiOptions/UIOptionsPreview.html` into the same directory as `home.html` and it will be loaded into the IFrame. Note that for the preview to work it will need access to the CSS files that are linked to in the head of the `UIOptionsPreview.html` file. Ensure that these have the correct paths.

Another option, instead of copying the default file to another directory, is to specify any file as an option called "previewTemplateUrl". Options are sent as the second parameter when calling the UIOptions component, so setting the preview template url would work like this:

```
 //initialize UI Options component
var initUIOptions = function () {
    var options = {
        previewTemplateUrl: "components/uiOptions/html/UIOptionsPreview.html"
    };
    return fluid.uiOptions(".uiOptions", options);
};
```

Guidelines for building a custom preview

It is more likely that you will want the preview to contain something that better reflects your site. For a preview to be an accurate portrayal of how UI Options can alter your site, it's important to use a variety of different markup tags. Don't just take a paragraph of plain text from one page as your preview - rather, try to include headings, links, form controls, graphics, lists, and tables for a more complete site preview.

Create a file with sample content called "mypreview.html" in the same directory as home.html. Alter your UI Options initialization function as follows to implement your preview:

```
 //initialize UI Options component
var initUIOptions = function () {
    var options = {
        previewTemplateUrl: "mypreview.html"
    };
    return fluid.uiOptions(".uiOptions", options);
};
```

If you reload your `home.html` page now, you will see the preview and be able to use the controls in the UI Options component. The UI Enhancer will remember your saved settings and apply them the next time you load this page. By default, the UI Enhancer uses a cookie to store your settings.

Step 6 (Optional): Use an accordian to simplify the interface

The UI Options interface can get be quite busy given all the setting controls. To simplify the interface you may want to use an accordian to hide and show parts of the UI Options form. We will use jQuery accordian plugin. If you are not using the `InfusionAll.js` file, you will need to load the `ui.accordian.js` file in the head of the `home.html` file.

```
<script type="text/javascript" src="lib/jquery/ui/js/ui.accordion.js"></script>
```

Now we will modify the `initUIOptions` function we created in fluid:step 4. We will add an `afterRender` listener to the UI Options component and include there the declarations needed to decorate the component with an accordian. Don't forget to pass the newly created `options` variable to `uiOption`
.

```
//initialize UI Options component
var initUIOptions = function () {
    var options = {
        listeners: {
            afterRender: function () {
                $(".uiOptions .fl-col:eq(0)").accordion({
                    header: 'h2',
                    clearStyle: true,
                    autoHeight: false
                });
                $(".uiOptions .fl-col h2:eq(0)").focus();
            }
        }
    };
    return fluid.uiOptions(".uiOptions", options);
};
```

If you reload the `home.html` file you will see the accordian feature when you click on "Easier to see" or "Easier to find".

Step 7 (Optional): Turn UI Options into a dialog

You may want the UI Options component to be displayed in a dialog. Here you can use the jQuery dialog plugin. If you are not using the `InfusionAll.js` file, you will need to load the `ui.dialog.js` file in the head of the `home.html` file.

```
<script type="text/javascript" src="lib/jquery/ui/js/ui.dialog.js"></script>
```

In addition, you will want to modify the html slightly by adding another and a <button> as shown below. We will use an ID based selector of "dialogContent" for this purpose.

```
<body>
    <h1>Some Sample Title</h1>
    <p>Some content with <a id="myLink">a link</a> and<button id="myButtonTest">a button</button>. </p>

    <!-- div that will contain the UI Options component -->
    <div id="myUIOptions">
        <div id="dialogContent">
        </div>
    </div>
    <button id="myButton">Edit Appearance</button>
</body>
```

In the `myInit.js` file add a `setupDialog` function that creates a jQuery dialog and sets it to open when the `myButton` button is clicked. In your HTML, take care to place your `myButton` button in a place that is easily discoverable to your users.
You may want to set some options in the dialog that will make it look a little nicer than the default. Also modify `initUIOptions` adding listeners to the `onSave` and `onCancel` functions that will close the dialog.

```
var setupDialog = function() {

    // Create the dialog
    uiOptionsNode.dialog({
        bgiframe: true,
        width: '60em',
        modal: true,
        autoOpen: false,
        draggable: true
    });

    // Bind event handlers for it.
    $("#myButton").click(function() {
        uiOptionsNode.dialog("open");
    });
};

var initUIOptions = function() {
    var options = {
        listeners: {
            afterRender: function() {
                $(".uiOptions .fl-col:eq(0)").accordion({
                    header: 'h2',
                    clearStyle: true,
                    autoHeight: false
                });
                $(".uiOptions .fl-col h2:eq(0)").focus();
            },
            onCancel: function() {
                uiOptionsNode.dialog("close");
            },
            onSave: function() {
                uiOptionsNode.dialog("close");
            }
        }
    };

    return fluid.uiOptions(".uiOptions", options);
};

$(document).ready(function(){
    initUIEnhancer();
    loadUIOptions();
    setupDialog();
});
```

Reloading the `home.html` page now will result in the UI Options component disappearing from the page. Clicking the button will pop up the dialog, however, the dialog will be transparent. Applying the classes `fl-widget` and `fl-grabbable` on the dialog should take care of this issue:

```
    // Create the dialog
    uiOptionsNode.dialog({
        bgiframe: true,
        width: '60em',
        modal: true,
        dialogClass: 'fl-widget fl-grabbable',
        autoOpen: false,
        draggable: true
    });
```

Another issue that you may notice is that if you make a change and then close the dialog the change is remembered the next time the dialog is opened, which is not the way a dialog is supposed to behave. Because this is a modal dialog, one way to address this issue is to de-activate the close functionality (use css to hide the close link and set closeOnEscape to false).

```
// Create the dialog
uiOptionsNode.dialog({
    bgiframe: true,
    width: '60em',
    modal: true,
    closeOnEscape: false,
    autoOpen: false,
    draggable: true
});
```

Step 8 (Optional): Add UI Enhancer to other pages on your site

Your site may not have the UI Options interface available on every page, however, you do want your users settings to be applied to your entire site. In order to do this you will need to add a UI Enhancer to every page by following the instructions in fluid:step 3.