Gaurish Technologies Private Limited

JQuery

Module-6B

JQuery Documentation

What is jQuery?

JQuery is great library for developing ajax based application. Jquery is great library for the JavaScript programmers, which simplifies the development of web 2.0 applications.
JQuery helps programmer to keep code simple and concise. The JQuery library is designed to keep things very simple and reusable.

Why JQuery?

You can use simple JavaScript to perform all the functions that jQuery provides. Then why jQuery? The jQuery library is providing many easy to use functions and methods to make rich applications. These functions are very easy to learn and even a designer can learn it fast. Due to these features jQuery is very popular and in high demand among the developers. You can use jQuery in all the web based applications irrespective of the technology.

What are the features of JQuery?

JQuery have lot of functionalities but some of the key features are given below:

Selection of DOM elements:

The JQuery selector provides us capability to select DOM elements so that we can add functionality to them using methods of JQuery. It is using CSS 3.0 syntax which provides us freedom to select one or more elements. Using CSS, you can select element by id, class and collaborate with events to increase its functionality.

The wrapped set

The selected elements reside inside a object known as wrapped set. It contain all the selected DOM elements, it has array like structure. You can traverse through this like an array and can select elements using index.

Events

JQuery provide simplified event handling, you can easily bind and unbind events and for supported browsers it also provides a normalized event model due to this it is very easy to handle events. When any event occurs, it is called under the context of the event that triggered it.

Extensibility through plug-ins

The JQuery architecture provides us freedom to extend functionality using plug-ins. The plug-ins is easy to use and easy to clip with your page. You just need to set parameters to use this jQuery plug-ins and also need to include plug-in file. Some of the main jQuery plug-ins are:

1. XML and XSLT tools
2. Cookie handling
3. Datagrids
4. Drag and drop events.
5. Modal windows
6. Dynamic lists
7. Webservices

8. Ajax helpers
9. Even a JQuery-based Commodore 64 emulator.

Cross-browser support

In JavaScript, the DOM implementations for event handling vary considerably between browsers. Whereas JQuery providing a normalized event model for all supported browsers that makes it very easy to handle events.

Ajax support

AJAX stands for Asynchronous JavaScript and XML. Using AJAX we can connect to database and also can fetch the data from the server's database without refreshing the page. JQuery have very effective AJAX methods library to extend the functionality of AJAX.

Compatibility with languages

The jQuery script can be used with nearly all the web languages. Some of Frequently used languages with jQuery are given below:

1. PHP
2. JSP
3. ASP
4. Servlet
5. CGI

How to use JQuery?

Before we can start using JQuery's functions we need to include the source, that is the file containing the code which makes JQuery tick. You can do this in two ways. Firstly, you can download a copy from official site of JQuery and include it like so:

```
<script type="text/javascript" src="path/to/jquery.min.js">
```

If you do it this way, be sure to download the minified version, as it reduces the strain on your server.

The second and preferred way, is to include it from Google's Content Delivery Network, or CDN for short. There are two main benefits to doing it this way. Firstly, we can make sure we are always using the most recent version of the library, and secondly it means your server does not have to load the file and it takes up less bandwidth. To include it from the CDN we use similar code to above:

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1/jquery.min.js">
```

If you have loaded your jQuery from any CDN and it went down then your jQuery code will stop working. So to avoid this situation use the following code

```
<script type="text/javascript"
src=http://ajax.googleapis.com/ajax/libs/jquery/1.5.1/jquery.min.js ></script>
<script type="text/javascript">
        if (typeof jQuery == 'undefined')
        {
                document.write (unescape ("%3Cscript src='Scripts/jquery.1.5.1.min.js' type
        =' text/javascript'%3E%3C/script%3E"));
        }
</script>
```

It first loads the jQuery from Google CDN and then checks the jQuery object. If jQuery is not loaded successfully then it will references the jQuery.js file from hard drive location. In this example, the jQuery.js is loaded from Scripts folder.

How to write code for JQuery?

<script type="text/javascript"></script>
Inside these tags is where we will write our code.
The first thing you need to be up and running with JQuery is what's called the "document ready" handler. Pretty much everything you code in JQuery will be contained inside one of these. This accomplishes two things: First, it ensures that the code does not run until the DOM is ready. This confirms that any elements being accessed are actually in existence, so the script won't return any errors. Second, this ensures that your code is unobtrusive. That is, it's separated from content (XHTML) and presentation (CSS).
Here is what it looks like:

```
$(document).ready (function () {
        // all JQuery code goes here
});
```

To select an element with JQuery, we use the $ symbol followed by parenthesis: $(‘your selector goes here’). That selector can be any valid selector, including new CSS3 selectors. For example, to select all divs on my page, it’s as simple as:

$('div p') //selects all paragraphs that exist within a parent div.
$('#something') //selects the element with an id of 'something'
$('.something') //selects the element(s) with a class of 'something'

Why document. ready?

Traditionally developer's are using Window.onload () function to initiate some action on page load. There is one drawback with this function. It does not fires until all the images including the advertisement banner are loaded. So, window.onload () can be painfully slow. The JQuery provides the solution for this problem. The **$(document).ready (function () {})** solves the issue. It is fired once the Document Object Model is ready. So, you can use this to run any type of JavaScript to suite your business needs.
Here is the code that will display alert message once Document Object Model is ready:

```
$(document).ready (function (){

        // your code here...
        alert ("Hello");

 });
```

Hello world alert box example (HelloAlert.html)

```
<html>
        <head>
                <title>jQuery Hello World Alert box</title>
                <script type="text/javascript" src="jquery-1.4.2.js"></script>
         </head>
                <script type="text/javascript">
```

```
                $(document).ready(function(){
                        $("#cl").click(function(){
                                alert("HELLO WORLD!");
                        });
                });
        </script>
    <body>
        <font color="red">CLICK BELOW BUTTON TO SEE ALERT BOX
        </font>
        <br>
        <br>
        <button id="cl">Click Me</button>
    </body>
</html>
```

The following line includes the jQuery library into webpage:

```
<script type="text/javascript" src="jquery-1.4.2.js"></script>
```

****

Assignment1:

How to write JQuery function body:

```
(function ($){
    $.fn.extend({
        Jquery_function_name:
        function ()
         {
            // do something when slide down is finished
        }
    });
}) (jQuery);
```

Syntax of JQuery Function calling:

```
$(). Jquery_function_name();
```

Ex.

```
<html>
    <head>
        <script type="text/javascript"
    src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script
    >
        <script>
            (function ($){
                $.fn.extend({
                    display:
                    function (name)
                    {
```

```
                                        $("#div1").html("Good morning
"+name+"!");
                                }
                        });
                }) (jQuery);

                $(document).ready (function ()
                {
                        $("#b1").click (function ()
                        {
                                var name=$("#name").val();
                                $().display (name);
                        });
                });

        </script>
</head>
<body>
        <h1>JQuery Function Calling</h1>
        <label> Write your name </label> <input type="text"
id="name" size="20">
        <input type="button" id="b1" value="click" >
        <div id="div1" style="padding:20px;"></div>
</body>
</html>
```

Selecting elements in JQuery:

The JQuery library allows you to select elements in your XHTML by wrapping them in $("") (you could also use single quotes), which is the JQuery wrapper. Here are some examples of "wrapped sets" in JQuery:

```
$("div"); // selects all HTML div elements
$("#myElement"); // selects one HTML element with ID "myElement"
$(".myClass"); // selects HTML elements with class "myClass"
$("p#myElement"); // selects paragraph elements with ID "myElement"
$("ul li a.navigation"); // selects anchors with class "navigation" that are nested in list items
```

JQuery supports the use of all CSS selectors, even those in CSS3. Here are some examples of alternate selectors:

```
$("p > a"); // selects anchors that are direct children of paragraphs
$("input [type=text]"); // selects inputs that have specified type
$("a: first"); // selects the first anchor on the page
$("p: odd"); // selects all odd numbered paragraphs
$("li:first-child"); // every list item that's first child in a list
```

JQuery also allows the use of its own custom selectors. Here are some examples:

```
$(": animated"); // selects elements currently being animated
$(": button"); // selects any button elements (inputs or buttons)
$(": radio"); // selects radio buttons
```

```
$(": checkbox"); // selects checkboxes
$(": checked"); // selects selected checkboxes or radio buttons
$(": header"); // selects header elements (h1, h2, h3, etc.)
```

Manipulating and Accessing CSS Class Names:

JQuery allows you to easily add, remove, and toggle CSS classes, which come in handy for a variety of practical uses. Here are the different syntaxes for accomplishing this:

```
$("div").addClass("content"); // adds class "content" to all <div> elements
$("div").removeClass("content"); // removes class "content" from all <div> elements
$("div").toggleClass("content");
// toggles the class "content" on all <div> elements (adds it if it doesn't exist,
// and removes it if it does)
```

You can also check to see if a selected element has a particular CSS class, and then run some code if it does. Youn would check this using an if statement. Here is an example:

```
if ($("#myElement").hasClass("content")) {
// do something here
}
```

You could also check a set of elements (instead of just one), and the result would return "true" if any one of the elements contained the class.

Manipulating CSS Styles with JQuery:

CSS styles can be added to elements easily using JQuery, and it's done in a cross-browser fashion. Here are some examples to demonstrate this:

```
$("p").css("width", "400px"); // adds a width to all paragraphs
$("#myElement").css("color", "blue") // makes text color blue on element
#myElement
$("ul").css("border", "solid 1px #ccc") // adds a border to all lists
```

Adding, Removing, and Appending Elements and Content There are a number of ways to manipulate groups of elements with JQuery, including manipulating the content of those elements (whether text, inline elements, etc).

Get the HTML of any element (similar to innerHTML in JavaScript):

```
var myElementHTML = $("#myElement").html();
// variable contains all HTML (including text) inside #myElement
```

If you don't want to access the HTML, but only want the text of an element:

```
var myElementHTML = $("#myElement").text();
// variable contains all text (excluding HTML) inside #myElement
```

Using similar syntax to the above two examples, you can change the HTML or text content of a specified element:

```
$("#myElement").html ("<p>This is the new content. </p>");
// content inside #myElement will be replaced with that specified
$("#myElement").text ("This is the new content.");
// text content will be replaced with that specified
```

To append content to an element:

```
$("#myElement").append ("<p>This is the new content. </p>");
// keeps content intact, and adds the new content to the end
$("p").append ("<p>This is the new content. </p>");
// add the same content to all paragraphs
```

JQuery also offers use of the commands appendTo(), prepend(), prependTo(), before(),

insertBefore (), after(), insertAfter(), which work similarly to append but with their own unique characteristics.
JQuery Specific event handlers can be established using the following code:

```
$("a").click (function () {
        // do something here
        // when any anchor is clicked
});
```

The code inside function () will only run when an anchor is clicked. Some other common events you might use in JQuery include: blur, focus, hover, keydown, load, mousemove, resize, scroll, submit, select.

Here some more frequently used events:

blur() : binds a function when focus from the element is out(mostly used with text boxes)

```
Ex:   $("#test").blur(function(){
      });
```

change (): binds a function when value of selected elements is changed(mostly used with dropdown)

```
Ex:   $("#test").change(function(){
      });
```

click(): binds a function when you click on the selected elements (mostly used with buttons)

```
Ex:   $("#test").click (function () {
      });
```

focus () : binds a function when you focus an elements

```
Ex:   $("#test").focus (function(){
      });
```

keydown () : binds a function when you press a key to write in selected elements.

```
Ex:   $("#test").keydown (function (){
      });
```

keypress(): binds a function when any key is press in selected elements.

```
Ex:   $("#test").keypress (function () {
      });
```

keyup():binds a function whe you write something in selected elements.

```
Ex:   $("#test").keyup (function () {
      });
```

mouseover(): binds a function when you move your mouse over selected elements.

```
Ex:   $("#test"). mouseover(function(){

      });
```

Showing and Hiding Elements with jQuery:

The syntax for showing, hiding an element (or toggling show/hide) is:

```
$("#myElement").hide("slow", function() {
        // do something once the element is hidden
}
$("#myElement").show("fast", function() {
```

```
        // do something once the element is shown
    }
    $("#myElement").toggle (1000, function() {
        // do something once the element is shown/hidden
    }
```

Remember that the "toggle" command will change whatever state the element currently has, and the parameters are both optional. The first parameter indicates the speed of the showing/hiding. If no speed is set, it will occur instantly, with no animation. A number for "speed" represents the speed in milliseconds. The second parameter is an optional function that will run when the command is finished executing. You can also specifically choose to fade an element in or out, which is always done by animation:

```
    $("#myElement").fadeOut ("slow", function () {
        // do something when fade out finished
    }
    $("#myElement").fadeIn ("fast", function () {
        // do something when fade in finished
    }

    To fade an element only partially, either in or out:
    $("#myElement").fadeTo (2000, 0.4, function () {
        // do something when fade is finished
    }
```

The second parameter (0.4) represents "opacity", and is similar to the way that opacity is set in CSS. Whatever the opacity is to start with, it will animate (fadeTo) until it reaches the setting specified, at the speed set (2000 milliseconds).
The optional function (called a "callback function") will run when the opacity change is complete. This is the way virtually all callback functions in jQuery work.

JQuery Animations and Effects:

You can slide elements, animate elements, and even stop animations in mid-sequence. To slide elements up or down:

```
    $("#myElement").slideDown (fast, function () {
    // do something when slide down is finished
    }
    $("#myElement").slideUp (slow, function () {
    // do something when slide up is finished
    }
    $("#myElement").slideToggle (1000, function () {
    // do something when slide up/down is finished
    }
```

To animate an element, you do so by telling JQuery the CSS styles that the item should change to. JQuery will set the new styles, but instead of setting them instantly (as CSS or raw JavaScript would do), it does so gradually, animating the effect at the chosen speed:

```
    $("#myElement").animate (
    {
        opacity: .3,
        width: "500px",
        height: "700px"
    }, 2000, function () {
```

```
        // optional callback after animation completes
    });
```

Animation with jQuery is very powerful, and it does have its quirks (for example, to animate colors, you need a special plugin).

Assignment2:

What is ajax methods?

AJAX is the art of exchanging data with a server, and update parts of a web page - without reloading the whole page.

There a number of ajax functions mostfrequently used:

$.ajax(): This method is mostly used for requests where the other methods cannot be used. Mostly we use it to call a function on js page. Parameters which are passed from ajax function:

Data:	Specifies data to be sent to the server
dataType:	The data type expected of the server response.
timeout:	The local timeout (in milliseconds) for the request
type:	Specifies the type of request. (GET or POST)
url:	Specifies the URL to send the request to. Default is the current page
error ():	A function to run if the request fails.
success ():	A function to be run when the request succeeds
complete:	A function to run when the request is finished (after success and error unctions).

Syntax: $.ajax({*name:value, name:value, ... }*)

Ex:

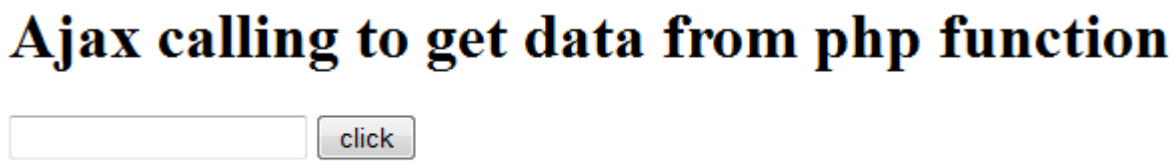
Here example of ajax function calling using simple php files and codeigniter is displayed

a) Here is the example for calling function using simple php files.

1) Create an simple php page named ‘call.php’

```
<html>
  <head>
        <script type="text/javascript"
        src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script
        >
        <script>
                $(document).ready(function()
                {
                        $("#b1").click(function()
                        {
                                var txt=$("#txt").val();
                                var dataString={'data':txt};
                                $.ajax({
```

```
                                    url:'http://localhost/demo_function_call/
            comman_function.php?action=factorial',
                                    type: 'POST',
                                    cache:true,
                                    dataType: 'html',
                                    data: dataString,
                                    timeout:120000,
                                    error: function(){
                                            alert('something went wrong while trying
            to verify email-id');
                                    },
                                    success: function(str){
                                            $("#div1").html("<br/>Fatorial of
            "+txt+" is <br/><br/>"+str);
                                    }
                            });
                    });
            });
        </script>
    </head>
    <body>
        <h1> Ajax calling to get data from php function </h1>
        <input type="text" id="txt" size="20">
        <input type="button" id="b1" value="click" >
        <div id="div1"></div>
    </body>
</html>
```



Now create a php page for php functions named ‘comman_function.php’

```
<?php

if (isset($_GET[action])){
// Retrieve the GET parameters and executes the function
  $funcName    = $_GET[action];

  $funcName();
 } else if (isset($_POST[action])){
  // Retrieve the POST parameters and executes the function
  $funcName    = $_POST[action];

$funcName($vars);
```

```
        } else {
                  // If there is no action in the URL, then do this
                 echo "<INPUT NAME='btnSubmitAdmin' TYPE='button'
ONCLICK='javascript:javaFunction () ' VALUE='Call Javafunction () which redirects to a
PHP function'>";
        }
         function factorial(){
        $txt=$_POST ['data'];
        $msg="";
        $facto=1;

        for ($i=$txt; $i>=1; $i--)
        {
                $facto=$i*$facto;


                if($i==1)
                {
                        $msg .=$i ." = ".$facto;
                } else
                {
                        $msg .=$i ." X ";
                }
        }
        echo $msg;

}
?>
```


Ajax calling to get data from php function

5 click

Fatorial of 5 is

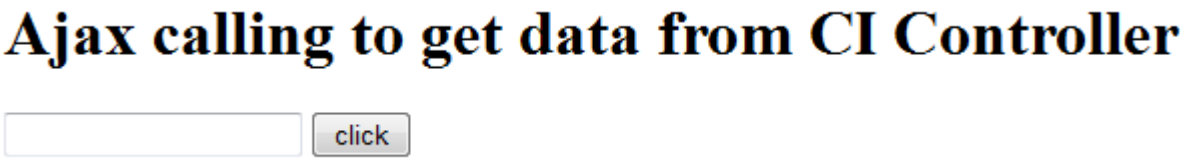
5 X 4 X 3 X 2 X 1 = 120

b) Here is the example for calling function using Code Igniter.

1) Create an view file named ‘demo_view.php’

```
<html>
<head>
```

```
<script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/1.4.2/jquery.min.js"></script>
<script>
$(document).ready(function()
{
    $("#b1").click(function()
    {
            var txt=$("#txt").val();
            var dataString={'data':txt};
            $.ajax({
                    url:'http://localhost/codeigniter/demo_controller/factorial',
                    type: 'POST',
                    cache:true,
                    dataType: 'html',
                    data: dataString,
                    timeout:120000,
                    error: function(){
                            alert('something went wrong while trying to verify email-id');
                    },
                    success: function(str){
                            $("#div1").html("<br/>Fatorial of "+txt+" is <br/><br/>"+str);
                    }
            });
    });
});
</script>
</head>
<body>
<h1> Ajax calling to get data from php function </h1>
<input type="text" id="txt" size="20">
<input type="button" id="b1" value="click" >
<div id="div1"></div>
</body>
</html>
```



Now create a controller named ‘demo_controller.php’

```
<?php
class Demo_controller extends CI_Controller  {

        public function __construct()
        {
                parent::__construct();
        }
```

```
        public function index()
        {
                $this->load->view('demo_view');
        }
        public function factorial()
        {
                $txt=$_POST ['data'];
                $msg="";
                $facto=1;

                for ($i=$txt; $i>=1; $i--)
                {
                        $facto=$i*$facto;


                        if($i==1)
                        {
                                $msg .=$i ." = ".$facto;
                        } else
                        {
                                $msg .=$i ." X ";
                        }
                }
                echo $msg;
        }
}

?>
```

Ajax calling to get data from CI Controller

6 click

Fatorial of 6 is

6 X 5 X 4 X 3 X 2 X 1 = 720

ajaxComplete():The ajaxComplete() method specifies a function to be run when an AJAX request completes.
Unlike ajaxSuccess(), functions specified with the ajaxComplete() method will run when the request is completed, even it is not successful.
Syntax:

$(*selector*).ajaxComplete(*function(event,xhr,options)*)

EX:

```
$("#txt").ajaxStart(function(){
  $("#wait").css("display","block");
});
```

```
$("#txt").ajaxComplete(function(){
  $("#wait").css("display","none");
});
```

$.get(): The get() method is used to perform an AJAX HTTP GET request.

Syntax: $(*selector*).get(*url,data,success(response,status,xhr),dataType)*

Ex:

```
$("button").click(function(){
  $.get("demo_ajax_load.txt", function(result){
    $("div").html(result);
  });
});
```

$.post(): The post() method is used to perform an AJAX HTTP POST request.

Syntax: $(*selector*).post (*url, data, success (response, status, xhr), dataType)*

Ex:

```
$("input").keyup(function(){
  txt=$("input").val();
  $.post("demo_ajax_gethint.asp",{suggest:txt},function(result){
    $("span").html(result);
  });
});
```

serialize():The serialize() method creates a URL encoded text string by serializing form values. You can select one or more form elements (like input and/or text area), or the form element itself.The serialized values can be used in the URL query string when making an AJAX request.

Syntax: $(*selector*).serialize()

Ex:

```
$("button").click(function(){
        $("div").text($("form").serialize());
});
```

Assignment3: