jQuery

jQuery is a lightweight, "write less, do more", JavaScript library.

The purpose of jQuery is to make it much easier to use JavaScript on your website. jQuery takes a lot of common tasks that require many lines of JavaScript code to accomplish, and wraps them into methods that you can call with a single line of code.

jQuery also simplifies a lot of the complicated things from JavaScript, like AJAX calls and DOM manipulation.

The jQuery library contains the following features:

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

Why jQuery?

There are lots of other JavaScript libraries out there, but jQuery is probably the most popular, and also the most extendable. Many of the biggest companies on the Web use jQuery, such as: Google, Microsoft, IBM, Netflix.

Will jQuery work in all browsers?

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.

Adding jQuery to Your Web Pages

There are several ways to start using jQuery on your web site:

- Download the jQuery library from jQuery.com
- Include jQuery from a CDN (Content Delivery Network), like Google

Google CDN: notice that the <script> tag should be inside the <head> section).

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
</head>
```

If downloaded: The jQuery library is a single JavaScript file, and you reference it with the HTML <script> tag (Place the downloaded file in the same directory as the pages where you wish to use it.)

```
<head>
<script src="jquery-3.4.1.min.js"></script>
</head>
```

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".

The Document Ready Event

JQuery methods in 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. This also allows you to have your JavaScript code before the body of your document, in the head section.

jQuery Selectors

jQuery selectors allow you to select and manipulate HTML element(s).

jQuery selectors are used to "find" (or select) HTML elements based on their name, id, classes, types, attributes, values of attributes and much more.

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:

```
$(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 HTML element: $("#test")

Example: When a user clicks on a button, the element with id="test" will be hidden:

```
$(document).ready(function(){
  $("button").click(function(){
    $("#test").hide();
  });
});
```

More Examples of jQuery Selectors: (https://www.w3schools.com/jquery/jquery_selectors.asp)

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

Sometimes it is preferable to place jQuery functions in a separate file and use the src attribute to refer to the .js file as below:

```
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"></script>
<script src="my_jquery_functions.js"></script>
</head>
```

jQuery Event Methods

(https://www.w3schools.com/jquery/jquery_events.asp)

An event represents the precise moment when something happens.

Examples:

- moving a mouse over an element
- selecting a radio button
- clicking on an element

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

jQuery Syntax For Event Methods:

In jQuery, most DOM events have an equivalent jQuery method.

To assign a click event to all paragraphs on a page, you can do this:

```
$("p").click();
```

The next step is to define what should happen when the event fires. You must pass a function to the event:

```
$("p").click(function(){  // action goes here!!  });
```

Commonly Used jQuery Event Methods

- **$(document).ready():** The `$(document).ready()` method allows us to execute a function when the document is fully loaded.

- **click():** The `click()` method attaches an event handler function to an HTML element. The function is executed when the user clicks on the HTML element.
  The following example says: When a click event fires on a <p> element; hide the current <p> element:

```
$("p").click(function(){
  $(this).hide();
});
```

- **dblclick():** The `dblclick()` method attaches an event handler function to an HTML element. The function is executed when the user double-clicks on the HTML element:

```
$("p").dblclick(function(){
  $(this).hide();
});
```

- **mouseenter():** The `mouseenter()` method attaches an event handler function to an HTML element.
  The function is executed when the mouse pointer enters the HTML element:

```
$("#p1").mouseenter(function(){
  alert("You entered p1!");
});
```

- **mouseleave():** The `mouseleave()` method attaches an event handler function to an HTML element.
  The function is executed when the mouse pointer leaves the HTML element:

```
$("#p1").mouseleave(function(){
 alert("Bye! You now leave p1!");
});
```

- **mousedown():** The `mousedown()` method attaches an event handler function to an HTML element.
  The function is executed, when the left, middle or right mouse button is pressed down, while the mouse is over the HTML element:

```
$("#p1").mousedown(function(){
 alert("Mouse down over p1!");
});
```

- **mouseup():** The `mouseup()` method attaches an event handler function to an HTML element.
  The function is executed, when the left, middle or right mouse button is released, while the mouse is over the HTML element:

```
$("#p1").mouseup(function(){
 alert("Mouse up over p1!");
});
```

- **hover():** The `hover()` method takes two functions and is a combination of the `mouseenter()` and `mouseleave()` methods.
  The first function is executed when the mouse enters the HTML element, and the second function is executed when the mouse leaves the HTML element:

```
$("#p1").hover(function(){
 alert("You entered p1!");
},
function(){
 alert("Bye! You now leave p1!");
});
```

- **focus():** The `focus()` method attaches an event handler function to an HTML form field. The function is executed when the form field gets focus:

```
$("input").focus(function(){
  $(this).css("background-color", "#cccccc");
});
```

- **blur():** The `blur()` method attaches an event handler function to an HTML form field. The function is executed when the form field loses focus:

```
$("input").blur(function(){
  $(this).css("background-color", "#ffffff");
});
```

- **The on() Method :** The `on()` method attaches one or more event handlers for the selected elements.
  Attach a click event to a <p> element:

```
$("p").on("click", function(){
  $(this).hide();
});
```

  Attach multiple event handlers to a <p> element:

```
$("p").on({
  mouseenter: function(){
    $(this).css("background-color", "lightgray");
  },
  mouseleave: function(){
    $(this).css("background-color", "lightblue");
  },
  click: function(){
    $(this).css("background-color", "yellow");
  }
});
```

jQuery Special Effects

Hide, Show, Toggle, Slide, Fade, and Animate.

jQuery hide() and show()

With jQuery, you can hide and show HTML elements with the `hide()` and `show()` methods:

Syntax:

$(*selector*).hide(*speed,callback*);
$(*selector*).show(*speed,callback*);

The optional **speed parameter** specifies the speed of the effect, 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.

```
$(document).ready(function(){
	$("#hide").click(function(){
		$("p").hide(1000);
	});

	$("#show").click(function(){
		 $("p").show(1000);
	});
});
```

jQuery toggle()

You can also toggle between hiding and showing an element with the `toggle()` method.

Syntax: **$(*selector*).toggle(*speed,callback*);**

```
$("button").click(function(){
  $("p").toggle();
});
```

jQuery Effects – Fading

jQuery has the following fade methods: (https://www.w3schools.com/jquery/jquery_fade.asp)

Method	Syntax	Description
fadeIn()	**$(*selector*).fadeIn(*speed,callback*);**	used to fade in a hidden element.
FadeOut()	**$(*selector*).fadeOut(*speed,callback*);**	used to fade out a visible element.
FadeToggle()	**$(*selector*).fadeToggle(*speed,callback*);**	toggles between the fadeIn() and fadeOut() methods.
fadeTo()	**$(*selector*).fadeTo(*speed,opacity,callback*);**	allows fading to a given opacity (value between 0 and 1).

- 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 required opacity parameter in the fadeTo() method specifies fading to a given opacity (value between 0 and 1).

*Note: For fadeTo() Method, both speed and opacity parameter are not optional.
(https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_fadeto)

jQuery Effects – Sliding

With jQuery you can create a sliding effect on elements.
jQuery has the following slide methods: (https://www.w3schools.com/jquery/jquery_slide.asp)

Method	Syntax	Description
slideDown()	**$(*selector*).slideDown(*speed,callback*);**	used to slide down an element.
slideUp()	**$(*selector*).slideUp(*speed,callback*);**	used to slide up an element.
slideToggle()	**$(*selector*).slideToggle(*speed,callback*);**	toggles between the slideDown() and slideUp() methods.

jQuery Effects – Animation

(https://www.w3schools.com/jquery/jquery_animate.asp)

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.

Example: animate() method moves a element to the right, until it has reached a left property of 250px:

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

***Note:** By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!

jQuery animate() - Manipulate Multiple Properties: Notice that multiple properties can be animated at the same time:

```
$("button").click(function(){
  $("div").animate({
    left: '250px',
    opacity: '0.5',
    height: '150px',
    width: '150px'
  });
});
```

jQuery animate() - Using Relative Values: It is also possible to define relative values (the value is then relative to the element's current value). This is done by putting += or - = in front of the value:

```
$("button").click(function(){
  $("div").animate({
    left: '250px',
    height: '+=150px',
    width: '+=150px'
  });
});
```

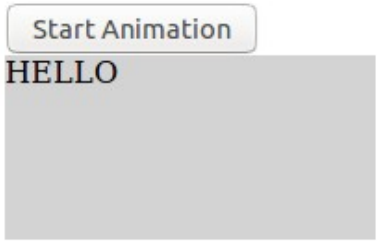
jQuery animate() - Uses Queue Functionality

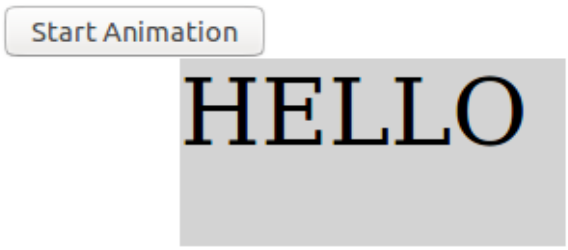
If you write multiple animate() calls after each other, jQuery creates an "internal" queue with these method calls. Then it runs the animate calls ONE by ONE.

```
$("button").click(function(){
  var div = $("div");
  div.animate({height: '300px', opacity: '0.4'}, "slow");
  div.animate({width: '300px', opacity: '0.8'}, "slow");
  div.animate({height: '100px', opacity: '0.4'}, "slow");
  div.animate({width: '100px', opacity: '0.8'}, "slow");
});
```

The example below first moves the `<div>` element to the right, and then increases the font size of the text: (https://www.w3schools.com/jquery/tryit.asp?filename=tryjquery_animation2)

```
$("button").click(function(){
  var div = $("div");
  div.animate({left: '100px'}, "slow");
  div.animate({fontSize: '3em'}, "slow");
});
```





jQuery Stop Animations

The jQuery the stop() method kills the current animation being performed on the selected element.

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.

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

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.

The example below has no callback parameter, and the alert box will be displayed before the hide effect is completed:

```
$("button").click(function(){
  $("p").hide(1000);
  alert("The paragraph is now hidden");
});
```

The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

```
$("button").click(function(){
  $("p").hide("slow", function(){
    alert("The paragraph is now hidden");
  });
});
```

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 methods (on the same element) within a single statement.

Tip: This way, 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:

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

This also works just fine:

```
$("#p1").css("color", "red")
  .slideUp(2000)
  .slideDown(2000);
```

jQuery DOM Manipulation

(Compare with JavaScript DOM manipulation)

One very important part of jQuery is the possibility to manipulate the DOM.
jQuery comes with a bunch of DOM related methods that make it easy to access and manipulate elements and attributes (such as change, add or remove elements)

DOM = Document Object Model
The DOM defines a standard for accessing HTML and XML documents:

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**
- **`attr()` Sets or return attribute values.**

Get Content - text(), html(), and val()

(https://www.w3schools.com/jquery/jquery_dom_get.asp)

The following example demonstrates how to get content with the jQuery `text()` and `html()` methods:

```
$("#btn1").click(function(){
  alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
  alert("HTML: " + $("#test").html());
});
```

The following example demonstrates how to get the value of an input field with the jQuery `val()` method:

```
$("#btn1").click(function(){
  alert("Value: " + $("#test").val());
});
```

Get Attributes – attr()

The jQuery `attr()` method is used to get attribute values.

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

Set Content - text(), html(), and val()

The following example demonstrates how to set content with the jQuery `text()`, `html()`, and `val()` methods:

```
$("#btn1").click(function(){
  $("#test1").text("Hello world!");
});
$("#btn2").click(function(){
  $("#test2").html("<b>Hello world!</b>");
});
$("#btn3").click(function(){
  $("#test3").val("Dolly Duck");
});
```

Set Attributes – attr()

The jQuery `attr()` method is also used to set/change attribute values.

The following example demonstrates how to change (set) the value of the href attribute in a link:

```
$("button").click(function(){
  $("#w3s").attr("href", "https://www.w3schools.com/jquery/");
});
```

Add New HTML Content

We will look at four jQuery methods that are used to add new content:

- **`append()` - Inserts content at the end of the selected elements**

```
$("button").click(function(){
$("p").append("Some appended text.");
```

- **`prepend()` - Inserts content at the beginning of the selected elements**

```
$("button").click(function(){
$("p").prepend("Some prepended text.");
```

- **`after()` - Inserts content after the selected elements**

```
$("img").after("Some text after");
```

- **`before()` - Inserts content before the selected elements**

```
$("img").before("Some text before");
```

Remove Elements/Content

To remove elements and content, there are mainly two jQuery methods:

- **`remove()` - Removes the selected element (and its child elements)**

```
$("#div1").remove();
```

- **`empty()` - Removes the child elements from the selected element**

```
$("#div1").empty();
```

jQuery css() Method

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:

```
$("p").css("background-color", "yellow");
```

Plugins

(https://www.tutorialspoint.com/jquery/jquery-plugins.htm)

(http://tutorials.jenkov.com/jquery/plugins.html)

Plugins are independent units of functionality that can be reused between applications. For instance, a plugin could be as little as a single function, or consists of several functions and objects (data).

A plug-in is piece of code written in a standard JavaScript file. These files provide useful jQuery methods which can be used along with jQuery library methods.

The core concept here is to create something extensible that you can add to your projects to give you quick functionality. jQuery's plugin functionality simplifies the process of building reusable code.

There are plenty of jQuery plug-in available which you can download from repository link at https://jquery.com/plugins.

How to use Plugins

To make a plug-in's methods available to us, we include plug-in file very similar to jQuery library file in the <head> of the document.

We must ensure that it appears after the main jQuery source file, and before our custom JavaScript code.

How to develop a Plug-in

This is very simple to write your own plug-in. Following is the syntax to create a a method −

```
jQuery.fn.methodName = methodDefinition;
```

Here *methodNameM* is the name of new method and *methodDefinition* is actual method definition. The guideline recommended by the jQuery team is as follows −

- Any methods or functions you attach must have a semicolon (;) at the end.
- Your method must return the jQuery object, unless explicity noted otherwise.
- You should use this.each to iterate over the current set of matched elements - it produces clean and compatible code that way.
- Prefix the filename with jquery, follow that with the name of the plugin and conclude with .js.
- Always attach the plugin to jQuery directly instead of $, so users can use a custom alias via noConflict() method.

Example: Following is a small plug-in to have method for changing css. Keep this code in jquery.greenify.js file:

```
jQuery.fn.greenify = function() {
      this.each(function() {
      $(this).css( "color", "green" );
   });
   return this;
};
```

Here is the html page showing usage of above method. Put *jquery.greenify.js* file in same directory of html page.

```
<html>
<head>
      <script type = "text/javascript"
      src = "https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>
      <script src = "jquery.greenify.js" type = "text/javascript">
      </script>

      <script>
         $(document).ready(function() {
            $("div").greenify();
            $("p").greenify();
         });
      </script>

</head>

   <body>
      <p>This is paragraph</p>
      <div>This is division</div>
   </body>
</html>
```

Example: Create Your First jQuery Plugin (https://www.youtube.com/watch?v=gDg5Ypf0hE0)

jQuery for slider design

When we think about a slider, we usually imagine an image gallery slider. However, there are **sliders** for selecting a value or a range of values such as price range sliders.

jQuery UI Slider. **jQuery UI** Sliders are useful to select numeric data as an input from the user within a given range. The main advantage of **slider** over text input is that it becomes impossible for the users to enter an invalid value. Every value they can pick with the **slider** is valid.

To create a slider use the **slider()** method.

To display this slider like any other jQuery UI widget, we have to link to jQuery and jQuery UI. Copy this code inside your HTML file to link the file to jquery and jquery UI through CDN(Content Delivery Network). Here we have used google's CDN but you can also use jquery or Microsoft's CDN

```
<link href= "https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/themes/smoothness/jquery-ui.css"
        rel="stylesheet" >
<script src= "https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">  </script>
<script src= "https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js"> </script>
```

The basic slider is horizontal and has a single handle that can be moved with the mouse or by using the arrow keys.

```
<!doctype html>
<head>
<title>jQuery UI Slider - Default functionality</title>
<link href= "https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/themes/smoothness/jquery-ui.css"
rel="stylesheet" >
<script src= "https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js">  </script>
<script src= "https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js"> </script>

 <script>
 $(document).ready( function() {
   $( "#slider" ).slider();
 } );
 </script>

</head>
<body>
<div id="slider"></div>
</body>
</html>
```

We can use various Options, Methods and Events in the slider function of JqueryUI, which can be seen on following site: (https://api.jqueryui.com/slider/)

https://www.tutorialspoint.com/jqueryui/jqueryui_slider.htm

The following examples shows the usage of range, min, max and values options and change event in the slider function of jQueryUI.

Example 1:


```
<!doctype html>
  <head>
    <title>jQuery UI Slider functionality</title>
    <link href= "https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/themes/smoothness/jquery-ui.css"
     rel="stylesheet" >
    <script src= "https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"> </script>
    <script src= "https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js"> </script>

    <script>
      $(document).ready(function() {
        $( "#slider" ).slider({
                          min: 0,            // min value; default 0
                          max: 100,          // max value
                          value: 47 ,        // position of the handle
                                             //change event is Triggered after the user slides the handle
                          change:function(event,ui){ getDetails(ui.value); }
            });

        var current= $("#slider").slider("option","value");   // shows initial value(47) when page is loaded
        getDetails(current);
       });

      function getDetails(maximum)     //getDetails function is called to display the value in this format
      {                                 //when page is loaded or when the value is changed.
            $("#range").text("$" + maximum);
      }
    </script>
  </head>
  <body>
      <b>Price Range: </b>    <span id= "range"></span>
      <div id = "slider"></div>
  </body>
</html>
```

Example 2: In this example we have set the range option to true to capture a range of values with two drag handles.


```
<!doctype html>
  <head>
    <title>jQuery UI Slider functionality</title>
     <link href= "https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/themes/smoothness/jquery-ui.css"
     rel="stylesheet" >
    <script src= "https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.min.js"> </script>
    <script src= "https://ajax.googleapis.com/ajax/libs/jqueryui/1.12.1/jquery-ui.min.js"> </script>

    <script>
      $(document).ready(function() {
        $( "#slider" ).slider({
                                range:true,
                                min: 0,
                                max: 100,
                                values: [ 20,35] ,
                                change:function(event,ui){
                                      getDetails(ui.values[0],ui.values[1]); }
                                });

        var current= $("#slider").slider("option","values");
        getDetails(current[0],current[1]);
       });

      function getDetails(minimum, maximum)
      {
            $("#range").text("$"+ minimum + "- $" + maximum);
      }
    </script>
  </head>
  <body>
      <b>Price Range: </b>    <span id= "range"></span>
      <div id = "slider"></div>
  </body>
</html>
```

More Examples: **jQuery UI Project: Price Range Filter**

https://www.youtube.com/watch?v=h6V6mQpcEa0

https://www.youtube.com/watch?v=5c4t7lp11Z0

Image Slider:

An image slider is one of the most popular features, specifically on homepages. Basically, it's a slideshow that comprises images, videos, and text that may either scroll automatically or let visitors take charge. These sliders can be placed anywhere on the website with an intention to get more attention.

Example: open the following link to understand this example:

https://www.youtube.com/watch?v=9zWD7MlUA_8&list=PLf1qCOLSl4H0hXYdLxJqVDTwbXaMUoIxh&index=23

```
<!Doctype html>
<html>
    <head>
        <style>
            .picture
            {
                width:500px;
                height:250px;
                top:10px;
                left:10px;
                position: absolute;     /*to display images one below the other*/
            }
        </style>
        <script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>

        <script>
            $(document).ready(function(){
                $(".picture").hide();           //Initially hide all images
                $(".picture:first").fadeIn();   //first image will be faded in
                setInterval(function () {Next($(".picture:visible")) }, 5000);
            });

            function Next (image)   //Next() will be called by setInterval() to fetch the next image.
            {
                $(image).fadeOut();
                var next= ($(image).next().length)? $(image).next() : $(".picture:first");
                next.fadeIn();
            }
        </script>
    </head>
    <body>
            <img class="picture"   src="images/image1.jpeg"/>
            <img class="picture"   src="images/image2.jpeg"/>
            <img class="picture"   src="images/image3.jpeg"/>
            <img class="picture"   src="images/image4.jpeg"/>
    </body>
</html>
```

Explanation:

- Initially all images are hidden except the first image.
- We have used setInterval() function to specify a regular time based trigger to automate the task based on time. Here setInterval() function will show the next image after every 5 seconds.
- How next image is being displayed is explained below:

```
var next= ($(image).next().length)? $(image).next() : $(".picture:first");
```

This is a ternary operator.
if $(image).next().length)? returns 1, then that means there is next image and $(image).next() will be executed, otherwise $(".picture:first") will be exected to fetch the first image, which will be the case after the 4th image.

More examples:

- **Image slider with animation effects:** https://www.youtube.com/watch?v=t0y7o5J68ZQ
- **Image slider with previous and next buttons:**

  https://www.youtube.com/watch?v=J2HLW4A40X8

Using concept of DOM in jQuery, design validation for a registration form:

Compare form validation performed with JavaScript in the previous topics with this one for better understanding.

Following files can be used for reference:

https://html.form.guide/jquery/validation-using-jquery-examples.html

https://code.tutsplus.com/tutorials/easy-form-validation-with-jquery--cms-33096

https://www.tutorialspoint.com/How-to-validate-a-form-using-jQuery

When you include forms on your website for user input, it is always advisable to perform some validation on the data entered. Client side validation is usually done with JavaScript. Client side validation can also be done using jQuery.
We'll see two ways of doing this.

- First we'll validate a form's data by writing the validation rules from scratch
- Next, we'll see how this process can be made easier by using the jQuery Validation Plugin .

1. At first make a html form.
   - Add jQuery library in your file.
   - To use external script file jQform.js we have put the name of the script file in the src (source) attribute of a <script> tag.
   - External CSS file: jQform.css is defined within the <link> element, inside the <head> section of an HTML page:

```
<!Doctype html>
<html>

<head>
    <script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
    <script src="jQform.js"></script>
    <link rel="stylesheet" href="jQform.css" />
</head>

<body>
   <h2>Form:</h2>
   <form id="first_form" method="post" action="">
      <div> <label>Name:</label> <input type="text" id="name" name="name"></input> </div>
      <div> <label>Age:</label> <input type="text" id="age" name="age"></input> </div>
      <div> <label>Email:</label> <input type="email" id="email" name="email"></input> </div>
      <div> <label>Password:</label><input type="password" id="password" name="password"></input></div>
      <div> <input type="submit" value="Submit" /> </div>
   </form>
</body>
</html>
```

2. Now create an External CSS file: jQform.css

```
form label {
    display: inline-block;
    width: 100px;
}
form div {
    margin-bottom: 10px;
}
.error {
    color: red;
    margin-left: 5px;
}
label.error {
    display: inline;
}
```

3. Now create Jquery **Script external file** jQform.js

- preventDefault() prevent submission of the form's data.
- We run some tests on the form's field values to check whether they are valid or not. If the user left a field blank, a span tag will be appended after the field with an error message.

- $(".error").remove(); This removes any element with the error class from the documents. This ensures that on subsequent submissions, the form will not have the previous error messages. If we didn't include this, then every time the form was submitted with errors, an error message would get appended on the previous error message, resulting in the form displaying multiple error messages.

```
$(document).ready(function() {
      $('#first_form').submit(function(e) {
         e.preventDefault();
         var name = $('#name').val();
         var age = $('#age').val();
         var email = $('#email').val();
         var password = $('#password').val();

         $(".error").remove();

         if (name.length < 1) {
            $('#name').after('<span class="error">This field is required</span>');
         }
         if (age < 18) {
            $('#age').after('<span class="error">You must be at least 18 years old</span>');
         }
         if (email.length < 1) {
            $('#email').after('<span class="error">This field is required</span>');
         } else {
            var regEx = /^([a-zA-Z0-9_\.\-\+])+\@(([a-zA-Z0-9\-])+\.)+([a-zA-Z0-9]{2,4})+$/;
            var validEmail = regEx.test(email);
            if (!validEmail) {
               $('#email').after('<span class="error">Enter a valid email</span>');
            }
         }
         if (password.length < 8) {
            $('#password').after('<span class="error">Password must be at least 8 characters long</span>');
         }
      });
});
```

Using the jQuery Validation Plugin

In the previous example, we wrote code that validated the forms data from scratch. To make this process faster and easier, you can use a jQuery validation plugin. With this, you only have to specify a few rules for each field of your form that is to be validated, and the plugin will take care of validation for you.

- Make a html form. First add jQuery library in your file and then add jQuery validation plugin to validate forms' data in easier way.
- Use the External CSS file: jQform.css created above.
- Create external script file jQformplug.js and put the name of the script file in the src (source) attribute of a <script> tag.

```
<!Doctype html>
<html>
<head>
   <script src="http://code.jquery.com/jquery-1.10.2.min.js"></script>
   <script src="https://cdn.jsdelivr.net/jquery.validation/1.16.0/jquery.validate.min.js"></script>
   <script src="jQformplug.js"></script>
    <link rel="stylesheet" href="jQform.css" />
</head>

<body>
   <h2>Form:</h2>
   <form id="first_form" method="post" action="">
      <div> <label>Name:</label> <input type="text" id="name" name="name"></input> </div>
      <div> <label>Age:</label> <input type="text" id="age" name="age"></input> </div>
      <div> <label>Email:</label> <input type="email" id="email" name="email"></input> </div>
      <div> <label>Password:</label><input type="password" id="password" name="password"></input></div>
      <div> <input type="submit" value="Submit" /> </div> </form>
</body>
</html>
```

External jQuery script file jQformplug.js :

```
    $(document).ready(function() {
       $('#first_form').validate({
          rules: {
             name: 'required',
             age: {
                required: true,
                number: true,
                min: 18
             },
             email: {
                required: true,
                email: true,
             },
             password: {
                required: true,
                minlength: 8,
             }
          },
          messages: {      //define error messages.
             age: {
                required: "Please enter your age",
                number: "Please enter your age as a numerical value",
                min: "You must be at least 18 years old"
             },
             email: {
                email: 'Enter a valid email'
             },
             password: {
                minlength: 'Password must be at least 8 characters long'
             }
          },
          submitHandler: function(form) { form.submit();  }
       });
    });
```