|  |
| --- |
| [[https://myetudes.org/etudes-melete-tool/images/printer.png](https://myetudes.org/portal/tool/4c4d3792-8b10-40ce-8016-d7a5ac569a1c/print_module.jsf?printModuleId=1436385325) Send to Printer](https://myetudes.org/portal/tool/4c4d3792-8b10-40ce-8016-d7a5ac569a1c/print_module.jsf?printModuleId=1436385325) | [Close Window](https://myetudes.org/portal/tool/4c4d3792-8b10-40ce-8016-d7a5ac569a1c/print_module.jsf?printModuleId=1436385325) |
|  | |
| 14. Libraries and Frameworks  14.1. Overview  *Copyright (c) 2014, Rula Khayrallah*  So far in this course, we have focused on writing JavaScript code that works in most of the current browsers.   This is not a luxury we have in the industry.  The challenge in client side JavaScript programming is to deal with all the browser inconsistencies.  For this reason, it is helpful to build JavaScript code on top of some of the available frameworks.  These frameworks provide a higher level interface that is compatible and tested across all browsers.  Several JavaScript frameworks are available today.  Here are just a few:  Prototype:  <http://prototypejs.org/>  The Prototype library includes DOM and Ajax utilities.  **YUI** – <http://yuilibrary.com/>  YUI is the Yahoo!  User Interface Framework.  It is an open source JavaScript library for building interactive web applications.  It includes language utilities, DOM utilities, as well as user interface widgets.  **Dojo** – <http://dojotoolkit.org/>  Dojo is also an open source library that includes UI widgets, DOM manipulation utilities, a system for managing module interdependencies and more.  Closure -  <https://developers.google.com/closure/library/>  The closure library is the JavaScript library that Google uses for Gmail.  jQuery - <http://jquery.com/>  Although jQuery does not do anything JavaScript cannot do, it has become extremely popular for several reasons:  It uses a simple syntax (CSS selectors) for referring to elements in the document.  It provides a simple way to operate on sets of elements as a group, rather than one at a time.  It removes the need to deal with the browser incompatibilities in our code.  There are several open source plugins available for jQuery, including a popular user interface toolkit jQuery UI.  We’ll cover jQuery in more details in the next few sections.  14.2. Getting Started with jQuery  *Copyright (c) 2014, Rula Khayrallah*  In this course, we’ll download the jQuery library to our scripts directory and then refer to it in our web pages with the src attribute of the <script> tag.  This is called serving our own copy.  In production code, we can also use a hosted version from a Content Delivery Network such as Google or Microsoft:  the main advantage of doing that is that users will likely already have a copy in the browser’s cache and no download will be necessary.  To download jQuery go to:  <http://jquery.com/>  We’ll use the uncompressed, development jQuery 1.11.1 since that version is best for debugging purposes.  As you can see, that’s just a JavaScript file.  We’ll save it in our scripts directory as jquery-1.11.1.js.  To use jQuery, we now include the following in our HTML document:  < script defer src =" ../scripts/jquery-1.11.1.js" > </ script >  Note that as long as we use the defer attribute it does not matter whether we include the script tag in the header or the body element. **When our own code uses jquery, jquery has to be loaded before our own script can start execution.**  14.3. jQuery Demo  *Copyright (c) 2014, Rula Khayrallah*  jQuery provides a simple way to select some elements and do something with them.  The basic syntax is:  **$(selector).action();**  The $ is an alias for the function called jQuery, so we could also use the following construct:   jQuery(selector).action();  jQuery supports most CSS3 selectors.  To follow along, download the following files available under Resources:  jqdemo.html  jqdemo.css  tree.png  car.png  You can then open jqdemo.html in Firefox, and use the Firebug console to try the commands below.  We can **select elements by their HTML tags**:  >>> $('p');   // all the <p> elements in the document  Object[p, p, p]  We can **select elements by their id**:  >>> $('**#**profile');  //the selector matches the element with id "profile"  Object[div#profile]  We can **select elements by class name:**  >>> $('.important'); // the selector matches all elements with class important.  Object[span.important, li.important]  We can combine selectors:  $('li.important'); // the selector matches all li (list item) elements with class important  Object[li.important]  The selector below matches all <input> elements that are descendants of an element that has the id profile.  >>> $("#profile input")  Object**[**input#name property value = "" attribute value = "null", input#email property value = "" attribute value = "null"**]**  Notice the square brackets around the object returned by $.  **The jQuery object returned by $ is an array like object**.  So to check whether a selection contains any elements, we can check its length property:  >>> $('.important').length;  2  >>> $('h1').length;  0  To **access a particular element** in that returned object, we can **use the square brackets**.  >>> $('.important')[0];  <span class="important">  >>> $('.important')[1];  <li class="important">  Once we have a selection,**we can call jQuery methods on the selection**. Note however that we cannot use the DOM element properties such as textContent, innerHTML and value directly on the selection, since that selection is an array like object, NOT a DOM element.  When a jQuery method is used to get (or read) a value, it is called a getter. **In general, getters only get the value of the first element in the selection.**  When a method is used to set a value, it is called a setter.  **Setters affect all elements in a selection.  Setters also return the jQuery object allowing us to chain methods.**  Here are just some examples of the methods available on jQuery objects.  For a full list,  check out <http://api.jquery.com/> .  **Note that jQuery uses the same method name for getter and setter.** A getter takes no argument.  A setter takes one.  Unlike other jQuery getters, the text method returns the text content of**all the elements** in the selection.  $('li').text(); / /text getter – get the text content of all **li elements.**  >>> $('li').**text**();  "It uses a simple syntax (CSS selectors) for referring to elements in the document. It provides a simple way to operate on sets of elements as a group, rather than one at a time.It removes the need to deal with the browser incompatibilities in our code.There are several open source plugins available for jQuery, including a popular user interface toolkit jQuery UI.  "  $('li').text('New Text'); //text setter - set the text content **of all li elements.**  Note that the web page has been modified and all the list items now consist of 'New Text'.  Make sure you reload the page to get back to the original text before proceeding with the examples below.  >>> $('li').**html**();  // get the innerHTML of the first list element  "It uses a simple syntax (CSS selectors) for referring to elements in the document. "  Enter a name and an email address in the input fields, then try the following:  >>> $('input').**val**() ; // val getter - get the value of **the first input element**  "Rula"  $('#profile input').val(''); // set the value of all the #profile input elements to the empty string  Note that the two input fields are cleared.  We can manipulate CSS classes with jQuery, adding a class, removing a class or toggling between classes:  $('div').**addClass**('important');  // add the class important to all <div> elements  Object[div#profile.important, div#lesson.important, div.important, div.important]  Note the change in style in the corresponding <div> elements.  >>> $('div').**removeClass**('important');  // remove the class important from div elements  Object[div#profile, div#lesson, div, div]  With the **toggleClass** method, we can switch back and forth:  >>> $('div').**toggleClass**('selected');  // the class is added here  >>> $('div').**toggleClass**('selected');  // the class is removed here  We can also specify more than one class to be toggled:  >>> $('div').**toggleClass**('important selected');  // both classes are added  >>> $('div').**toggleClass**('important selected');  // both classes are removed  With jQuery, we can **hide and show** HTML elements with the hide() and show() methods.  Make sure to scroll down the page to see the tree.  >>> $('#tree').**hide**();   // Hide the tree  >>> $('#tree').**show**();  //  Show the tree  With the **toggle**() method, hidden elements are shown and shown elements are hidden.  >>> $('#tree').hide();  // first hide the tree  Then toggle all img elements on the page:  >>> $('img').toggle();  The tree is shown, the car is hidden.  Toggle again:  >>> $('img').toggle();  Now the car is shown and the tree is hidden.  jQuery provides several methods that may be used to create animated effect even in older browser.  You can fade an element in and out of visibility.  Make sure you reload the page before you try the following:  >>> $('#tree').fadeOut("slow");  The tree slowly disappears.  You can also specify the time in milliseconds.  >>> $('#tree').fadeIn(5000);  The tree reappears.  You can use animate() to change CSS properties.  >>> $('#car').animate({left:'500px'});  **Event handling with jQuery**  jQuery provides simple event registration methods for the most common events.  We can specify a function name or an anonymous function expression as the event handler.  >>> $('#treebutton').click (function() { $('#tree').fadeToggle("slow");});  Now when you press on the tree button, the tree will fade in or out of visibility.  Not all events have their own method such as click.  However for all valid JavaScript events, **we can use the use the generic method 'on' and provide the JavaScript event name:**  >>> $('#carbutton').**on**("**click**",  function() { $('#car').fadeToggle("slow");});  And now whenever you press on the car button the car will also fade in and out of visibility.  In this case, using $(...).click(...) and $(...).on("click",...) are equivalent.  We can remove the event handler with the **off** method:  >>> $('#carbutton').**off**("**click**")  And now the car button is disabled.  Just like the addEventListener method, the event object is passed as an argument to the event handler:  >>> $('#carbutton').on("click",  function(**event**) { console.log(**event.target.id**)});  **The document ready event:**  When it’s important that the document object model be loaded and ready for our code to manipulate, we use the jQuery ready event as follows:  $(document).**ready**(function() {       // Include the code here  }) ;  You'll often see the above abbreviated as:  $(function() {       // Include the code here  }) ;  This notation means exactly the same thing as the one above. |  |