
jQuery
write less, do more.

JQuery Introduction

- Fast and concise JS library
- Created by John Resig in 2006
- simplifies
  - HTML document traversing
  - event handling
  - Animation
  - Ajax interactions

What is Jquery?

- **DOM manipulation** − easy to select DOM elements, negotiate them and modify their content by using cross-browser open source selector engine called **Sizzle**.
- **Event handling** − offers an elegant way to capture events without the need to clutter the HTML code itself with event handlers.
- **AJAX Support** − helps to develop a responsive and feature rich site using AJAX technology

What is Jquery?

- **Animations** – comes with plenty of built-in animation effects
- **Lightweight** – The jQuery is very lightweight library – about 85KB in size
- **Cross Browser Support** – has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
- **Latest Technology** – supports CSS3 selectors and basic XPath syntax

How to use jQuery?

- **Local Installation** – You can download jQuery library on your local machine and include it in your HTML code

- **CDN Based Version** – You can include jQuery library into your HTML code directly from Content Delivery Network (CDN)

Local Installation

- Go to the https://jquery.com/download/ to download the latest version available

- Now put downloaded **jquery-3.3.1.min.js** file in a directory of your website, e.g. /jquery

Example-1 (Local Installation)

```
<script src = "jquery/jquery-3.3.1.min.js"></script>

<script>
        $(document).ready(function() {
                document.write("Hello,World!");
    });
</script>

<body>
        <h1>This statement will be overwritten</h1>
</body>
```

Example-2 (CDN based version)

```
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.3.1/
jquery.min.js"></script>

<script>
        $(document).ready(function() {
                document.write("Hello,World!");
    });
</script>
<body>
         <h1>This statement will be overwritten</h1>
</body>
```

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

```
$(document).ready(function() {
        // do stuff when DOM is ready
});
```

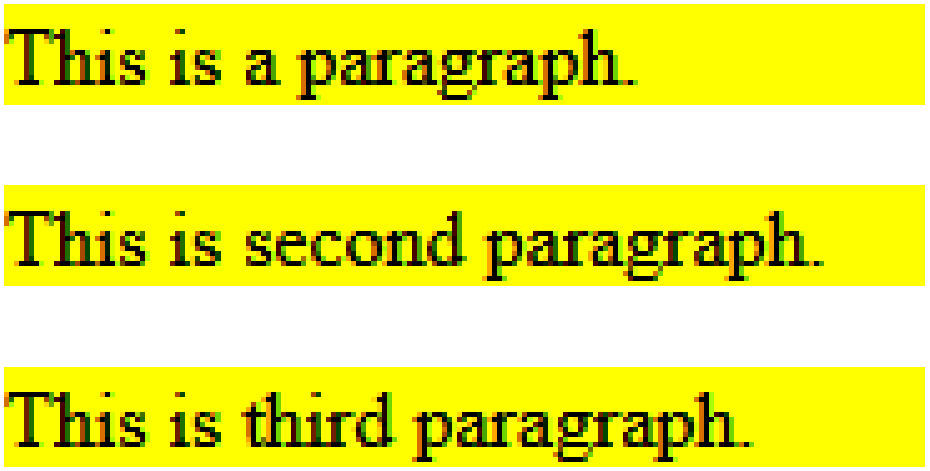
- This is to prevent any jQuery code from running before the document is finished loading (is ready).
- It is good practice to wait for the document to be fully loaded and ready before working with it. This also allows you to have your JavaScript code before the body of your document, in the head section.
- 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 (Ex. 1)

```
<script>
        $(document).ready(function() {
                $("p").css("background-color", "yellow");
        });
</script>
```

```
<body>
     <div>
      <p class = "myclass">This is a paragraph.</p>
      <p id = "myid"> This is second paragraph.</p>
      <p> This is third paragraph.</p>
     </div>
</body>
```

jQuery Selectors (Ex.1 Output)



How to use Selectors?

Selector	Description
Name	Selects all elements which match with the given element **Name**.
#ID	Selects a single element which matches with the given **ID**.
.Class	Selects all elements which match with the given **Class**.
Universal (*)	Selects all elements available in a DOM.
Multiple Elements (E, F, G)	Selects the combined results of all the specified selectors **E, F** or **G**

Selectors Examples

Selector	Description
$("*")	selects all elements in the document
$("p > *")	selects all elements that are children of a paragraph element
$("#specialID")	gets the element with id="specialID"
$("li:not(.myclass)")	Selects all elements matched by that do not have class = "myclass"
$("a#specialID.specialClass")	matches links with an id of *specialID* and a class of *specialClass*
$("p a.specialClass")	matches links with a class of *specialClass* declared within <p> elements
$("ul li:first")	gets only the first element of the

Events Handling (ready, focus, blur)

```
<script>
    $(document).ready(function() {
        $("input").focus (function(){
            $(this).css("background-color", "#cccccc");
        });
        $("input").blur (function(){
            $(this).css("background-color", "#ffffff");
        });
    });
</script>
```

```
Name: <input type="text" name="fullname">   <br>
Email: <input type="text" name="email">
```

Events Handling (Output)

Get Attribute Value (Ex.)

```
<script>
    $(document).ready(function() {
            var title = $("em").attr("title");
            $("#divid").text(title);
     });
</script>
```

```
<div>
   <em title = "Bold and Brave">first paragraph.</em>
   <p id = "myid">This is second paragraph.</p>
   <div id = "divid"></div>
</div>
```

jQuery – Attributes (Ex. Output)

This is first paragraph.

This is second paragraph.

Bold and Brave

Set Attribute Value (Ex.)

```
<script>
        $(document).ready(function() {
             $("#myimg").attr("src", "images/jquery.png");
        });
</script>
```

```
<body>
     <div>
          <img id = "myimg" src = ""/>
     </div>
</body>
```

Set Attribute Value (Ex. Output)

Applying Styles (addClass)

- The **addClass(classes)** method can be used to apply defined style sheets onto all the matched elements.

- You can specify multiple classes separated by space.

Applying Styles (addClass)

```
<script>
        $(document).ready(function() {
              $("em").addClass("selected");
              $("#myid").addClass("highlight");
        });
</script>
```

```
<style>
        .selected { color:red; }
        .highlight { background:yellow; }
</style>
```

```
<body>
    <em title = "Bold and Brave">This is first paragraph.</em>
    <p id = "myid">This is second paragraph.</p>
</body>
```

Applying Styles (output)

This is first paragraph.

This is second paragraph.

DOM Traversing

- jQuery provides a variety of DOM traversal methods to help us select elements in a document *randomly* as well as in *sequential* method

- Most of the DOM Traversal Methods do not modify the jQuery object and they are used to filter out elements from a document based on given conditions

Find Elements by Index

```
<script>
        $(document).ready(function() {
            $("li").eq(2).addClass("selected");
        });
</script>
```

```
<style>  .selected { color:red; }  </style>
```

```
<div>  <ul>
          <li>list item 1</li>          <li>list item 2</li>
          <li>list item 3</li>          <li>list item 4</li>
          <li>list item 5</li>          <li>list item 6</li>
 </ul> </div>
```

Find Elements by Index (output)

- list item 1
- list item 2
- list item 3
- list item 4
- list item 5
- list item 6

Filtering out Elements

- The **filter(selector)** method can be used to filter out all elements from the set of matched elements that do not match the specified **selector**

Filtering out Elements

```
<script>
        $(document).ready(function() {
                $("li").filter(".middle").addClass("selected");
        });
</script>
```

```
<style>  .selected { color:red; } </style>
```

```
<div>  <ul>
          <li class = "top">list item 1</li>
          <li class = "middle">list item 2</li>
          <li class = "bottom">list item 3</li>
 </ul> </div>
```

Filtering out Elements (Output)

- list item 1
- list item 2
- list item 3

Filter(function) method

```
$(document).ready(function() {
        $("li").filter(function (index) {
            return (index == 1 ||
                    $(this).attr("class") == “bottom");
         }).addClass("selected");
 });
```

```
<style>  .selected { color : red; }   </style>
```

```
<div>  <ul>
        <li class = "top">list item 1</li>
        <li class = "middle">list item 2</li>
        <li class = "bottom">list item 3</li>
 </ul> </div>
```

Filter(function) method

- list item 1
- list item 2
- list item 3

Locating Descendant Elements

```
<script>
	$(document).ready(function() {
		$("p").find("span").addClass("selected");
	});
</script>
```

```
<style>    .selected { color:red; }   </style>
```

```
<p>1st paragraph <span>THIS IS RED</span></p>
<p>2nd paragraph <span>THIS IS ALSO RED</span></p>
```

Locating Descendant Elements (Output)

1st paragraph THIS IS RED

2nd paragraph THIS IS ALSO RED

JQuery DOM Traversing Methods

```
$(document).ready(function() {
        $("div").children(".selected").addClass("blue");
});
```

```
<style>   .blue { color : blue; } </style>
<div>
        <span> Hello </span>
        <p class = "selected">Hello Again</p>
        <div class = "selected">And Again</div>
        <p class = "biggest“  >And One Last Time</p>
</div>
```

JQuery DOM Traversing Methods (output)

Hello

Hello Again

And Again

And One Last Time

CSS Selectors Methods

The jQuery library supports nearly all of the selectors included in Cascading Style Sheet (CSS) specifications 1 through 3, as outlined on the World Wide Web Consortium's site

Apply CSS Properties

```
$(document).ready(function() {
        $("li").eq(2).css("color", "red");
});
```

```
<div>
    <ul>
        <li>list item 1</li>
        <li>list item 2</li>
        <li>list item 3</li>
        <li>list item 4</li>
        <li>list item 5</li>
        <li>list item 6</li>
    </ul>
 </div>
```

Apply CSS Properties (output)

- list item 1
- list item 2
- list item 3
- list item 4
- list item 5
- list item 6

Apply Multiple CSS Properties

```
$(document).ready(function(){
                $("li").eq(2).css({"color":"red",
                                   "background-color":"yellow"});
});
```

```
<div> <ul>
            <li>list item 1</li>
            <li>list item 2</li>
            <li>list item 3</li>
            <li>list item 4</li>
            <li>list item 5</li>
            <li>list item 6</li>
 </ul> </div>
```

Apply Multiple CSS Properties (output)

- list item 1
- list item 2
- list item 3
- list item 4
- list item 5
- list item 6

Setting Element Width & Height

```
$("div:first").width(100);
$("div:first").css("background-color", "blue");
```

```
div {
        width:70px; height:50px; float:left;
        margin:5px; background:red;
        cursor:pointer;
}
```

```
<div></div>
<div>d</div>
<div>d</div>
<div>d</div>
<div>d</div>
```

Setting Element Width & Height (Output)

DOM Manipulation

```
$("div").click(function () {
            var content = $(this).html();
            $("#result").text( content );
});
```

```
#box { margin:10px;                    padding:12px;
          border:2px solid #666;      width:60px;      }
```

```
<p>Click on the square below:</p>
<span id = "result"></span>
<div id = "box" style = "background-color:yellow;">
        This is Yellow Square!!
</div>
```

DOM Manipulation (Output)

Click on the square below:

Click on the square below:

This is Yellow Square!!

DOM Manipulation

```
$("div").click(function () {
        $(this).html("My content is changed!!!");
});
```

```
#box { margin:10px;                    padding:12px;
       border:2px solid #666;      width:60px;}
```

```
<p>Click on the square below:</p>

<div id = "box" style = "background-color:yellow;">
      This is Yellow Square!!
</div>
```

DOM Manipulation (Output)

Click on the square below:

Click on the square below:

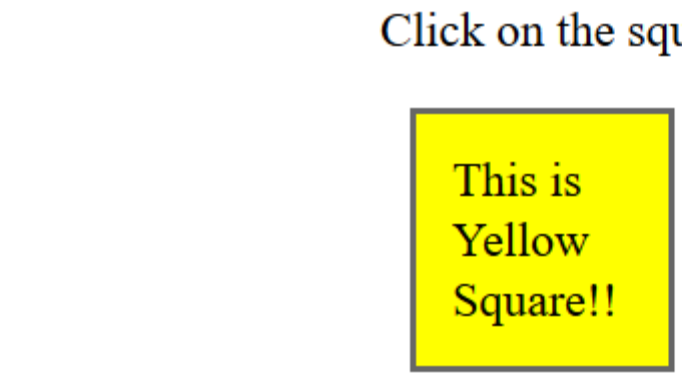
DOM Element Replacement

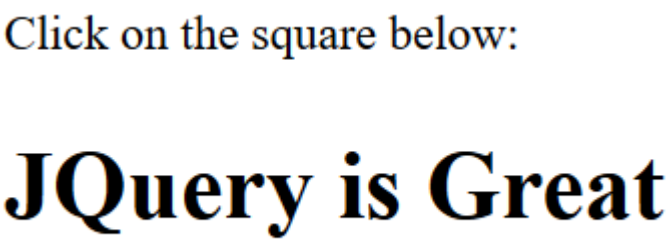
```
$("div").click(function () {
          $(this).replaceWith("<h1>JQuery is Great</h1>");
 });
```

```
#box  { margin:10px;                    padding:12px;
          border:2px solid #666;    width:60px;     }
```

```
<p>Click on the square below:</p>

<div id = "box" style = "background-color:yellow;">
       This is Yellow Square!!
</div>
```

DOM Element Replacement (Output)





Removing DOM Elements

- The jQuery **empty()** method removes the child elements of the selected element(s).

- The jQuery **remove()** method removes the selected element(s) and its child elements.

Removing DOM Elements

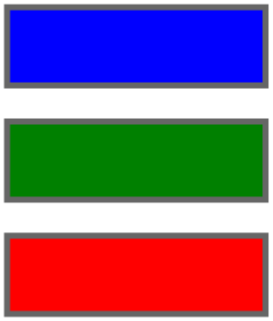
```
$("div").click(function () {
            $(this).remove( );
});
```

```
div {  margin:10px;                    padding:12px;
       border:2px solid #666;     width:60px;     }
```

```
<p>Click on any square below:</p>

<div style = "background-color:blue;">    </div>
<div style = "background-color:green;">  </div>
<div style = "background-color:red;">     </div>
```

Removing DOM Elements (Output)

Click on any square below:



Click on any square below:

Click on any square below:

Inserting DOM Elements

- The **after(content)** method insert content after each of the matched elements

- The **before(content)** method inserts content before each of the matched elements

Inserting DOM Elements

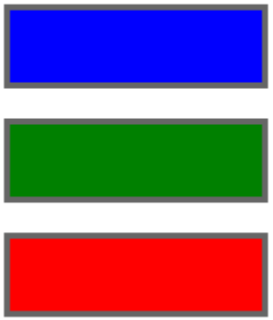
```
$("div").click(function () {
            $(this).before('<div class="div"></div>' );
});
```

```
.div { margin:10px;                    padding:12px;
       border:2px solid #666;     width:60px;}
```

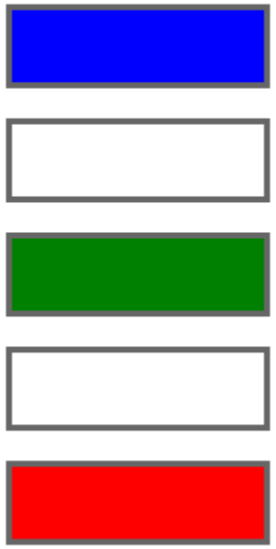
```
<p>Click on any square below:</p>
<span id = "result"> </span>

<div class = "div" style = "background-color:blue;">
</div>
<div class = "div" style = "background-color:green;">
</div>
<div class = "div" style = "background-color:red;">
</div>
```

Inserting DOM Elements (Output)

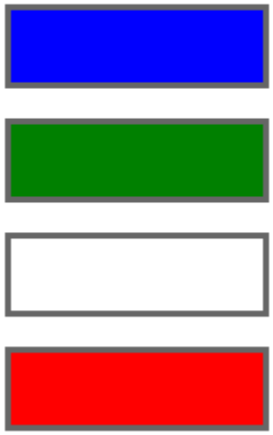
Click on any square below:



Click on any square below:



Click on any square below:



jQuery – Event Handling

- We can create dynamic web pages by using events
- Events are actions that can be detected by your Web Application.
- Examples events:
  - A mouse click
  - A web page loading
  - Taking mouse over an element
  - Submitting an HTML form
  - A keystroke on your keyboard, etc.

Binding Event Handlers

The full syntax of the bind() command :

selector.bind(eventType, [eventData], handler)

eventType − A string containing a JavaScript event type, such as click or submit

eventData − This is a map of data that will be passed to the event handler.

handler − A function to execute each time the event is triggered.

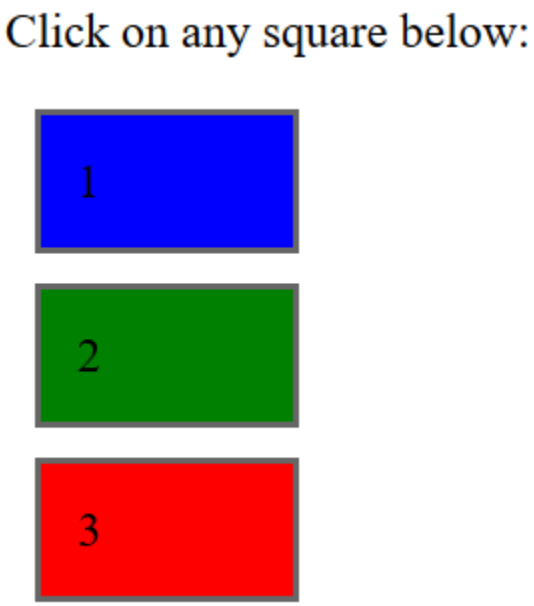
Binding Event Handlers

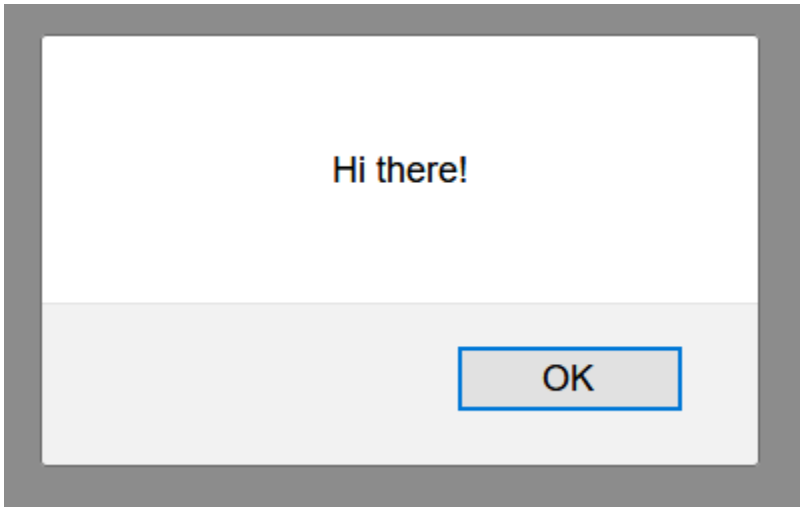
```
$('div').bind('click', function( event ){
            alert('Hi there!');
});
```

```
.div { margin:10px;                  padding:12px;
       border:2px solid #666;   width:60px;}
```

```
<p>Click on any square below:</p>

<div class = "div" style = "background-color:blue;">1</div>
<div class = "div" style="  background-color:green;">2</div>
<div class = "div" style = "background-color:red;">3</div>
```

Binding Event Handlers (Output)





The Event Object & its Attributes

- The callback function takes a single parameter - the JavaScript event object
- The event object is often unnecessary and the parameter is omitted
- There are certain attributes which you can access

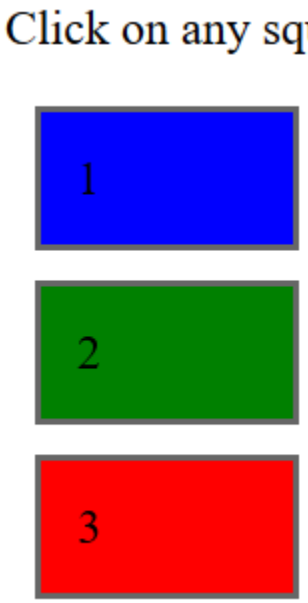
The Event Object & its Attributes

```
$('div').bind('click', function( event ){
        console.log('Event type is ' + event.type);
        console.log ('pageX : ' + event.pageX);
        console.log ('pageY : ' + event.pageY);
        console.log ('Target : ' + event.target.innerHTML);
});
```

```
.div { margin:10px;  padding:12px;
        border:2px solid #666; width:60px;}
```

```
<p>Click on any square below:</p>

<div class = "div" style = "background-color:blue;">1</div>
<div class = "div" style="background-color:green;">2</div>
<div class = "div" style = "background-color:red;">3</div>
```

Removing Event Handlers (Output)




```
Event type is click
pageX : 47
pageY : 73
Target : 1

Event type is click
pageX : 67
pageY : 129
Target : 2

Event type is click
pageX : 66
pageY : 189
Target : 3
```

jQuery - AJAX

- **load()** method to load any static or dynamic data using AJAX

[selector].load(URL, [data], [callback]);

- **URL** − The URL of the server-side resource to which the request is sent

- **data** − This is an object whose properties are serialized into properly encoded parameters to be passed to the request. If specified, the request is made using the **POST** method. If omitted, the **GET** method is used

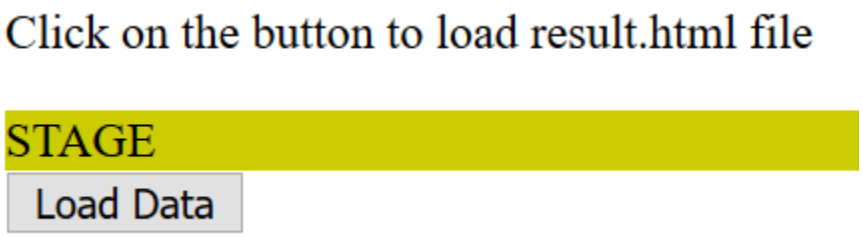
- **callback** − A callback function invoked after the response data has been loaded into the elements of the matched set. The first parameter passed to this function is the response text received from the server and second parameter is the status code

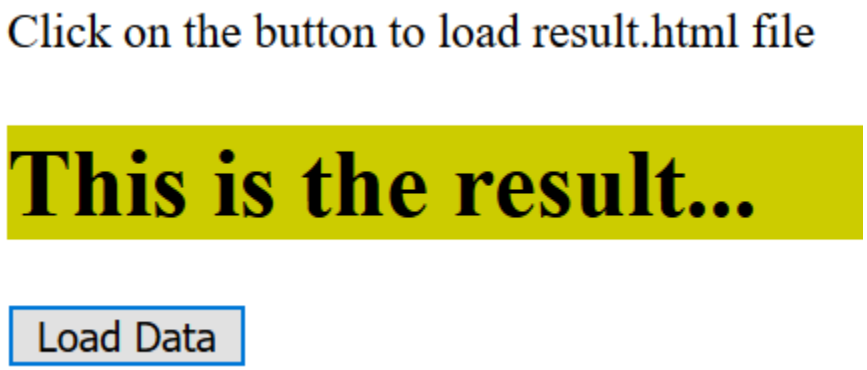
Loading Simple Data

```
$("#driver").click(function(event){
		$('#stage').load('result.html');
});
```

```
<p>Click on the button to load result.html file</p>

<div id = "stage" style = "background-color:cc0;">
	STAGE
</div>

<input type = "button" id = "driver" value = "Load Data" />
```

Loading Simple Data (Output)





Getting JSON Data

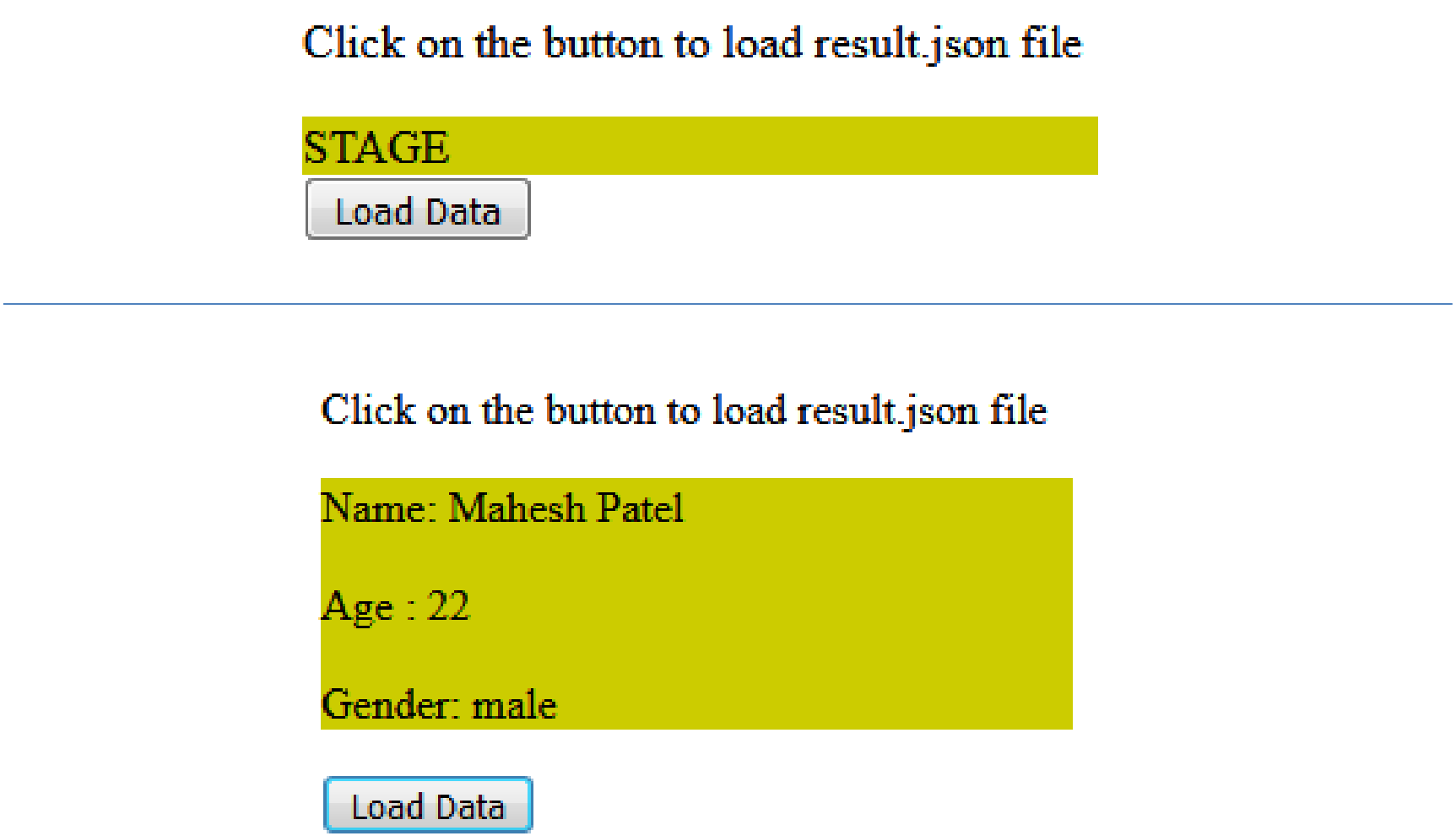
[selector].getJSON(URL, [data], [callback]);

- **URL** − The URL of the server-side resource contacted via the GET method.
- **data** − An object whose properties serve as the name/value pairs used to construct a query string to be appended to the URL, or a preformatted and encoded query string
- **callback** − A function invoked when the request completes. The data value resulting from digesting the response body as a JSON string is passed as the first parameter to this callback, and the status as the second.

Getting JSON Data

```
$("#driver").click(function(event) {
    $.getJSON('result.json', function(jd) {
      $('#stage').html('<p> Name: ' + jd.name + '</p>');
      $('#stage').append('<p>Age : ' + jd.age+ '</p>');
      $('#stage').append('<p> Gender: ' + jd.gender+ '</p>');
    });
});

<p>Click on the button to load result.json file</p>
<div id = "stage" style = "background-color : #cc0;">
        STAGE
</div>
<input     type = "button"   id = "driver"   value = "Load Data" />
```

Getting JSON Data (Output)



jQuery – Effects

- hide(), show()
- fadeIn(), fadeOut(), fadeToggle(), fadeTo()
- slideDown(), slideUp(), slideToggle()
- animate(), stop()

Effects – Show and Hide Elements

[selector].show([speed], [callback])
[selector].hide([speed], [callback])

- **speed** − A string ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000)

- **callback** − a function to be executed whenever the animation completes; executes once for each element animated against.

Effects – Show and Hide Elements

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

```
.mydiv { margin:10px;      padding:12px;
       border:2px solid #666; width:100px;   height:100px;  }
```

```
<div class = "mydiv">
       This is a SQUARE
</div>
<input id = "hide" type = "button" value = "Hide" />
<input id = "show" type = "button" value = "Show" />
```

Effects – Show and Hide Elements (Output)

Fading Effects

- With jQuery you can fade an element in and out of visibility
- jQuery has the following fade methods:
  - fadeIn()
  - fadeOut()
  - fadeToggle()
  - fadeTo()

jQuery fadeIn() Method

$(*selector*).fadeIn([*speed], [callback]*);

speed – “slow”, “fast” or milliseconds

callback – parameter is a function to be executed after the fading completes

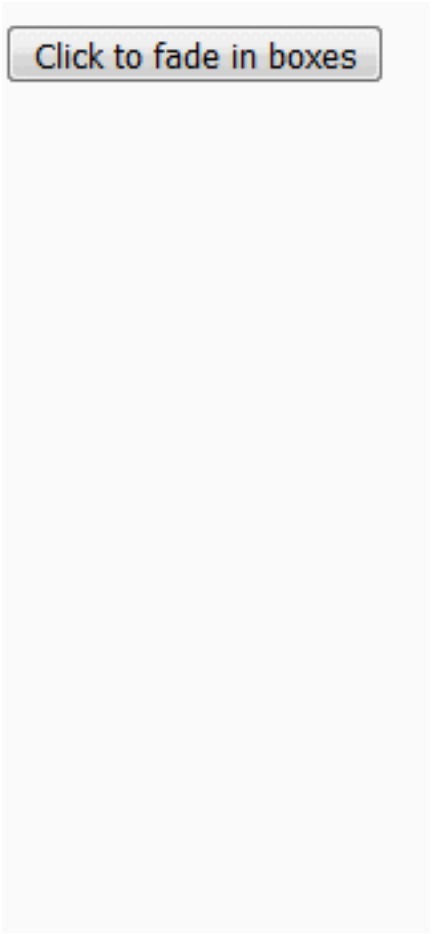
Fade-in Effect

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

```
div  { width:80px;        height:80px;        display:none;   }
```

```
<button>Click to fade in boxes</button>

<div id="div1" style="background-color:red;"></div><br>
<div id="div2" style="background-color:green;"></div><br>
<div id="div3" style="background-color:blue;"></div>
```

Fade-in Effect (Output)



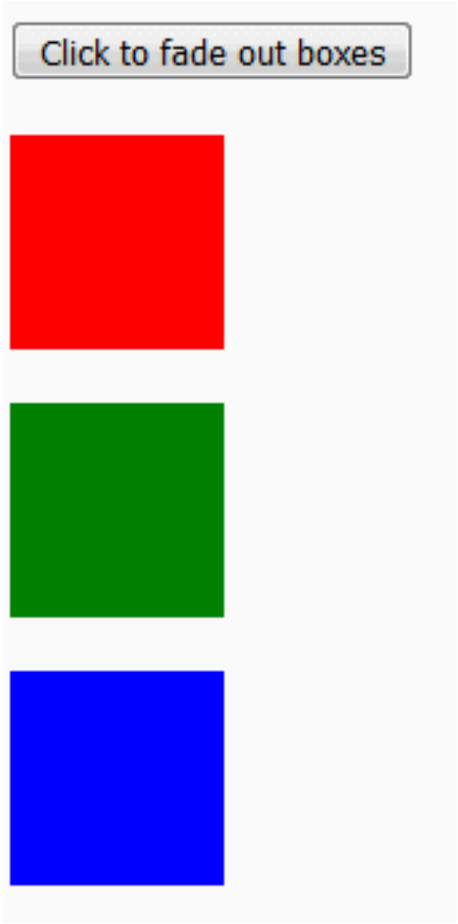
Fade-out Effects

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

```
div  { width:80px;       height:80px;   }
```

```
<button>Click to fade out boxes</button>

<div id="div1" style="background-color:red;"></div><br>
<div id="div2" style="background-color:green;"></div><br>
<div id="div3" style="background-color:blue;"></div>
```

Fade-out Effect (output)



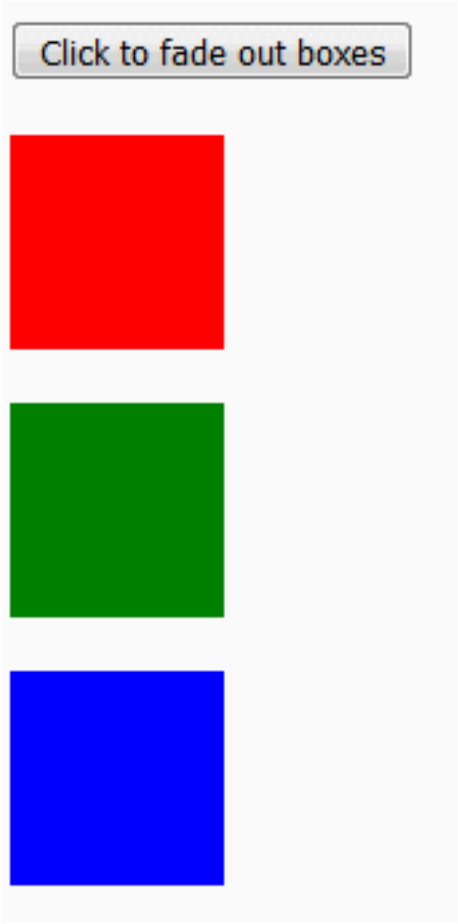
Fade-toggle Effect

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

```
div  { width:80px;       height:80px;   }
```

```
<button>Click to fade in/out boxes</button>

<div id="div1" style="background-color:red;"></div><br>
<div id="div2" style="background-color:green;"></div><br>
<div id="div3" style="background-color:blue;"></div>
```

Fade-toggle Effect (output)



Fade-to Effects

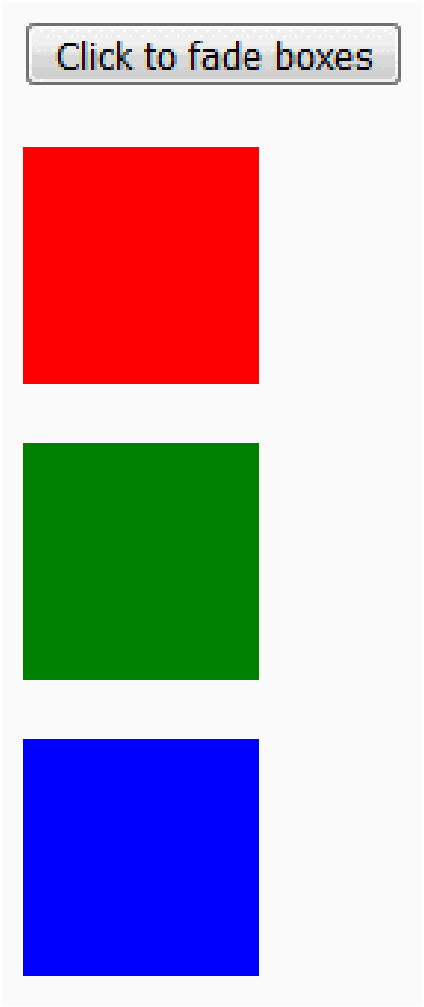
$(*selector*).fadeTo(*speed, opacity, [callback]*);

```
$("button").click(function(){
    $("#div1").fadeTo("slow". 0.2);
    $("#div2").fadeTo("slow", 0.5);
    $("#div3").fadeTo("slow", 0.7);
});
```

```
div  { width:80px;       height:80px;   }
```

```
<button>Click to fade in/out boxes</button>
<div id="div1" style="background-color:red;"></div><br>
<div id="div2" style="background-color:green;"></div><br>
<div id="div3" style="background-color:blue;"></div>
```

Fade-to Effect (output)



jQuery Sliding Effects

- With jQuery you can create a sliding effect on elements

- jQuery has the following slide methods:
  - slideDown()
  - slideUp()
  - slideToggle()

jQuery Sliding Methods Syntax

```
$(selector).slideDown([speed], [callback]);
$(selector).slideUp([speed], [callback]);
$(selector).slideToggle([speed], [callback]);
```

speed – “slow”, “fast” or milliseconds

callback – parameter is a function to be executed after the fading completes

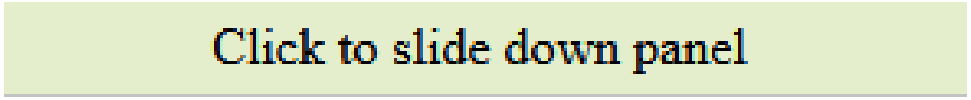
Sliding Effect

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

```
#panel, #flip {  padding: 5px;  text-align: center;
                 background-color: #e5eecc;
                 border: solid 1px #c3c3c3;
}
#panel {
    padding: 50px;    display: none;
}
```

```
<div id="flip">Click to slide down panel</div>
<div id="panel">Hello world!</div>
```

Sliding Effect (output)



jQuery Animations

$(*selector*).animate({*params*},*[speed], [callback]*);

- params – defines the CSS properties to be animated
- speed – specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds
- callback – parameter is a function to be executed after the animation completes

jQuery Animations

- 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!
- all property names must be camel-cased when used with the animate() method
- e.g. you will need to write paddingLeft instead of padding-left

jQuery animate()

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

```
div {
    background : red;          height:100px;
    width:100px; margin: 20px;    position:absolute;
}
```

```
<button>Start Animation</button>
<div></div>
```

jQuery animate (output)

jQuery animate() – Use of queue

```
$("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");
});
```

```
div {
    background : blue;          height:100px;
    width:100px; margin: 20px;    position:absolute;
}
```

```
<button>Start Animation</button>
<div></div>
```

jQuery animate (output)

Toggling the Elements

- jQuery provides methods to toggle the display state of elements between revealed or hidden.

- If the element is initially displayed, it will be hidden; if hidden, it will be shown

[selector].toggle([speed], [callback]);

Toggling the Elements

```
$(".clickme").click(function(event){
            $(".target").toggle('slow', function(){
                    $(".log").text('Transition Complete');
            });
});
```

```
.clickme {  margin:10px;    padding:12px;    width:100px;
          border:2px solid #666;     height:50px;        }
```

```
<div class = "content">
        <div class = "clickme">Click Me</div>
        <div class = "target">
            <img src = "./images/jquery.png" alt = "jQuery" />
        </div>
        <div class = "log"></div>
</div>
```

Toggling the Elements (Output)

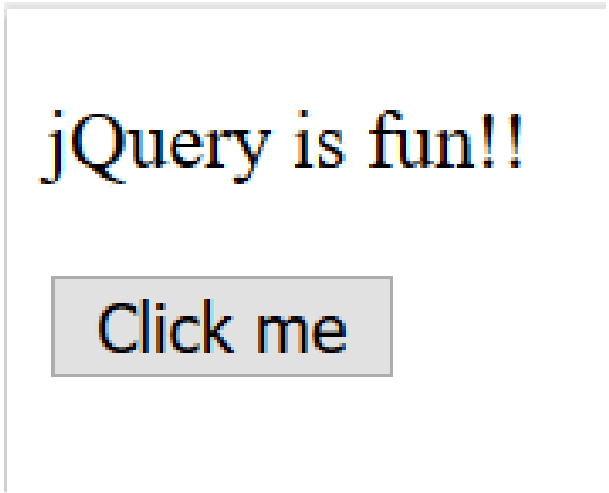
jQuery Method Chaining

- With jQuery, you can chain together actions/methods
- Chaining allows us to run multiple jQuery methods (on the same element) within a single statement

jQuery Method Chaining

```
$("button").click(function(){
        $("#p1").css("color","red")
                .slideUp(2000)
                .slideDown(2000);
});

<p id="p1">jQuery is fun!!</p>
<button>Click me</button>
```

jQuery Method Chaining (Output)



jQuery – Utilities

$.trim()	$.each()	$.inArrary()
$.extend()	$.proxy()	$.fn.extend()
$.contains()	$.data()	$.dequeue()
$.isWindow()	$.globalEval()	

$.trim()

- $.trim() is used to Removes leading and trailing whitespace

- Ex.:

```
$.trim( " lots of extra whitespace     " );
```

$.each()

- $.each() is used to Iterates over arrays and objects

```
$.each([ "foo", "bar", "baz" ],
  function( idx, val ) {
  console.log( "elem" + idx + " is " +val );
  }
);
```

$.inArray()

- $.inArray() is used to Returns a value's index in an array, or -1 if the value is not in the array

```
var myArray = [ 1, 2, 3, 5 ];
if ( $.inArray( 4, myArray ) !== -1 ) {
    console.log( "found it!" );
}
```

$.extend()

- $.extend() is used to change the properties of the first object using the properties of subsequent objects

```
var first = { name: "Jack", age: 23 };
var second = { name: "John" };

var third = $.extend( first, second );

console.log( first.name + "  " +  third.name );
```

$.proxy()

- $.proxy() returns a function that will always run in the provided scope — that is, sets the meaning of “this” inside the passed function to the second argument

$.proxy()

```
var myFunction = function() {
        console.log(this);
};
var myObject = {
      name: "John“
};
myFunction();       // window

var myProxyFunction = $.proxy( myFunction,myObject );

myProxyFunction();    //myObject
```

$.contains()

- $.contains() is used to returns true if the DOM element provided by the second argument is a descendant of the DOM element provided by the first argument, whether it is a direct child or nested more deeply.

- Examples:

  - $.contains(document.documentElement, document.body); //true

  - $.contains(document.body, document.documentElement); //false

References

- https://www.tutorialspoint.com/jquery/
- https://www.w3schools.com/jquery/default.asp