JQuery

Created By :- Ankit M Patel

The EasyLearn Academy

Introduction

- **jQuery is a clientside JavaScript Library.**
- jQuery greatly simplifies JavaScript programming.
- jQuery is easy to learn.
- The purpose of jQuery is to make JavaScript much easier to use on your website.
- jQuery is a lightweight, "**write less, do more**", JavaScript library.
- jQuery can perform lots of common tasks that require many lines of JavaScript code to do the same
- Jquery wraps them into library methods that you can call with a single line of code.
- jQuery also simplifies a lot of the complicated things of JavaScript, like AJAX calls and DOM manipulation.

Technologies used in JQuery

- HTML
- CSS
- JavaScript

Features/application of JQuery

- HTML/DOM manipulation
- CSS manipulation
- HTML event methods
- Effects and animations
- AJAX
- Utilities

Who use Jquery?

- Many of the biggest companies on the Web use jQuery, such as:
- Google
- Microsoft
- **IBM**
- Netflix
- **amazon**

Browser compatibility

- The jQuery team knows all about cross-browser issues, and they have written this knowledge into the jQuery library.
- jQuery will run exactly the same in all major browsers, including Internet Explorer 6!

How to use Jquery in website?

- There are several ways to start using jQuery on your web site. You can:
  - Download the jQuery library from jQuery.com
  - Include jQuery from a CDN (Content Delivery Network), like Google
    <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">

Downloading jQuery

- here are two versions of jQuery available for downloading:
  - Production version - this is for your live website because it has been minified and compressed
  - Development version - this is for testing and development (uncompressed and readable code)
- Both versions can be downloaded from jQuery.com.

jQuery Syntax

- **With jQuery you select (query) HTML elements and perform "actions" on them.**
- **jQuery Syntax**
- The jQuery syntax is tailor made for **selecting** HTML elements and performing some **action** on the element(s).
- Basic syntax is: **$(*selector*).*action*()**
  - A $ sign to define/access jQuery
  - A (*selector*) to "query (or find)" HTML elements
  - A jQuery *action*() to be performed on the element(s)
- Examples:
  - $(this).hide() - hides the current element.
  - $("**p**").hide() - hides all <p> elements.
  - $("**.test**").hide() - hides all elements with class="test".
  - $("**#test**").hide() - hides the element with id="test".

Simple example

- <!DOCTYPE html>
- <html>
  - <head>
  - </head>
  - <body>
  - If you click on me, I will disappear.
  - Click me away!
  - Click me too!
  - <script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
  - </script>
  - <script>
  - $(document).ready(function(){
  - $("p").click(function(){
  - $(this).hide();
  - });
  - });
  - </script>
  - </body>
- </html>

The Document Ready Event

- You might have noticed that all jQuery methods in our examples, are inside a document ready event:
- $(document).ready(function(){

  // jQuery methods go here…

  });
- **This is to prevent any jQuery code from running before the document is finished loading (is ready).**
- It is good practice to wait for the document to be fully loaded and ready before working with it.
- Here are some examples of actions that can fail if methods are run before the document is fully loaded:
  - Trying to hide an element that is not created yet
  - Trying to get the size of an image that is not loaded yet

jQuery Selectors

- Query selectors allow you to select and manipulate HTML element(s).
- jQuery selectors are used to "find" (or select) HTML elements based on their id, classes, types, attributes, values of attributes and much more.
- It's based on the existing CSS Selectors, and in addition, it has some own custom selectors.
- **All selectors in jQuery start with the dollar sign and parentheses: $().**

The element Selector

- The jQuery element selector selects elements based on the element name.
- You can select all <p> elements on a page like this:
- $("p")
- **Example**
- When a user clicks on a button, all <p> elements will be hidden:
- **Example**
- **$(document).ready(function(){**
  $(“:button").click(function(){
  $("p").hide();
  });
  });

The #id Selector

- The jQuery #id selector uses the id attribute of an HTML tag to find the specific element.
- An id should be unique within a page, so you should use the #id selector when you want to find a **single, unique element**.
- To find an element with a specific id, write a hash character, followed by the id of the element:
- $("#test")
- **Example**
- When a user clicks on a button, the element with id="test" will be hidden:
- **Example**
- **$(document).ready(function(){**
  $(“#test").click(function(){
  $(this).hide();
  });
  });

The .class Selector

- The jQuery class selector finds elements with a specific class.
- To find elements with a specific class, write a period character, followed by the name of the class:
- $(".test")
- **Example**
- When a user clicks on a button, the elements with class="test" will be hidden:
- **Example**
- ```
  $(document).ready(function(){
    $("button").click(function(){
      $(".test").hide();
    });
  });
  ```
  ```

jQuery Selectors

Syntax	Description
$("*")	Selects all elements
$(this)	Selects the current HTML element
$("p.intro")	Selects all <p> elements with class="intro"
$("p:first")	Selects the first <p> element
$("ul li:first")	Selects the first element of the first
$("ul li:first-child")	Selects the first element of every
$("[href]")	Selects all elements with an href attribute
$("a[target='_blank']")	Selects all <a> elements with a target attribute value equal to "_blank"
$("a[target!='_blank']")	Selects all <a> elements with a target attribute value NOT equal to "_blank"
$(":button")	Selects all <button> elements and <input> elements of type="button"
$("tr:even")	Selects all even <tr> elements
$("tr:odd")	Selects all odd <tr> elements

What are Events?

- jQuery is tailor-made to respond to events in an HTML page.
- All the different visitor's actions that a web page can respond to are called events.
- An event represents the precise moment when something happens.
- Examples:
  - moving a mouse over an element
  - selecting a radio button
  - clicking on an element
- The term "**fires**" is often used with events. Example: "The keypress event fires the moment you press a key".

Here are some common DOM events

Mouse Events	Keyboard Events	Form Events	Document/Window Events
click	keypress	submit	load
dblclick	keydown	change	resize
mouseenter	keyup	focus	scroll
mouseleave		blur	unload

```
<!DOCTYPE html>
<html>
<head>
</head>
<body>
Name: <input type="text" name="fullname"><br>
Email: <input type="text" name="email">
<script src="//ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js">
</script>
<script>
$(document).ready(function(){
  $("input").focus(function(){
    $(this).css("background-color","#cccccc");
  });
  $("input").blur(function(){
    $(this).css("background-color","#ffffff");
  });
});
</script>
</body>
</html>
```

jQuery hide() and show()

- With jQuery, you can hide and show HTML elements with the hide() and show() methods:
- **Example**
- ```
  $("#hide").click(function(){
    $("p").hide();
  });

  $("#show").click(function(){
    $("p").show();
  });
  ```
- $(*selector*).hide(*speed,callback*);

  $(*selector*).show(*speed,callback*);
- The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the hide() or show() method completes
  ```

jQuery toggle()

- With jQuery, you can toggle between the hide() and show() methods with the toggle() method.
- Shown elements are hidden and hidden elements are shown:
- **Example**
- $("button").click(function(){
  $("p").toggle();
  });

jQuery Callback Functions

- JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.
- To prevent this, you can create a callback function.
- A callback function is executed after the current effect is finished.
- Typical syntax: **$(*selector*).hide(*speed,callback*);**
- **Examples**
- The example below has a callback parameter that is a function that will be executed after the hide effect is completed:
- **Example with Callback**
- ```
  $("button").click(function(){
    $("p").hide("slow",function(){
      alert("The paragraph is now hidden");
    });
  });
  ```
  ```

jQuery Effects - Fading

- With jQuery you can fade elements in and out of visibility.
- **jQuery Fading Methods**
- With jQuery you can fade an element in and out of visibility.
- jQuery has the following fade methods:
  1. fadeIn()
  2. fadeOut()
  3. fadeToggle()
  4. fadeTo()

jQuery fadeIn() Method

- **The jQuery fadeIn() method is used to fade in(to show) a hidden element**.
- **Syntax:**
- $(*selector*).fadeIn(*speed,callback*);
- The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the fading completes.
- The following example demonstrates the fadeIn() method with different parameters:
- **Example**

```
$("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
});
```

jQuery fadeOut() Method

- The jQuery fadeOut() method is used to fade out a visible element.
- **Syntax:**
- $(*selector*).fadeOut(*speed,callback*);
- The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the fading completes.
- The following example demonstrates the fadeOut() method with different parameters:
- **Example**
- ```
  $("button").click(function(){
    $("#div1").fadeOut();
    $("#div2").fadeOut("slow");
    $("#div3").fadeOut(3000);
  });
  ```
  ```

Query fadeToggle() Method

- The jQuery fadeToggle() method toggles between the fadeIn() and fadeOut() methods.
- If the elements are faded out, fadeToggle() will fade them in.
- If the elements are faded in, fadeToggle() will fade them out.
- **Syntax:**
- $(*selector*).fadeToggle(*speed,callback*);
- The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the fading completes.
- The following example demonstrates the fadeToggle() method with different parameters:
- **Example**
- ```
  $("button").click(function(){
    $("#div1").fadeToggle();
    $("#div2").fadeToggle("slow");
    $("#div3").fadeToggle(3000);
  });
  ```
  ```

Query fadeTo() Method

- The jQuery fadeTo() method allows fading to a given opacity (value between 0 and 1).
- **Syntax:**
- $(*selector*).fadeTo(*speed,opacity,callback*);
- The required speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The required opacity parameter in the fadeTo() method specifies fading to a given opacity (value between 0 and 1).
- The optional callback parameter is a function to be executed after the function completes.
- The following example demonstrates the fadeTo() method with different parameters:
- **Example**
- ```
  $("button").click(function(){
    $("#div1").fadeTo("slow",0.15);
    $("#div2").fadeTo("slow",0.4);
    $("#div3").fadeTo("slow",0.7);
  });
  ```
  ```

jQuery Sliding Methods

- With jQuery you can create a sliding effect on elements.
- jQuery has the following slide methods:
  - slideDown()
  - slideUp()
  - slideToggle()

jQuery slideDown() Method

- The jQuery slideDown() method is used to slide down an element.
- **Syntax:**
- $(*selector*).slideDown(*speed,callback*);
- The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the sliding completes.
- The following example demonstrates the slideDown() method:
- **Example**

```
$("#flip").click(function(){
    $("#panel").slideDown();
  });
```

jQuery slideUp() Method

- The jQuery slideUp() method is used to slide up an element.
- **Syntax:**
- $(*selector*).slideUp(*speed,callback*);
- The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the sliding completes.
- The following example demonstrates the slideUp() method:
- **Example**

```
$("#flip").click(function(){
    $("#panel").slideUp();
});
```

jQuery slideToggle() Method

- The jQuery slideToggle() method toggles between the slideDown() and slideUp() methods.
- If the elements have been slid down, slideToggle() will slide them up.
- If the elements have been slid up, slideToggle() will slide them down.
- $(*selector*).slideToggle(*speed,callback*);
- The optional speed parameter can take the following values: "slow", "fast", milliseconds.
- The optional callback parameter is a function to be executed after the sliding completes.
- The following example demonstrates the slideToggle() method:
- **Example**

```
$("#flip").click(function(){
  $("#panel").slideToggle();
});
```

jQuery Animations - The animate() Method

- The jQuery animate() method is used to create custom animations.
- Syntax:
- **$(*selector*).animate({*params*},*speed, callback*);**
- The required params parameter defines the CSS properties to be animated.
- The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the animation completes.
- The following example demonstrates a simple use of the animate() method; it moves a element to the left, until it has reached a left property of 250px:
- **To change x y position of any element, one must set its position to absolute using css.**

Example

```
$("button").click(function(){
  $("div").animate({left:'250px'});
});
```

jQuery Method Chaining

- Until now we have been writing jQuery statements one at a time (one after the other).
- **However, there is a technique called chaining, that allows us to run multiple jQuery commands, one after the other, on the same element(s).**
- Biggest advantage of this techniue is that browsers do not have to find the same element(s) more than once.
- To chain an action, you simply append the action to the previous action.
- The following example chains together the css(), slideUp(), and slideDown() methods.
- The "p1" element first changes to red, then it slides up, and then it slides down:
- **Example**
- $("#p1").css("color","red").slideUp(2000).slideDown(2000);

jQuery stop() Method

- The jQuery stop() method is used to stop an animation or effect before it is finished.
- **The stop() method works for all jQuery effect functions, including sliding, fading and custom animations.**
- **Syntax:**
- $(*selector*).stop(*stopAll,goToEnd*);
- The optional stopAll parameter specifies whether also the animation queue should be cleared or not. Default is false, which means that only the active animation will be stopped, allowing any queued animations to be performed afterwards.
- The optional goToEnd parameter specifies whether or not to complete the current animation immediately. Default is false.
- So, by default, the stop() method kills the current animation being performed on the selected element.
- The following example demonstrates the stop() method, with no parameters:
- Example
- ```
  $("#stop").click(function(){
    $("#panel").stop();
  });
  ```
  ```

jQuery DOM Manipulation

jQuery DOM Manipulation

- One very important part of jQuery is the capability to manipulate the DOM.
- jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes.
- Using this method, one can get only text or text with html from controls like textbox, checkbox or html tag like div or paragraph without right much code.
- Three simple, but useful, jQuery methods for DOM manipulation are:
  - **text()** - Sets or returns the text content of selected elements
  - **html()** - Sets or returns the content of selected elements (including HTML markup)
  - **val()** - Sets or returns the value of form fields like textbox, checkbox etc

So now let us see some example

attr()/prop()

- The jQuery attr()/prop() method is used to get or set attribute values of any html tag.
- **To retrieve and change DOM properties such as the checked, selected, or disabled state of form elements, use the .prop() method.**
- For example you can have a web-page with images on which if user moves the mouse, it size will be increased by 15% and then it will be decreased by 15% when mouse lost its focus from image. This can be done using attr() method
- The following example demonstrates how to get the value of the href attribute in a link and how to set the value for the same:
- **Example**

```
$("button").click(function(){
alert($("#w3s").attr("href"));
$("#w3s").attr("href","sample.php")
});
```

How to add new HTML content using JQuery?

- With jQuery, it is easy to add new elements/content.
- There are four jQuery methods that are used to add new content:
  1. **append()** - Inserts content at the end of the selected elements
  2. **prepend()** - Inserts content at the beginning of the selected elements
  3. **after()** - Inserts content after the selected elements
  4. **before()** - Inserts content before the selected elements

Remove Elements/Content

- To remove elements and content, there are mainly two jQuery methods:
  - **remove()** - Removes the selected element (and its child elements)
  - **empty()** - Removes the child elements from the selected element

How to manipulate css using JQuery?

- We can manipulate css using Jquery very easily.
- Jquery Provides serveral method to perform this task.
- Let us see these methods in detail.
  1. addClass() - Adds one or more classes to the selected elements
  2. removeClass() - Removes one or more classes from the selected elements
  3. toggleClass() - Toggles between adding/removing classes from the selected elements
  4. css() - Sets or returns the style attribute
- One can give multiple selected name in $() to select multiple html tag as well as can give multiple class name sepeated by space in above methods.

jQuery css() Method

- the css() method sets or returns one or more style properties for the selected elements.
- **How Return a CSS Property**
- To return the value of a specified CSS property, use the following syntax:
- css("*propertyname*");
- The following example will return the background-color value of the FIRST matched element:
- **Example**
- $("p").css("background-color");

Set a CSS Property

- To set a specified CSS property, use the following **syntax**:
- css("*propertyname*","*value*");
- The following example will set the background-color value for ALL matched elements:
- **Example**
- $("p").css("background-color“,"yellow");

Set Multiple CSS Properties

- To set multiple CSS properties, use the following syntax:
- css({"*propertyname*":"*value*“,"*propertyname*":"*value*",...});
- The following example will set a background-color and a font-size for ALL matched elements:
- **Example**
- $("p").css({"background-color":"yellow","font-size":"200%"});

What is AJAX?

- AJAX is **Asynchronous JavaScript** and X.M.L. means extensible markup language.
- Nowdays X.M.L is replaced by J.S.O.N. means **javascript object notation**.
- AJAX is used to load data in the background and display it on the webpage, without reloading the whole page.
- Examples of applications using AJAX: Gmail, Google Maps, Youtube, and Facebook tabs.

How to use Ajax with Jquery?

- jQuery provides several methods for AJAX functionality.
- With the jQuery AJAX methods, you can request text, HTML, XML, or JSON from a remote server using both HTTP Get and HTTP Post .
- you can load the external data directly into the selected HTML elements of your web page!

jQuery load() Method…

- The jQuery load() method is a simple, but powerful AJAX method.
- The load() method loads data from a server and puts the returned data into the selected element.
- **Syntax:**
- $(*selector*).load(*URL,data,callback*);
- The required URL parameter specifies the URL you wish to load.
- The optional data parameter specifies a set of querystring key/value pairs to send along with the request.
- The optional callback parameter is the name of a function to be executed after the load() method is completed.
- **Here is the content of our example file: "demo_test.txt":**
- <h2>jQuery and AJAX is FUN!!!</h2> This is some text in a paragraph.
- The following example loads the content of the file "demo_test.txt" into a specific element:
- **Example**
- $("#p1").load("demo_test.txt");

jQuery load() Method

- The optional callback parameter specifies a callback function to run when the load() method is completed. The callback function can have different parameters:
  - responseTxt - contains the resulting content if the call succeed
  - statusTxt - contains the status of the call
  - xhr - contains the XMLHttpRequest object

example

- $(document).ready(function(){
- $(":button.buttonone").click(function(){
- $("#div1").load("test.php");
- });
- });
- To run example of load() method, file must be saved in www directory of wamp server.

example

- $(document).ready(function(){
- $(":button.buttonone").click(function(){
- $("#div1").load("test.php");
- });
- });
- To run example of load() method, file must be saved in www directory of wamp server.

jQuery $.get() Method

- The $.get() method requests data from the server with an HTTP GET request.
- **Syntax:**
- $.get(*URL,callback*);
- The required URL parameter specifies the URL you wish to request.
- The optional callback parameter is the name of a function to be executed if the request succeeds.
- The following example uses the $.get() method to retrieve data from a file on the server:
- **Example**
- ```
  $("button").click(function(){
    $.get("test.php",function(data,status){
      alert("Data: " + data + "\nStatus: " + status);
    });
  });
  ```
- Data contains the data return as response from the server and status contains the response return via server. It may success or failed.
  ```

Query $.post() Method

- The $.post() method requests data from the server using an HTTP POST request.
- **Syntax:**
- $.post(*URL,data,callback*);
- The required URL parameter specifies the URL you wish to request.
- The optional data parameter specifies some data to send along with the request.
- The optional callback parameter is the name of a function to be executed if the request succeeds.
- The following example uses the $.post() method to send some data along with the request:
- **Example**

```
$("button").click(function(){
  $.post("demo_test_post.php",
  {
    name:“The EasyLearn",
    city:”Bhavnagar"
  },
  function(data,status){
    alert("Data: " + data + "\nStatus: "
+ status);
  });
});
```

- **It is not suitable to submit form data as shown in above example in real situation. So one should use serialize method as 2nd argument in post method.**

serialize() ….

- It is used to encode a set of form elements as a string for submission.
- The .serialize() method creates a text string in standard URL-encoded notation.
- It can act on a jQuery object that has selected individual form controls, such as <input>, <textarea>, and <select>:
- For exmaple one can given below code to serialize input textarea and select tag in form.
- <form id=“myform”>
- </form>
- $(“#myform").serialize();

Serialize() example

```
$(document).ready(function(){
  $(":button.b1").click(function(){
        $.post("test_post2.php",$("#myform").serialize(),
    function(data,status){
      alert("Data: " + data + "\nStatus: " + status);
         $("#div1").html(data);
    }).done(function() { alert("second success"); })
.fail(function() { alert("error"); })
.always(function() { alert("finally finished"); });
  });
});
```