Tutorial - Progress

This page will walk you through an example of adding the Fluid Progress component to an HTML file.

This tutorial assumes that:

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

For more general information about the Progress, see Progress. For technical API documentation, see Progress API.

Tutorial: How to Use the Progress

Scenario

This tutorial shows one example of how Progress might be incorporated into a widget, component, or application. Fluid Progress is designed to be very flexible and customizable. There are many ways to display progress. Please use our example as a jumping off point for your own creative use of Fluid Progress.

Let's assume that you have an application that does *something* – something that you know will take a certain amount of time, or has some other clear, quantifiable end-point or goal, and that the amount of progress can be measured as a percentage of that goal. Perhaps the number of bytes of a file uploaded, an amount of time passed, a percentage of numbers crunched, etc. You want to communicate to your users how far along that process is and when the process will be complete. This is what Fluid Progress is for.

This example "pops-up" a progress display as a process begins and hides the progress bar when the process is over.

> For the purpose of this tutorial we'll use a little function called `anIllusionOfProgress` who's sole purpose is to mimic an incremental process and make calls to the Fluid Progress component. `anIllusionOfProgress` is not, in any way, part of Fluid Progress and is not needed for integration with the Progress component.

There are four basic steps to adding the Progress 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

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

Status

This component is in Preview status

On This Page

See Also

- Progress
- Progress API
- Uploader

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.

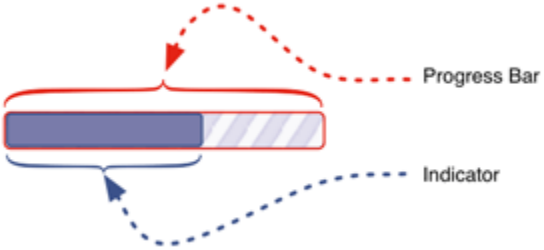
Step 1: Prepare your markup

We'll start with an HTML file called `pop-up-progress.html`. This page contains one thing, a big button that says,

Do something...

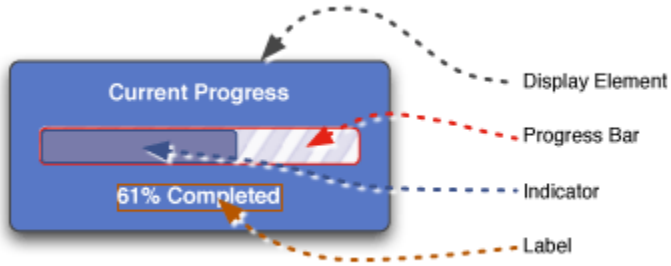
```
<div id="main">
    <p>
        <button id="showButton">Do something...</button>
    </p>
</div>
```

Now we need to add the markup that describes the Progress component.



The most basic Progress component is made up of *a bar* and *an indicator*. You can include other markup in your Progress but you must include these two elements. (Although the bar does not have to be visible.)

Our Tutorial example contains a few other elements.



The HTML looks like this:

```
<div class="container" id="main">
    <p>
        <button id="showButton">Do something...</button>
    </p>
    <div class="flc-progress progress-pop-up">
        <h3>Current Progress</h3>
        <div class="flc-progress-bar progress-bar">
            <div class="flc-progress-indicator progress-indicator">
            </div>
        </div>
        <p class="flc-progress-label progress-label">
            0% Complete
        </p>
    </div>
</div>
```

In the code above, the *indicator* element is contained within the *progress bar* element. While this is typical, it is not required. The indicator and the progress bar can be marked up quite independently of each other. The only requirement is that there be both an indicator element and a bar element as the width of the progress bar determines the final, 100%, width for the indicator.

Step 2: Write the script

For this tutorial, we'll assume that you're putting your Progress initialization and control code into a file named `progress-sample.js`. In your own integration you may want to add your Progress code to an existing Javascript file in your application.

In your file, write a function to initialize the Progress component:

```
var myProgress; // you'll need to refer to this later

jQuery(document).ready(function () {
    myProgress = fluid.progress("#main");

});
```

The code above is so simple because in this example we chose to use the defaults already defined for the Progress component. All we had to do is tell Progress where to find all it's parts – inside a container with an id of `main`. Progress knows to look for a `displayElement` with a class of `flc-progress`, and a `progressBar` with a class of `flc-progress-bar`, etc.

Let's make a small change to the default options. Let's change the default `speed` for updating the Progress indicator element. We'll make it run a bit slower and a bit smoother than the default. Like so:

```
jQuery(document).ready(function () {
    myProgress = fluid.progress("#main", {
        speed: 1000 // the default is 200, quite fast
    });
});
```

Finally we need to wire up our "do something" button to call our **`anIllusionOfProgress`** function:

```
$("#showButton").click(function () {
    anIllusionOfProgress(myProgress, 0, 20);
});
```

A full description of the public methods of Progress can be found in the Progress API.

Step 3: Add the scripts to your HTML

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

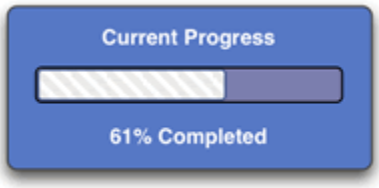
```
<script type="text/javascript" src="infusion-1./InfusionAll.js"></script>
<script type="text/javascript" src="js/progress-sample.js"></script>
```

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. 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/ui.core.js"></script>
<script type="text/javascript" src="lib/jquery/plugins/bgiframe/js/jquery.bgiframe.js"></script>  <!-- New in
v1.3 -->
<script type="text/javascript" src="framework/core/js/Fluid.js"></script>
<script type="text/javascript" src="components/progress/js/Progress.js"></script>
```

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

Step 4: Apply styles



Your progress won't look or behave like a Progress Bar without some styling. The Progress component does not come with default CSS for styling since you can build so many different kinds of progress bars with it.

For this example we're using a set of styles expressed in the progress-sample.css file from the Pop-up Progress standalone demo.

A few things to note about how we've styled the Progress elements in this example:

```
#main {
    position: relative; /* ensures the enclosed elements are positioned relative to the container */
}

.progress-bar {
    text-align: left;   /* pins the indicator to the left in IE */
}

.progress-indicator {
    height: 16px;       /* the height of the interior object will determine the
                           height of it's parent, in this case */
}
```

A brief pitch for the Fluid Skinning System and `fss-reset.css`: Styling your progress and your page will be made much easier if you take advantage of the Fluid Skinning System (FSS), especially the `fss-reset.css` which helps avoid cross-browser CSS inconsistencies.
