# Making a Content Slider with jQuery UI

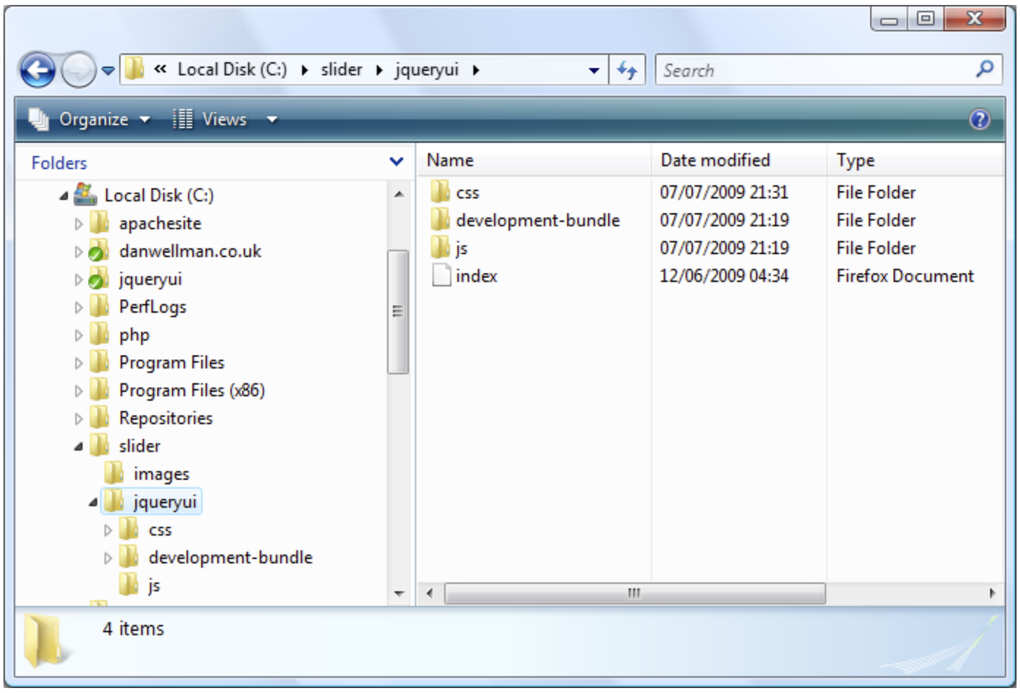
Dan Wellman on Jul 10th 2009 with 106 comments

In this tutorial we're going to be using the jQuery UI slider widget to create an attractive and functional content slider. We'll have a container, which has a series of elements each containing different blocks of content. There will be too many of these elements to display at once, so we can use the slider to move the different content blocks in and out of view.

Source
get the code

Demo
see it in action

jQuery UI is the official library of widgets and utilities built on top of jQuery; it's very easy to use, highly configurable and robust, and extremely easy to theme. To follow the tutorial you'll need a copy of the latest version of the library; it can be downloaded using the jQuery UI download builder at http://jqueryui.com/download. Although we can choose any of the themes available, I'd recommend using the default theme of smoothness. jQuery UI includes a copy of the current version of jQuery, so we don't need to download this separately.
Create a new folder somewhere handy and call it slider. Within this folder, create two new folders; one called jqueryui and one called images. Unpack the downloaded archive of the library to the jqueryui folder; in Explorer or Finder, you should end up with the following folder structure:



Local Disk (C:)
- apachesite
- danwellman.co.uk
- jqueryui
- PerfLogs
- php
- Program Files
- Program Files (x86)
- Repositories
- slider
  - images
  - jqueryui
    - css
    - development-bundle
    - js

| Name | Date modified | Type |
|---|---|---|
| css | 07/07/2009 21:31 | File Folder |
| development-bundle | 07/07/2009 21:19 | File Folder |
| js | 07/07/2009 21:19 | File Folder |
| index | 12/06/2009 04:34 | Firefox Document |

4 items

## Getting Started

Let’s make a start on the basic page and underlying HTML first; in your text editor create the following page:

view plaincopy to clipboardprint?

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN" "http://www.w3.org/TR/html4/strict.dtd">
<html>
 <head>
  <meta http-equiv="Content-Type" content="text/html; charset=utf-8">
  <title>jQuery UI Slider</title>
  <link rel="stylesheet" type="text/css" href="jqueryui/css/smoothness/jquery-ui-1.7.2.custom.css">
  <link rel="stylesheet" type="text/css" href="slider.css">
 </head>
 <body>
  <div id="sliderContent" class="ui-corner-all">
   <h2>Some well known galactic nebulae and their vital statistics</h2>
    <div class="viewer ui-corner-all">
     <div class="content-conveyor ui-helper-clearfix">

     <div class="item">
      <h2>Omega Nebula</h2>
      <img src="images/omega.jpg" alt="Omega Nebula">
      <dl class="details ui-helper-clearfix">
       <dt>Distance from Earth:</dt><dd>5000 - 6000 lightyears</dd>
       <dt>Diameter:</dt><dd>15 Lightyears</dd>
       <dt>Mass:</dt><dd>800 solar masses</dd>
       <dt>Catalogue number:</dt><dd>M17 / NGC6618</dd>
       <dt>Discovered in:</dt><dd>1764</dd>
       <dt>Discoverer:</dt><dd>Philippe Loys de Chéseaux</dd>
      </dl>
     </div>

    </div>
   </div>
   <div id="slider"></div>
  </div>
  <script type="text/javascript" src="jqueryui/js/jquery-1.3.2.min.js"></script>
  <script type="text/javascript" src="jqueryui/js/jquery-ui-1.7.2.custom.min.js"></script>
 </body>
</html>
```

Save this as slider.html in the slider folder. In the head of the page, we link to the jQuery UI style sheet, which contains all of the CSS that’s required for each of the library components. It may seem like a waste; in some ways it is as we’re only using a single component, but using a 26KB style sheet. However, using a tool like YUICompressor, we can easily shrink this, and with GZipping too we can get it down even further. We also link to our own custom style sheet, which we’ll create later.

We haven’t added any styling yet but for reference, the following screenshot shows the default slider widget:

## The Underlying Mark-up

On the page all we have is the mark-up for the content and the slider; we've got an outer container element which we've given the class name ui-corner-all. This is one of the classes targeted by the jQuery UI style sheet and will give our container (and the other elements we give it to) nice rounded corners. It uses CSS3 to achieve this so not all browsers are supported, but Firefox, Safari or Chrome users will see them.

Within the container we've got a heading element that describes the content, followed by another container element (which will also have rounded corners in supporting browsers); when we come to add the CSS, this element will be given an overflow rule of hidden which will hide most of the individual content blocks and allow us to slide them into view using the slider. This element will function as the viewer.
Within the viewer we have a final container element; the reason for this is for performance – when we adjust the left CSS property with jQuery, we'll only be selecting and manipulating one element instead of however many content blocks there are. We use another class name from the UI library on this element – the ui-helper-clearfix class, which automatically clears floated elements within whichever element it's applied to.

Following this is an example of a content block; I've only shown one of them in the code example above because to show more would be unnecessary repetition. In the source file there are seven of them, but you can put as many in as you like and the slider will still function as it should. Each content block contains a heading, an image and a definition list, which semantically is probably the best choice for this example, but not necessarily required in other implementations. The content blocks can feature pretty much whatever they need to, provided each container is of a fixed size; you'll see why this is important we come to add the JavaScript a little later on.

After the viewer element comes an empty container which will be transformed into the slider widget once we invoke the UI library. This is all underlying HTML that we'll need. Following this we link to jQuery and to the jQuery UI source files; again, this file contains all of the JavaScript needed to run the whole UI library, which for this tutorial is more than we need. There are individual files for the core and each component separately which can cut down the footprint of the library. Both the jQuery and jQuery UI JS files are already minified.

## Styling the Content

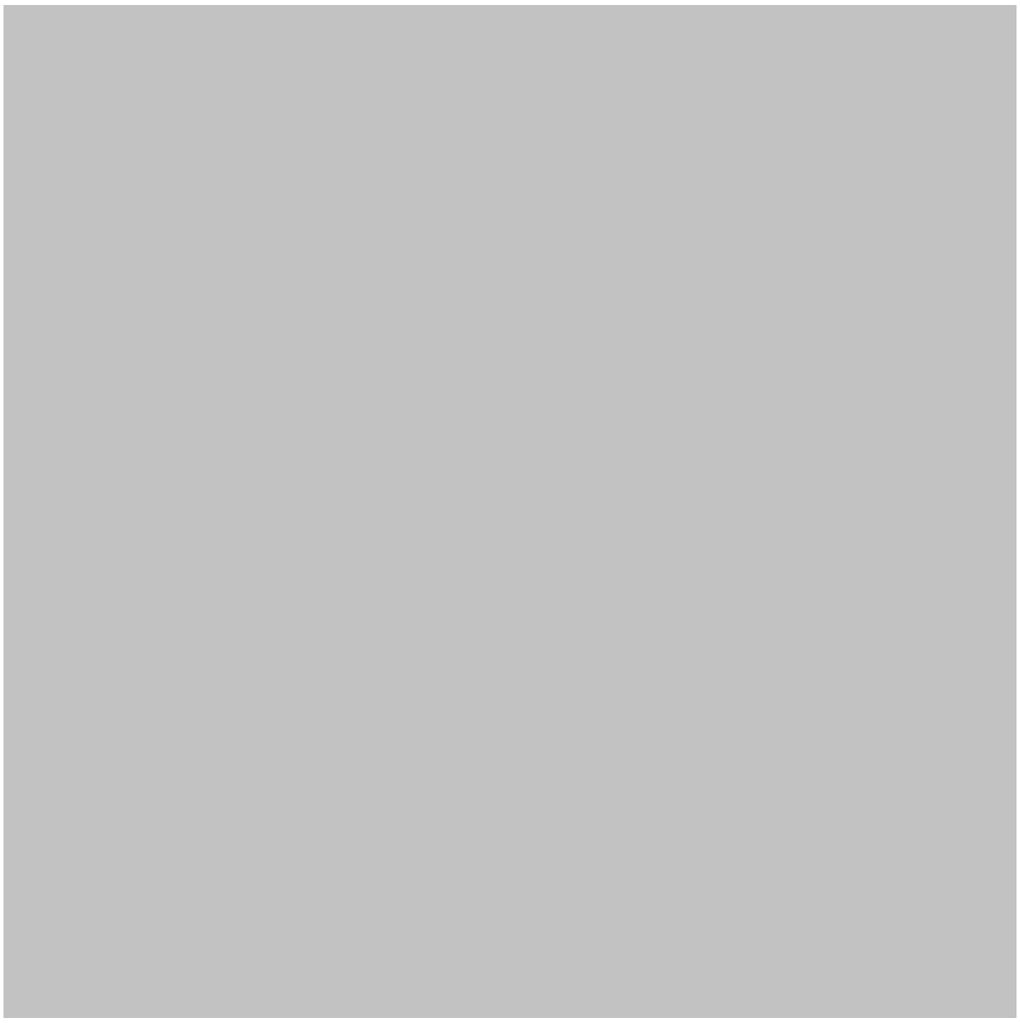
In truth we don't need to worry about styling the slider widget itself at all; the theme that we downloaded with the library will do that for us. The CSS we're about to add is pretty much purely arbitrary for the purpose of this tutorial, to tidy things up and give it a basic minimal look. As long as the individual content blocks (given a class name of item) are given a fixed width and are floated to the left within the conveyor element, and provided the viewer has its overflow set to hidden everything should work as expected.

In a new file in your text editor add the following code:

view plaincopy to clipboardprint?

```
h2 { text-align:center; font:normal 150% Georgia; }
#sliderContent {
  width:650px; margin:auto; padding:0 50px 50px; background-color:#ebebeb;
  border:1px solid #898989;
}
.viewer {
  width:607px; height:343px; margin:0 auto 40px; padding:1px; overflow:hidden;
  position:relative; border:1px solid #898989;
}
.content-conveyor { width:610px; height:335px; position:relative; }
.item {
  width:304px; float:left; font-family:Tahoma; text-align:center;
  background-color:#ebebeb;
}
.item h2 { font-size:100%; margin:10px 0; }
.item dl { margin:10px 0; }
.item dt, .item dd {
  float:left; width:149px; text-align:rightright; margin:0; font-size:70%;
}
.item dt { font-weight:bold; margin-right:5px; }
.item dd { text-align:left; }
.item img { border:1px solid #898989; background-color:#ffffff; padding:1px; }
```

Save this as slider.css in the slider folder. Our page should now look like this:



## Adding the Slider Widget

All we need to do now is add the JavaScript that will initialise the slider and control our content blocks. Directly after the script element linking to jQuery UI in slider.html add the following code:

view plaincopy to clipboardprint?

```
<script type="text/javascript">
 $(function() {

  //vars
  var conveyor = $(".content-conveyor", $("#sliderContent")),
  item = $(".item", $("#sliderContent"));

  //set length of conveyor
  conveyor.css("width", item.length * parseInt(item.css("width")));

  //config
  var sliderOpts = {
    max: (item.length * parseInt(item.css("width"))) - parseInt($(".viewer", $("#sliderContent")).css("width")),
    slide: function(e, ui) {
      conveyor.css("left", "-" + ui.value + "px");
    }
  };

  //create slider
  $("#slider").slider(sliderOpts);
 });
</script>
```

It’s a very short, simple snippet of code, with very little going on; let’s take a look at it line by line; Within the document.ready short-cut we first set up a few variables so that we can cache the elements from the page that we’ll be manipulating for performance reasons; this makes our code run faster because we’re only traversing the DOM and selecting each element once.

We select the conveyor element first of all by targeting its class name; because using a class selector is inefficient, we give the selector a context of the sliderContent element. The context is provided using an id selector, so the whole DOM does not need to be traversed. We also select the collection of content blocks in the same way.

Once we’ve cached our selectors we can set the length of the conveyor element; in the CSS it was set to the width of two of the content blocks, but for it to function correctly, the content boxes need to float alongside each other, so the conveyor needs to be wide enough to accommodate them all.

So that we don’t restrict how many content blocks can be put into the widget we don’t hardcode a set width into it; instead, we get the number of content blocks, and multiply this by the width of each block. This is why it’s important to set a fixed width on the blocks. We need to use JavaScript’s parseInt function when we retrieve the width of the blocks because the jQuery css method returns a string value in getter mode.

Next we create a literal configuration object which will be passed into the jQuery UI slider method and used to set some properties of the slider widget. Our configuration object has two properties, max and slide. The max property’s value is an integer which represents the width of the conveyor element minus the width of the viewer. This will be the maximum value that the slider handle can reach.
The value of the slide property is an anonymous function which will automatically receive two arguments; the original event object and a prepared object containing useful properties relating to the widget. We don’t use the first argument at all, which we define as e, but we need to include it to gain access to the second argument, which we term ui.

The slide event is a custom event exposed by the slider API, and the function we set as its value will be called each time a slide interaction occurs. Whenever the event is fired, we simply manipulate the left style property of our conveyor element negatively by the same amount as the slider is moved. We can get the value that the slider is moved to using the value property of the ui object.

We set the maximum value of the slider to the length of the conveyor element, in this example it ends up being 2128px, so the maximum value is 2128. This isn’t in pixels, as you’ll see in the next screenshot, the slider itself is around 650px in length. But, if we move the slider to about halfway along the track, the value reported in the ui object will be around 1064, so we move the left edge of the conveyor this many pixels to the left or right.

We don’t need to worry about detecting which direction the slider was moved in; if the slider handle has already been moved to the right, the left CSS property if the conveyor will already have a negative value. When we minus a negative number from a negative number, the outcome is of course a positive number so the conveyor will move back as it should. The completed page should now appear featuring the slider:

You should find that it works as expected and the different blocks of content can be moved in and out of view using the slider widget. As well as the standard drag interaction, also built into the slider is the useful addition of a click interaction; if click anywhere on the track, the handle is automatically moved to that position and the slide callback function is executed.

## Conclusion

In this tutorial we've looked at how the underlying HTML used for the slider (a simple empty container), the default styling applied by the library, and how it can be configured and initialized with our code.

The slider is a great addition to any interface; it's easy for us to set up and easy for our visitors to use, it's tactile and interactive and can be used in a variety of situations from moving content around like in this example, or as, say, a volume control on a streaming web app.

- Follow us on Twitter, or subscribe to the NETTUTS RSS Feed for more daily web development tuts and articles.

www.basekit.com Ads by Google

### By Dan Wellman

Dan Wellman has been writing web-design and scripting tutorials for approximately 7 years. His new book, jQuery UI 1.7: The User Interface library for jQuery, was released at the end of 2009.