


jQuery Sample

by Jeremy Fonte

 +27 Recommend this on Google[Tweet](#)

Welcome to jQuery Sample, home to all of the jQuery, AJAX, and jQuery UI examples and sample code you need. Just quality content - and 100% free.

All source code on this website is released under the [MIT License](#).

Select a topic:

- AJAX
 - [Basic AJAX Techniques](#)
 - [Generating a Table with AJAX and XML](#)
- Effects
 - [jQuery Effects: slide, fade, and animate](#)
- Events
 - [Capturing and Programming for Mouse Events](#)
- DOM Traversal/Manipulation
 - [Basic DOM Traversal with jQuery](#)
- Projects
 - [Creating a Drop-down Menu with jQuery](#)
- jQuery UI
 - [Using jQuery UI's Progress Bar](#)
 - [Creating an Accordion Menu with jQuery UI](#)

To learn about my other projects, please visit my blog at <http://fonte.me>

[Privacy Policy](#)

My Other Projects:

[Theory](#) - A Theoretical Math Library

[Fonte Labs](#) - Programming Experiments

[Fonte Muse](#) - Writings and Poems

[Web Audio Player](#) - Stream My Original Music

[JavaScript Example](#) - Learn JavaScript

[HTML and CSS](#) - HTML and CSS Tutorials

[Fonte Labs Podcast](#)

Basic AJAX Techniques

In this post, I'll be explaining the basics of making AJAX requests using jQuery, JavaScript and HTML. A quick summary of what AJAX is: the web browser sends an HTTP request to the server for a specified file or web service; the server then, generally, sends back some data or a file. Now, let's review the simpler AJAX commands jQuery provides.

1. The simplest version of `.load(url)` - this method is distinct from the other AJAX methods in that it is called on a DOM element, rather than being a global function. The response from the server is then injected into the provided DOM element. This means that you precede the `.load()` command with an element, for example:

HTML Code

```
<div id="myDiv">
  <!-- AJAX content will load here-->
</div>
```

JavaScript Code

```
//load the contents of myfile.txt into #myDiv
$('#myDiv').load('myfolder/myfile.txt');
```

This AJAX method also supports a "data" parameter, which lets you send data to the server, in the HTTP GET request that this method uses. The server can then determine what file or data to send back to the browser, based on what the data parameter was.

Try it out!

Load data from server into the div

2. A more versatile AJAX command provided by the jQuery library is `$.get()`. This method sends a get request to the specified URL, and is a global function (hence the "\$."). You provide a callback function as the second parameter, which lets you accomplish a task using the data

passed back from the server. Here's an example:

JavaScript Code

```
$.get('example/myDoc.txt', function(data) {  
    alert(data);  
});
```

Try it out!

Load data using AJAX - \$.get()

Generating a Table with AJAX and XML

jQuery allows you to navigate XML using the same DOM functions used for traversing HTML documents. In this project, we'll be retrieving an XML file, traversing it with jQuery's DOM traversal methods, and creating a table based on the XML data.

1. Our first task in this project is to retrieve the XML from the server using AJAX. We'll be using the \$.get AJAX method to grab the XML. Next, we'll use jQuery's DOM traversal methods to walk through the XML DOM - and as we go through the XML, we'll generate rows and columns in the HTML.

XML Data

```
<?xml version="1.0" encoding="UTF-8"?>  
<tabledata>  
  <row>  
    <column>1</column>  
    <column>2</column>  
    <column>3</column>  
  </row>  
  <row>  
    <column>4</column>  
    <column>5</column>  
    <column>6</column>  
  </row>  
  <row>  
    <column>7</column>  
    <column>8</column>  
    <column>9</column>  
  </row>  
</tabledata>
```

HTML Code

```
<div id="tableWrapper">  
  
</div>
```

JavaScript Code

```
$('#loadTable').click(function() {  
    $.get('example/tableData.xml', function(data) {  
        $('#div#tableWrapper').append('<table id="ajaxTable"></table>');  
        $(document).ready(function() {  
            $(data).children('tabledata').children('row').each(function() {  
                var thisRow = this;  
                $('#table#ajaxTable').append('<tr></tr>');  
                $(thisRow).children('column').each(function() {  
                    var thisColumn = $(this).text();  
                    $('#table#ajaxTable').children('tbody').children('tr').last().append('<td style="border: 1px solid black">' + thisColumn + '  
                });  
            });  
        });  
    });  
});
```

Try it out!

Load data into table using AJAX

jQuery Effects: Slide, Fade, and Animate

jQuery comes with a handful of excellent CSS-based animation effects. This includes sliding elements up and down/into and out of view, fading elements in and out, and performing custom animations of other CSS properties. Let's start with the sliding effects.

1. jQuery provides three commands for sliding elements in and out of view, and all of their names make the functionality fairly self-explanatory. "slideUp" slides an element up and out of view, "slideDown" slides an element down and into view - a common use for

these methods is to make a drop-down menu slide down when it's parent link is hovered over, and to have it slide up when a different menu item is being hovered over. Finally, "slideToggle" intelligently alternates between sliding an element up and sliding it down, each time "slideToggle" is called.

For the sliding effects (and the fading effects we'll cover next), you can provide either the argument 'slow', which is equivalent to 600ms, no argument which is equal to 400ms, or 'fast' which is equivalent to 200ms. You can also specify a custom time length by providing the number of milliseconds you'd like the effect/animation to take.

Here's a functional demo of sliding up, sliding down, and toggling sliding:

HTML Code

```
<div id="slideMe">
  This content will appear and disappear when the div is slid in and out.
</div>
<!-- One button for each slide command -->
<input id="btnSlideUp" value="Slide Up" type="button">
<input id="btnSlideDown" value="Slide Down" type="button">
<input id="btnSlideToggle" value="Slide Toggle" type="button">
```

JavaScript Code

```
$('#btnSlideUp').click(function() {
  $('#slideMe').slideUp('slow');
});
$('#btnSlideDown').click(function() {
  $('#slideMe').slideDown();
});
$('#btnSlideToggle').click(function() {
  $('#slideMe').slideToggle('fast');
});
```

Try it out!

This content will appear and disappear when the div is slid in and out.

2. There are two primary fading commands in jQuery: fadeIn(), and fadeOut(). When jQuery fades an element out or in, it basically reduces the opacity of the element to nothing when fading out, and then hides it all together so other elements can reposition themselves. Likewise, fading an element in brings it back into position and then fades it to full opacity.

HTML Code

```
<div id="fadeMe">
  This content will appear and disappear when the div is faded in and out.
</div>
<!-- One button for each fading command -->
<input id="btnFadeOut" value="Fade Out" type="button">
<input id="btnFadeIn" value="Fade In" type="button">
```

JavaScript Code

```
$('#btnFadeIn').click(function() {
  $('#fadeMe').fadeIn('slow');
});
$('#btnFadeOut').click(function() {
  $('#fadeMe').fadeOut();
});
```

Try it out!

This content will appear and disappear when the div is faded in and out.

3. Animate anything, animate everything - that's the power that jQuery's animate() method provides. Call this method on any jQuery selector, and provide an object where the css property is the key, and the value is what value to animate the property to. If you want to animate color, you'll have to use the jQuery Color plugin - but many other CSS properties can be animated right out of the box. In this example we'll alter the CSS properties of a div element.

HTML Code

```
<div id="animateMe">
  Watch me be animated!
</div>
<!-- One button for each fading command -->
<input id="btnAnimate" value="Animate It" type="button">
<input id="btnAnimateBack" value="Animate It Back" type="button">
```

CSS Code

```
div#animateMe {
  left: 0;
  top: 0;
  position: relative;
  width: 300px;
  height: 100px;
  border: 1px solid black;
  background-color: teal;
}
```

JavaScript Code

```
$('#btnAnimate').click(function() {
  $('#animateMe').animate({
    'left': '300px',
    'top': '200px',
    'border-width': '8px'
  }, 1000);
});
$('#btnAnimateBack').click(function() {
  $('#animateMe').animate({
    'left': '0',
    'top': '0',
    'border-width': '1px'
  }, 1000);
});
```

Try it out!

Watch me be animated!

Animate It

Animate It Back

Capturing and Programming for Mouse Events

jQuery has a full set of mouse event methods, including click(), dblclick(), hover(), mouseover(), mouseout(), and more. In this post I'll be covering a wide array of mouse methods that help to make your web app highly interactive and responsive.

1. Arguably the most important mouse event, .click() lets you attach custom functionality to the click event of the chosen element - a div, a button, or pretty much any other element. Once the event handler for the click event is attached to the element, your custom code will be triggered whenever that element has a click event fired.

HTML Code

```
<div id="clickMe">
  Click here and watch the text change!
</div>
```

CSS Code

```
div#clickMe {
  width: 300px;
  height: 100px;
  margin: 10px;
  background-color: green;
}
```

```
color: white;
border: 1px solid black;
}
```

JavaScript Code

```
$(document).ready(function() {
    //e is the event variable, can prevent the default behavior
    //of the event or stop it from propagating up to the element's
    //parent elements
    $('#div#clickMe').click(function(e) {
        var oldText = $(this).text();
        $(this).text(oldText + 'Click added some text!');
    });
});
```

Try it out!

Click here and watch the text change!

2. Another essential mouse interaction that programmers need to handle is the "hover" event. In actuality, this interaction consists of two events: first the mouse is moved over the element, and then the mouse leaves the element. These two events on their own are .mouseenter and .mouseleave, and jQuery's "hover" event combines these two events in one jQuery method.

The syntax for the hover event is a jQuery object, such as \$('#myDiv'), followed by ".hover(function(e){ }, function(e){ })". An example follows:

HTML Code

```
<div id="hoverDiv">
    hover over me!
</div>
```

CSS Code

```
div#hoverDiv {
    width: 300px;
    height: 100px;
    background-color: red;
    color: white;
    border: 1px solid black;
}
```

JavaScript Code

```
$(document).ready(function() {
    //this first function inside the hover method
    //is for when the mouse enters the element
    $('#div#hoverDiv').hover(function() {
        //set multiple CSS properties at once
        $(this).css({
            'background-color': 'blue',
            'color': 'pink'
        });
    }, function() {
        //set multiple CSS properties at once
        $(this).css({
            'background-color': 'red',
            'color': 'white'
        });
    });
});
```

Try it out!

hover over me!

Basic DOM Traversal with jQuery

If you're using JavaScript in the browser, working with the DOM effectively is an ever-present issue. Native JavaScript methods like "getElementById", "childNodes", and so on work well enough, but leave a lot to desire in flexibility and dev-friendly design.

The fundamental structure of the DOM is that of a tree. Any given node in the DOM can have parent nodes, child nodes, and sibling nodes. Nodes can also be related by being the next or previous node in relation to a sibling node. In this post I'll outline the key jQuery methods that allow a developer to seamlessly traverse the DOM.

1. The first DOM methods we'll examine are used to access the parents or children of a given node. Direct parent nodes can be accessed using the .parent() jQuery method - with a selector available to filter the results. For example, \$('p').parent('div') will access the direct parent of the paragraph element, which is expected to be a div.

If you need to access prior parents/ancestors that are more than one level up the DOM tree, you can use the .parents() traversal method - note that it is parents (plural), as opposed to the singular .parent() method.

Children of a given node can be accessed using the .children() method, also optionally filtered by a selector. If you need to traverse multiple levels under a given node, and find nodes of a specific type, you can use the .find() method to zoom in on those descendants.

HTML Code

```
<div id="parentNode" class="treeNode">
  Parent node's text.

  <div id="mainNode" class="treeNode">
    The target node

    <div id="childNode" class="treeNode">
      The main node's child node
    </div>
  </div>
</div>

<input value="Copy Parent Text" id="CopyParent" type="button">
<input value="Copy Child Text" id="CopyChild" type="button">
```

CSS Code

```
div#parentNode {
  width: 600px;
  height: 250px;
  z-index: 10;
}

div#mainNode {
  width: 400px;
  height: 180px;
  z-index: 20;
}

div#childNode {
  width: 300px;
  height: 100px;
  z-index: 30;
}

div.treeNode {
  border: 1px solid black;
  color: black;
  background-color: lime;
  margin: 20px;
}
```

jQuery Code

```
//Use DOM traversal methods to copy the parent node's text to the main node
$('#input#CopyParent').click(function() {
  var currentNodeHTML = $('#div#mainNode').html()
  var parentText = $('#div#mainNode').parent('div').text();
  $('#div#mainNode').html(parentText + currentNodeHTML);
})

//Use DOM traversal methods to copy the child node's text to the main node
$('#input#CopyChild').click(function() {
```

```
var currentNodeHTML = $('#div#mainNode').html()
var childText = $('#div#mainNode').children('div').text();
$('#div#mainNode').html(childText + currentNodeHTML);
})
```

Try it out!

Parent node's text.

The target node

The main node's child node

Copy Parent Text

Copy Child Text

Creating a Drop-down Menu with jQuery

jQuery UI has recently added a menu widget to its array of UI controls - however, sometimes jQuery UI can be overkill for a project - and for this reason, I'm going to outline how to make a drop-down menu with jQuery, HTML, and CSS.

1. The first step to creating a jQuery menu is to write the markup for the structure of the menu. A div is wrapped around the menu, and an unordered list forms the various nodes of the menu. Finally, anchors inside the list items provide the hyperlinks that make the drop-down menu functional.

HTML Code

```
<!-- All of the links on this demo menu are preceded by a hash - in production, they should be links to actual pages, i.e., a.html, b.php, etc-
<div id="menuDemo">
  <ul>
    <li id="homeMenu"><a href="#menuHome">Home</a>
    </li>
    <li><a href="#">Pages</a>
      <ul>
        <li><a href="#menuPagesOne">Page Number One</a></li>
        <li><a href="#menuPagesTwo">Page Number Two</a></li>
        <li><a href="#menuPagesThree">Page Number Three</a></li>
      </ul>
    </li>
    <li><a href="#">Links</a>
      <ul>
        <li><a href="#LinkNumber1">Link Number 1</a></li>
        <li><a href="#LinkNumber2">Link Number 2</a></li>
        <li><a href="#LinkNumber3">Link Number 3</a></li>
      </ul>
    </li>
  </ul>
</div>
```

CSS Code

```
div#menuDemo2 {
  width: 800px;
  height: 30px;
  position: relative;
  left: 0;
  top: 0;
  background-color: orange;
  color: white;
  border-bottom: 1px solid black;
  /*Remove this next one in production - Used for demo purpose only*/
  margin-bottom: 300px;
}

div#menuDemo2 ul {
  list-style: none;
  margin: 0;
  padding: 0;
  background-color: orange;
```

```
}

div#menuDemo2 > ul > li {
    float: left;
}

div#menuDemo2 ul li {
    width: 267px;
    background-color: orange;
}

div#menuDemo2 ul li#homeMenu {
    width: 266px;
    background-color: orange;
}

div#menuDemo2 ul li a {
    width: 266px;
    text-decoration: none;
}

div#menuDemo2 > ul > li > ul {
    display: none;
    background-color: orange;
    width: 267px;
}

div#menuDemo2 > ul > li:hover > ul {
    display: block;
}
```

[Try it out!](#)[Home](#)[Pages](#)[Links](#)

2. This drop-down menu is a simple implementation of what is essentially an HTML/CSS drop-down menu. Using jQuery, we can add some polished effects to the menu. Let's add a slide down effect when the user hovers over a top-level menu category, and a fade out effect when the user leaves the menu.

JavaScript Code

```
$('#div#menuDemo3 > ul > li').hover(function() {
    //effect when the user hovers over the menu
    //first hide the menu item, since the CSS displays it - then slide it down.
    $(this).children('ul').hide().slideDown();
}, function() {
    //effect when the user leaves the current menu area - fade out
    $(this).children('ul').fadeOut();
});
```

[Try it out!](#)[Home](#)[Pages](#)[Links](#)

Using jQuery UI's Progressbar Widget

If you need a control or widget to signify the progress of a task on our web app, jQuery UI's Progressbar widget is a simple and effective solution. To use this widget you'll need to include the jQuery UI library from a CDN, or download it and host it on your website.

Using the Progressbar widget is very simple. You initialize it by calling the Progressbar() method on a div element. There are a handful of parameters you can pass to the new progressbar as an object argument (key/value pairs). The most important of these is the "value" parameter, which determines the starting value of the progressbar. Starting with jQuery UI 1.10 Beta, if you supply false to the value, it will become an indeterminate progressbar - basically just a sign that something is working, rather than a direct measure of how much of the task is completed.

HTML Code

```
<div id="ProgContainer">
  <div id="progressbar"></div>
  <input id="btnProgress" value="Progress Bar +10%" type="button">
</div>
```

JavaScript Code

```
$(document).ready(function() {
  var progValue = 10;
  $('#progressbar').progressbar({
    'value': 10
  });

  $('#btnProgress').click(function() {
    progValue = $('#progressbar').progressbar('value');
    progValue += 10;
    $('#progressbar').progressbar({
      'value': progValue
    });
  });
});
```

Try it out!

Progress Bar +10%

Creating an Accordion Menu with jQuery UI

The accordion menu is one of the nicer UI elements that jQuery UI provides. It consists of multiple rows, each of which has a title, and any of them can be expanded to reveal the inner content of that item - but generally, only one row can be expanded at a time.

1. The simplest accordion menu is one with few custom options set - and that's the vanilla version we'll be testing out to start. Here's the breakdown: each accordion menu has a title (which is an h3 element), and a "contents" section, which is a div immediately following the h3 header. Finally, the whole package is wrapped in a div, and JavaScript code calls the accordion constructor function on that div.

HTML Code

```
<div id="accordion">
  <!-- First accordion menu item-->
  <h3>Step One</h3>
  <div>
    <p>
      Learn jQuery!
    </p>
  </div>

  <!-- Second accordion menu item-->
  <h3>Step Two</h3>
  <div>
    <p>
      Learn jQuery UI!
    </p>
  </div>

  <!-- Third accordion menu item-->
  <h3>Step Three</h3>
  <div>
    <p>
      WRITE EPIC CODE!!!
    </p>
  </div>
</div>
```

JavaScript Code

```
$(function() {  
    $( "#accordion" ).accordion();  
});
```

Try it out!**Step One**

Learn jQuery!

Step Two**Step Three**

2. There are many custom options available for the accordion menu, and in this section I'll apply a few of the nicer ones to create a customized menu. The first customization is the "collapsible" option, which I'll set to true - this allows any and all menu items in the accordion to be closed, including the active panel - which results in all panels being closed. The second customization is the "active" option, set to false, so that when the accordion menu loads, all panels will be shut. The final option is "heightStyle" set to "content" - this dictates that each accordion panel will only be as tall as its content requires.

HTML Code

```
<div id="customAccordion">  
    <!-- First accordion menu item-->  
    <h3>Custom Feature #1</h3>  
    <div>  
        <p>  
            Allow collapsible menus - including the active one.  
        </p>  
    </div>  
  
    <!-- Second accordion menu item-->  
    <h3>Custom Feature #2</h3>  
    <div>  
        <p>  
            Sets active panel to false, making the menu start out with all menu items closed.  
        </p>  
    </div>  
  
    <!-- Third accordion menu item-->  
    <h3>Custom Feature #3</h3>  
    <div>  
        <p>  
            Make each inside panel only as tall as it's content requires.  
        </p>  
    </div>  
</div>
```

JavaScript Code

```
$('#div#customAccordion').accordion({  
    collapsible: true,  
    active: false,  
    heightStyle: 'content'  
});
```

Try it out!**Custom Feature #1****Custom Feature #2****Custom Feature #3**