Tutorial - Layout Reorderer

This tutorial has no yet been updated to reflect the latest APIs.

This page will walk you through an example of adding the Fluid Layout Reorderer component to your application.

This tutorial assumes that:

- you are already familiar with HTML, Javascript and CSS
- you are familiar with what the Layout Reorderer is and does
- now you just want to know how to add it to your file.

For technical API documentation, see Layout Reorderer API.

Tutorial: How to Use the Layout Reorderer

Scenario

You're creating a portal system that will aggregate news content from a number of different tech gadgetsources. You'd like to allow site visitors to rearrange the portlets. This tutorial will show you how to use the Fluid Layout Reorderer to offer this functionality.

There are four basic steps to adding the Layout Reorderer to your application:

- Setup: Download and install the Fluid Infusion library
- Step 1: Prepare your markup
- Step 2: Write the script
- Step 3: Add the script to your HTML
- Step 4: Styling

Status
This component is in Production status

On This Page

See Also

- Layout Reorderer API
- Tutorial - Layout Reorderer Selectors

Still need help?

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

Setup: Download and install the Infusion library

1. Get the current source code from github as a ZIP file: https://github.com/fluid-project/infusion/archive/master.zip
2. Unpack the zip file you just downloaded and cd into the "infusion-master" folder that results.

3. If necessary install Node.js (http://nodejs.org/download/) and Grunt (“npm install -g grunt-cli”).
4. Make your own custom build by running the “grunt” command in the Terminal. See the README.md file for instructions on how to make a custom build of Infusion.
5. The grunt command will create a zip file in the products folder. Unzip that file and move the resulting `infusion` folder somewhere convenient for your development purposes, likely in a `lib` folder in your site hierarchy.
6. This `infusion` will include a single file containing all of the JavaScript you need: `infusion-custom.js`. You will link to this file in the headers of your HTML files.

Step 1: Prepare your markup

Let's assume that you're starting with HTML markup that uses `<div>` elements and CSS to arrange the portlets in columns, like this:

```
<div class="columnSetup2 fluid-vertical-order">
    <!-- Column #1 -->
    <div class="fl-container-flex25 fl-force-left">
        <div class="demo-layoutReorderer-module demo-last-login">
            <div class="demo-layoutReorderer-module-dragbar">Login Status:</div>
            <div class="demo-layoutReorderer-module-content">
                ... module content...
            </div>
        </div>
        <div class="demo-layoutReorderer-locked">
            <div class="demo-layoutReorderer-module-dragbar">Buy ecoSkinns Today!</div>
            <div class="demo-layoutReorderer-module-content">\
                ... module content...
            </div>
        </div>
    </div>

    <!-- Column #2 -->
    <div class="fl-container-flex50 fl-force-left">
        <div class="demo-layoutReorderer-locked">
            <div class="demo-layoutReorderer-module-dragbar">The Making of a Need</div>
            <div class="demo-layoutReorderer-module-content">
                ... module content...
            </div>
        </div>

        <div class="demo-layoutReorderer-module">
            <div class="demo-layoutReorderer-module-dragbar">Lorem ipsum dolor sit amet</div>
            <div class="demo-layoutReorderer-module-content">
                ... module content...
            </div>
        </div>
    </div>
</div>
```

We'll assume you're using the Fluid Skinning System - FSS to lay out your columns, plus some styling of your own for the portlet title bars:

```
.demo-layoutReorderer-module,
.demo-layoutReorderer-locked  {
    border: 1px solid #808285;
    margin: 6px;
    background-color: #fff;
}
.demo-layoutReorderer-module-dragbar {
    /* dragbar is the bar on top of each module */
    cursor: move;
    height: 20px;
    background-color: #A5CFEF;
    background-position: right center;
    background-repeat: no-repeat;
    background-image: url("../images/move.png");
}
/* demo-module-content is the box below the dragbar,
 * which contains the actual content of the module */
.demo-layoutReorderer-module-content {
    padding: 6px;
        width: 95%;
    overflow: auto;
}
...
```

To prepare your markup for the Layout Reorderer, you need to identify three basic things that the Layout Reorderer needs to know about:

1. the container of everything,
2. the columns,
3. the reorderable modules.

(For more detailed information about this, see also Tutorial - Layout Reorderer Selectors.)

In this example, the `<div>` element that contains the two columns would be the container. We can identify it by adding, for example, a unique id such as "`demo-layoutReorderer`":

```
  ...
<div id="demo-layoutReorderer" class="columnSetup2 fluid-vertical-order">
    <!-- Column #1 -->
    <div class="fl-container-flex25 fl-force-left">
     ...
```

This ID will be used in a CSS selector passed to the Layout Reorderer creator function.

> This example uses an ID, but you could use any CSS selector. You could add a class, or use the DOM element hierarchy; so long as the CSS selector will identify a single DOM element.

To identify the columns and reorderable modules, we could use the default selectors defined by the Layout Reorderer: `.flc-reorderer-column` and `.flc-reorderer-module`. If we simply add these classes to our columns and modules, the Layout Reorderer will automatically find them:

```
<div class="columnSetup2 fluid-vertical-order">
    <!-- Column #1 -->
    <div class="flc-reorderer-column fl-container-flex25 fl-force-left">
        <div class="flc-reorderer-module demo-layoutReorderer-module demo-last-login">
            <div class="demo-layoutReorderer-module-dragbar">Login Status:</div>
            <div class="demo-layoutReorderer-module-content">
                ... module content...
            </div>
        </div>
        <div class="flc-reorderer-module demo-layoutReorderer-locked">
            <div class="demo-layoutReorderer-module-dragbar">Buy ecoSkinns Today!</div>
            <div class="demo-layoutReorderer-module-content">\
                ... module content...
            </div>
        </div>
    </div>

    <!-- Column #2 -->
    <div class="flc-reorderer-column flc-reorderer-column fl-container-flex50 fl-force-left">
        <div class="flc-reorderer-module demoSelector-layoutReorderer-locked demo-layoutReorderer-locked">
            <div class="demo-layoutReorderer-module-dragbar">The Making of a Need</div>
            <div class="demo-layoutReorderer-module-content">
                ... module content...
            </div>
        </div>

        <div class="flc-reorderer-module demo-layoutReorderer-module">
            <div class="demo-layoutReorderer-module-dragbar">Ut non turpis banana</div>
            <div class="demo-layoutReorderer-module-content">
                ... module content...
            </div>
        </div>
    </div>
</div>
```

"We already have class names on the modules, why are we adding another one?"

Infusion components make a distinction between selectors used for visual styling and selectors used for DOM manipulation. Many component change and adjust visual styling in response to user actions, so we don't want to use the same classes that are used for styling for DOM manipulation - this might result in a portlet becoming un-draggable while in the middle of being dragged!!

Locked Modules

Many portlet systems wish to designate some portlets as more important than others - they want to ensure that their visitors see the portlets. The Layout Reorderer supports the ability to lock modules in place, that is to prevent modules from being moved.

In our example, we'll use a custom selector, `demoSelector-layoutReorderer-module-locked`, to identify the "Login Status" portlet as locked, so that it always remains in the upper left corner. When we instantiate the Layout Reoderer, we'll let it know about this custom selector.

```
<div class="columnSetup2 fluid-vertical-order">
    <!-- Column #1 -->
    <div class="flc-reorderer-column fl-container-flex25 fl-force-left">
        <div class="flc-reorderer-module demoSelector-layoutReorderer-module-locked demo-layoutReorderer-module
demo-last-login">
            ...
```

Grab Handles

When visitors use the mouse to drag and drop portlets, we'd like to ensure that they can only grab the title-bar of the portlets, and not anywhere within the portlet. The Layout Reorderer supports the concept of "grab bars." In our example, we'll use a custom selector, `demoSelector-layoutReorderer`, to identify the title bars as the grab bars. When we instantiate the Layout Reoderer, we'll let it know about this custom selector.

```
<div class="columnSetup2 fluid-vertical-order">
    <!-- Column #1 -->
    <div class="flc-reorderer-column fl-container-flex25 fl-force-left">
        <div class="flc-reorderer-module demoSelector-layoutReorderer-module-locked demo-layoutReorderer-module
demo-last-login">
            <div class="demo-layoutReorderer-module-dragbar demoSelector-layoutReorderer">Login Status:</div>
            <div class="demo-layoutReorderer-module-content">
                ... module content...
            </div>
        </div>
        <div class="flc-reorderer-module demo-layoutReorderer-locked">
            <div class="demo-layoutReorderer-module-dragbar demoSelector-layoutReorderer">Buy ecoSkinns Today!<
/div>
            <div class="demo-layoutReorderer-module-content">\
                ... module content...
            </div>
        </div>
    </div>

    <!-- Column #2 -->
    <div class="flc-reorderer-column flc-reorderer-column fl-container-flex50 fl-force-left">
        <div class="flc-reorderer-module demoSelector-layoutReorderer-locked demo-layoutReorderer-locked">
            <div class="demoSelector-layoutReorderer demo-layoutReorderer-module-dragbar">The Making of a Need<
/div>
            <div class="demo-layoutReorderer-module-content">
                ... module content...
            </div>
        </div>

        <div class="flc-reorderer-module demo-layoutReorderer-module">
            <div class="demo-layoutReorderer-module-dragbar demoSelector-layoutReorderer">Ut non turpis banana<
/div>
            <div class="demo-layoutReorderer-module-content">
                ... module content...
            </div>
        </div>
    </div>
</div>
```

That's all - these are the changes you need to make to your HTML.

Step 2: Write the script

To instantiate the Layout Reorderer, you can add a script block to your file that calls `fluid.reorderLayout()`. This script will tell the Layout Reorderer about:

- the ID we added to the container
- the custom selectors we added for the locked module and the grab handles
- the names of our styles

```
<script type="text/javascript">
    jQuery(document).ready(function () {
        fluid.reorderLayout("#demo-layoutReorderer", {
            selectors: {
                lockedModules: ".demoSelector-layoutReorderer-locked",
                grabHandle: ".demoSelector-layoutReorderer"
            },
            styles: {
                defaultStyle: "demo-layoutReorderer-movable-default",
                selected: "demo-layoutReorderer-movable-selected",
                dragging: "demo-layoutReorderer-movable-dragging",
                mouseDrag: "demo-layoutReorderer-movable-mousedrag",
                dropMarker: "demo-layoutReorderer-dropMarker",
                avatar: "demo-layoutReorderer-avatar"
            },
            disableWrap: true
        });
    });
</script>
```

Some things to note about this script:

- The call to `fluid.reorderLayout()` is only done on document ready, to ensure that the mark-up is properly rendered before the Layout Reorderer tries to work with it.
- The first argument to `fluid.reordererLayout()` is a CSS selector identifying the container element. In this example, the selector uses the unique ID that we added.
- The second argument to `fluid.reorderLayout()` is the `options` argument, an object containing key/value pairs that customize the behaviour of the Layout Reorderer. In this example, we're specifying:
    - custom selectors for the locked modules and the grab handles, in the `selectors` option,
    - custom style names in the `styles` option (see below for more information on styling),
    - we wish to prevent arrow-key navigation from wrapping around the top and bottom of columns, using the `disableWrap` option.

For detailed information about the various options that can be used to configure the Layout Reorderer, see the Layout Reorderer API page.

Step 3: Add the script to your HTML

You'll need to add the Fluid library to you HTML file. In the header of the file, link to the Javascript files with `<script>` tags:

```
<script type="text/javascript" src="infusion-1.5/MyInfusion.js"></script>
```

Keep in mind that the `MyInfusion.js` file is a concatenated collection of all required source files, so it can be difficult to debug with. You may want to include each of the required files individually. This would look like this:

```
<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.draggable.js"></script>
<script type="text/javascript" src="framework/core/js/FluidDocument.js"></script>
<script type="text/javascript" src="framework/core/js/jquery/jquery.keyboard-a11y.js"></script>
<script type="text/javascript" src="lib/json/js/json2.js"></script>
<script type="text/javascript" src="framework/core/js//Fluid.js"></script>
<script type="text/javascript" src="framework/core/js/FluidView.js"></script>
<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>
<script type="text/javascript" src="framework/core/js/ReordererDOMUtilities.js"></script>
<script type="text/javascript" src="components/reorderer/js/GeometricManager.js"></script>
<script type="text/javascript" src="components/reorderer/js/Reorderer.js"></script>
<script type="text/javascript" src="framework/core/js/LayoutReorderer.js"></script>
<script type="text/javascript" src="framework/core/js/ModuleLayout.js"></script>
```

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

That's it! That's all you need to do to make your news portlets reorderable.

Styling

The Layout Reorderer defines class names that are used to update the visual appearance of the modules at various times during the life of the component. By defining styles for these classes - or by overriding them using the `styles` option, you can customize the appearance of your applications.

Style Name: "defaultStyle"
Default Class: "fl-reorderer-movable-default"
What: any item that is orderable
When: default state

Style Name: "dragging"
Default Class: "fl-reorderer-movable-dragging"
What: any item that is orderable
When: while it is in the middle of being moved using keystrokes

Style Name: "mouseDrag"
Default Class: "fl-reorderer-movable-dragging"
What: any item that is orderable
When: while it is in the middle of being dragged using the mouse

Style Name: "avatar"
Default Class: "fl-reorderer-avatar"
What: a clone of the dragged element, moved with the cursor
When: while an item is being dragged
Style Name: "selected"
Default Class: "fl-reorderer-movable-selected"
What: the element that has been selected
When: once the keyboard has been used to select the item

Style Name: "hover"
Default Class: "fl-reorderer-movable-hover"
What: any orderable element
When: while the cursor is hovering over top of it

Style Name: "dropMarker"
Default Class: "fl-reorderer-dropMarker"
What: a special element created and used to indicate where a dragged item will end up when it is dropped
When: while the cursor dragging an item
