F22 IN3033 Markup Languages Lab Test jQuery

A picture containing text, clipart

Description automatically generatedDue: November 12

What is jQuery?<https://learn.jquery.com/>

jQuery is a fast and concise JavaScript Library created by John Resig in 2006 with a nice motto: **Write less, do more**.

jQuery simplifies HTML document object model traversal, event handling, animating, and Ajax interactions for rapid web development. jQuery is your tool to manipulate the BOM: reading and writing data, and detecting events.

jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code.

**jQuery is a SET OF APIs for quickly creating complex Workflows.**

**Here is the list of important core features supported by jQuery –**

The basis of jQuery and CSS is SELECTORS.

You can reference this material on CSS:

<https://www.dropbox.com/sh/f9saf563zzr6k5s/AABONssOnbhSdQdTwRxvenNja?dl=0>

<https://www.w3schools.com/jquery/jquery_ref_selectors.asp>

**3 kinds of Selectors:**

1. HTML Tag Name
2. You can assign a Class name to an HTML DOM Elements.
3. You can assign an ID name to an HTML Dom element.

* **DOM manipulation** − The jQuery made it easy to select DOM elements, negotiate them and modifying their content by using cross-browser open source selector engine called **Sizzle**.
* **Event handling** − The jQuery offers an elegant way to capture a wide variety of events, such as a user clicking on a link, without the need to clutter the HTML code itself with event handlers.
* **AJAX Support** − The jQuery helps you a lot to develop a responsive and featurerich site using AJAX technology.
* **Animations** − The jQuery comes with plenty of built-in animation effects which you can use in your websites.
* **Lightweight** − The jQuery is very lightweight library - about 19KB in size (Minified and gzipped).
* **Cross Browser Support** − The jQuery has cross-browser support, and works well in IE 6.0+, FF 2.0+, Safari 3.0+, Chrome and Opera 9.0+
* **You can implement the BOOTSTRAP JavaScript library to achieve cross browser.**
* **Latest Technology** − The jQuery supports CSS3 selectors and basic XPath syntax.

**How to use jQuery?**

There are two ways 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-2.1.3.min.js** file in a directory of your website, e.g. /jquery.

Example

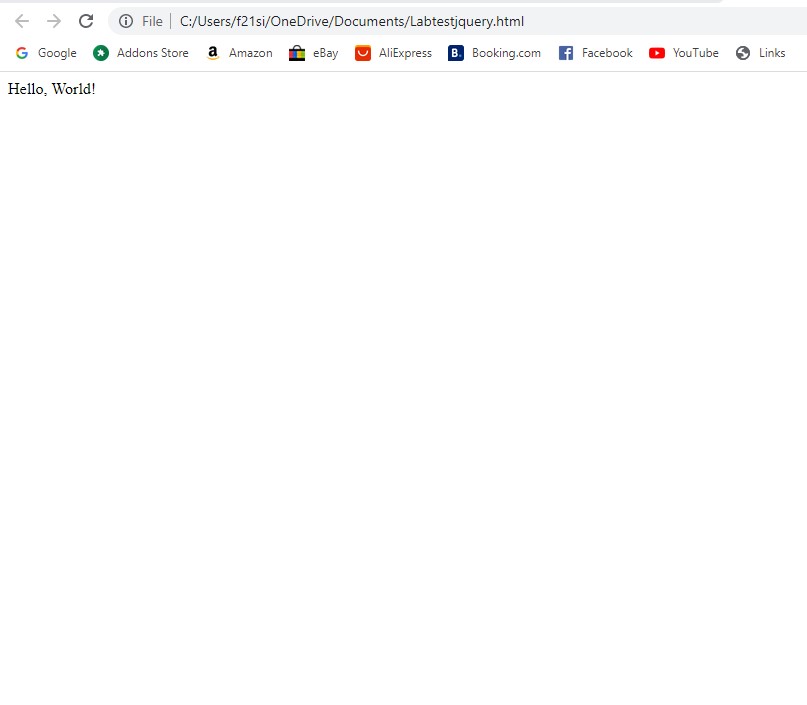
Now you can include *jquery* library in your HTML file as follows –

**Enter this code:**

Text

Description automatically generated

Provide screen shot of working code here:



CDN Based Version

You can include jQuery library into your HTML code directly from Content Delivery Network (CDN). Google and Microsoft provides content deliver for the latest version.

We are using Google CDN version of the library throughout this tutorial.

Example

Now let us rewrite above example using jQuery library from Google CDN.

[Live Demo](http://tpcg.io/K7aHSK)

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript">

$(document).ready(function(){

document.write("Hello, World!");

});

</script>

</head>

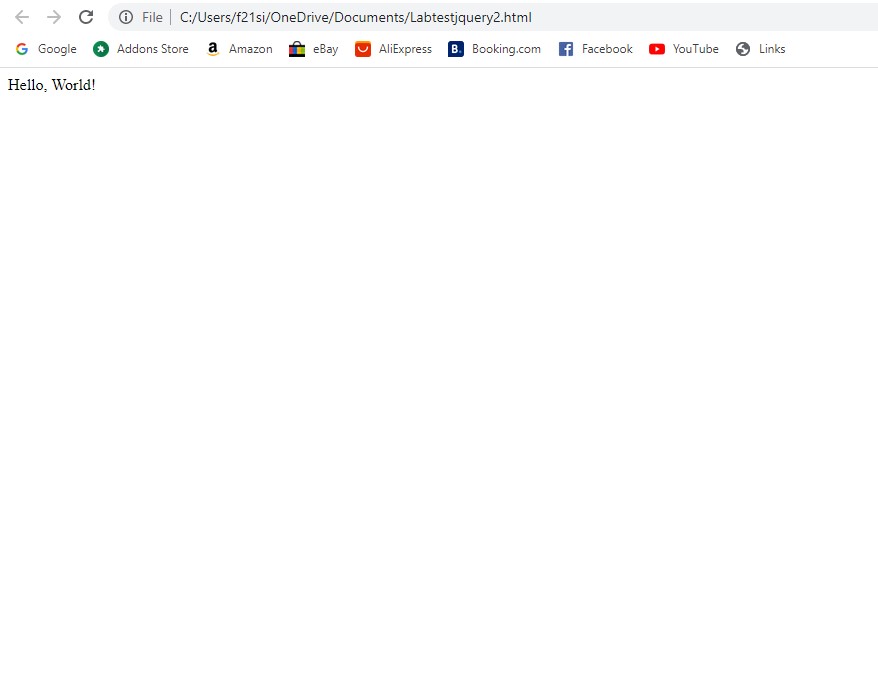
<body>

<h1>Hello</h1>

</body>

</html>

This will produce following result –



How to Call jQuery Library Functions?

As almost everything, we do when using jQuery reads or manipulates the document object model (DOM), we need to make sure that we start adding events etc. as soon as the DOM is ready.

If you want an event to work on your page, you should call it inside the $(document).ready() function.

Everything inside it will load as soon as the DOM is loaded and before the page contents are loaded.

To do this, we register a ready event for the document as follows −

$(document).ready(function() {

// do stuff when DOM is ready

});

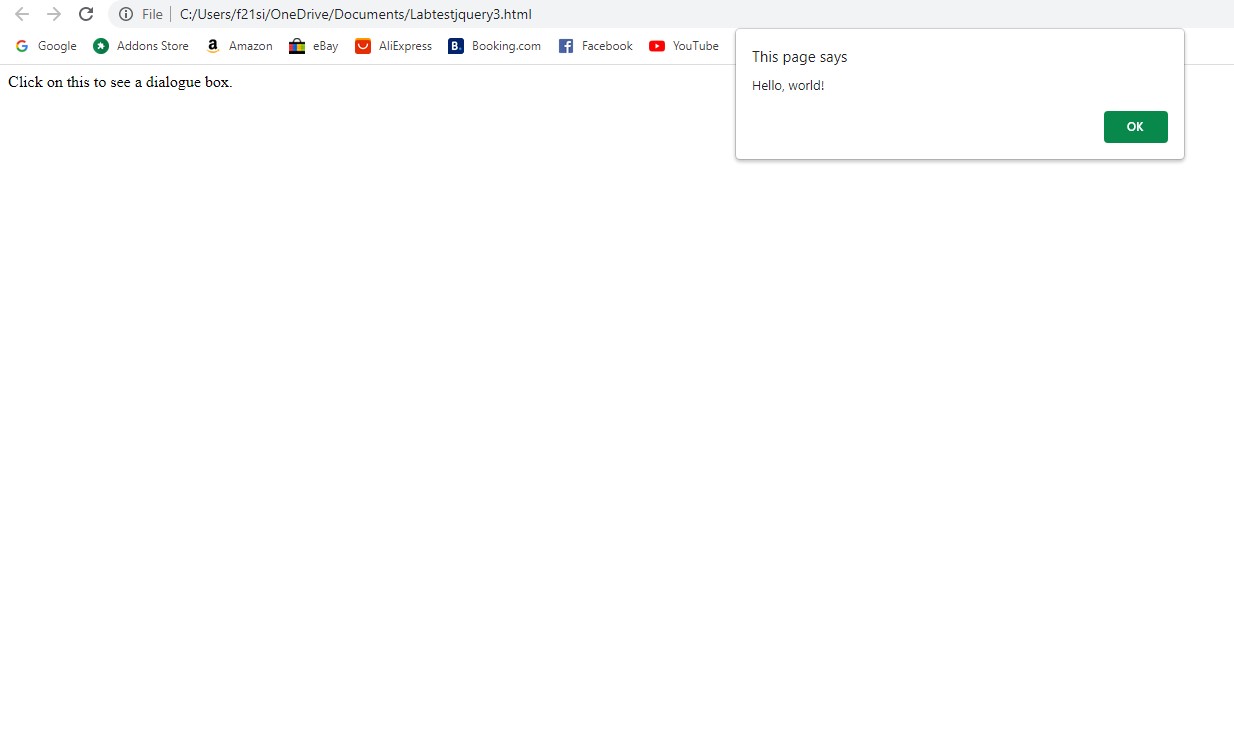
To call upon any jQuery library function, use HTML script tags as shown below –

Enter this code:

Text

Description automatically generated

**Screen Shot the result you get:**

****

How to Use Custom Scripts:

It is better to write our custom code in the custom JavaScript file : **custom.js**, as follows −

/\* Filename: custom.js \*/

$(document).ready(function(){

$("div").click(function(){

alert("Hello, world!");

});

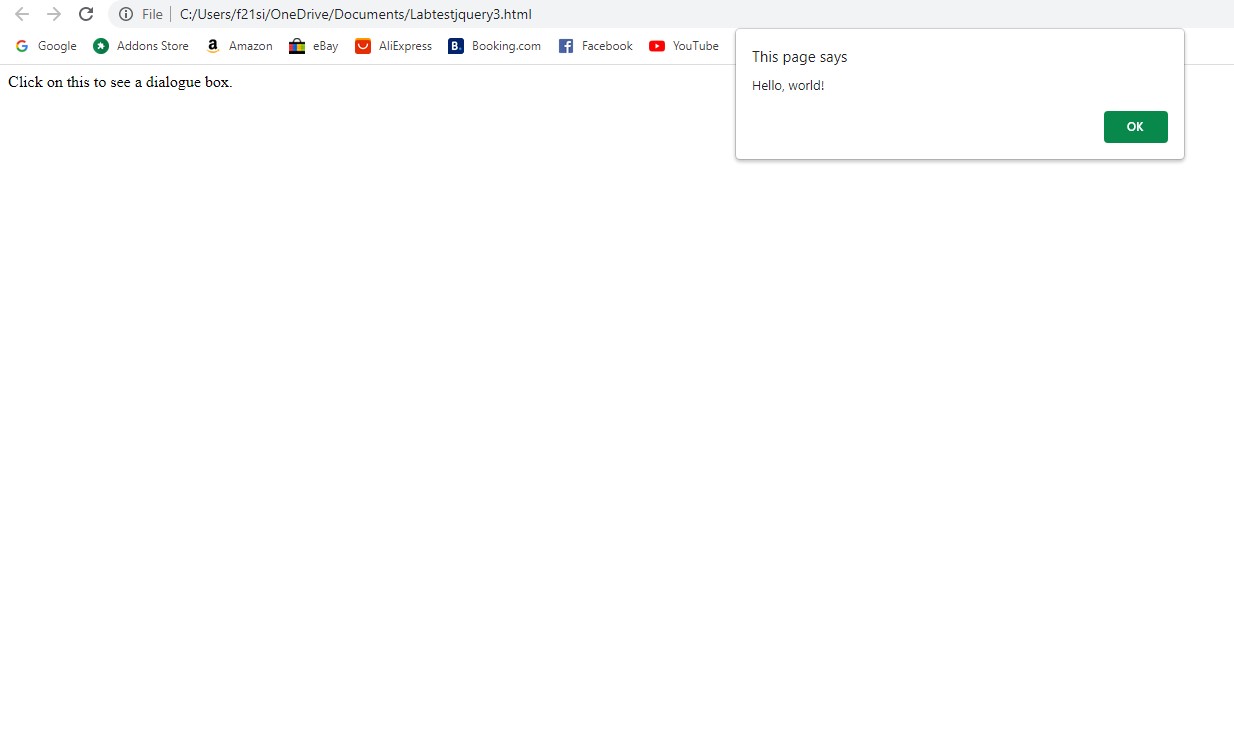
});

Now we can include **custom.js** file in our HTML file as follows –

Text

Description automatically generated

Screen shot the result you get:



Using Multiple Libraries

You can use multiple libraries all together without conflicting each others.

For example, you can use jQuery and MooTool javascript libraries together. You can check [jQuery noConflict](https://www.tutorialspoint.com/jquery/jquery-noconflict.htm) Method for more detail.

jQuery - Basics

jQuery is a framework built using JavaScript capabilities. So, you can use all the functions and other capabilities available in JavaScript. This chapter would explain most basic concepts but frequently used in jQuery.

String

A string in JavaScript is an immutable object that contains none, one or many characters. Following are the valid examples of a JavaScript String −

"This is JavaScript String"

'This is JavaScript String'

'This is "really" a JavaScript String'

"This is 'really' a JavaScript String"

Numbers

Numbers in JavaScript are double-precision 64-bit format IEEE 754 values. They are immutable, just as strings. Following are the valid examples of a JavaScript Numbers −

5350

120.27

0.26

Boolean

A boolean in JavaScript can be either **true** or **false**. If a number is zero, it defaults to false. If an empty string defaults to false.

Following are the valid examples of a JavaScript Boolean −

true // true

false // false

0 // false

1 // true

"" // false

"hello" // true

Objects

JavaScript supports Object concept very well. You can create an object using the object literal as follows −

var emp = {

name: "Zara",

age: 10

};

You can write and read properties of an object using the dot notation as follows −

// Getting object properties

emp.name // ==> Zara

emp.age // ==> 10

// Setting object properties

emp.name = "Daisy" // <== Daisy

emp.age = 20 // <== 20

Arrays

You can define arrays using the array literal as follows −

var x = [];

var y = [1, 2, 3, 4, 5];

An array has a **length** property that is useful for iteration −

var x =[1,2,3,4,5];

for(var i =0; i < x.length; i++){

// Do something with x[i]

}

Functions

A function in JavaScript can be either named or anonymous. A named function can be defined using *function* keyword as follows −

function named(){

// do some stuff here

}

An anonymous function can be defined in similar way as a normal function but it would not have any name.

A anonymous function can be assigned to a variable or passed to a method as shown below.

var handler =function(){

// do some stuff here

}

JQuery makes a use of anonymous functions very frequently as follows −

$(document).ready(function(){

// do some stuff here

});

Arguments

JavaScript variable *arguments* is a kind of array which has *length* property. Following example explains it very well −

function func(x){

console.log(typeof x, arguments.length);

}

func();//==> "undefined", 0

func(1);//==> "number", 1

func("1","2","3");//==> "string", 3

The arguments object also has a *callee* property, which refers to the function you're inside of. For example −

function func(){

return arguments.callee;

}

func();// ==> func

Context: all the variables that are visible – in scope – at a certain point in the flow of execution of your program.

JavaScript famous keyword **this** always refers to the current context.

Within a function **this** context can change, depending on how the function is called −

$(document).ready(function(){

// this refers to window.document

});

$("div").click(function(){

// this refers to a div DOM element

});

You can specify the context for a function call using the function-built-in methods **call()** and **apply()** methods.

The difference between them is how they pass arguments. Call passes all arguments through as arguments to the function, while apply accepts an array as the arguments.

function scope(){

console.log(this, arguments.length);

}

scope()// window, 0

scope.call("foobar",[1,2]);//==> "foobar", 1

scope.apply("foobar",[1,2]);//==> "foobar", 2

Scope

The scope of a variable is the region of your program in which it is defined. JavaScript variable will have only two scopes.

* **Global Variables** –Defined in your MODULE (file containing the JS code), but outside of any function.
* A global variable has global scope which means it is defined everywhere in your JavaScript code.
* **Local Variables** − A local variable will be visible only within a function where it is defined. Function parameters are always local to that function.

Within the body of a function, a local variable takes precedence over a global variable with the same name −

var myVar ="global";// ==> Declare a global variable

function(){

var myVar ="local";// ==> Declare a local variable

document.write(myVar);// ==> local

}

**Callback**

A callback is a plain JavaScript function passed to some method as an argument or option.

Some callbacks are just events, called to give the user a chance to react when a certain state is triggered.

jQuery's event system uses such callbacks everywhere for example −

$("body").click(function(event){

console.log("clicked: "+event.target);

});

Most callbacks provide arguments and a context.

In the event-handler example, the callback is called with one argument, **an Event**.

Some callbacks are required to return something, others make that return value optional. To prevent a form submission, a submit event handler can return false as follows −

$("#myform").submit(function() {

return false;

});

**Closures**

**Closures are created whenever a variable that is defined outside the current scope is accessed from within some inner scope.**

Following example shows how the variable **counter** is visible within the **create, increment, and print functions, but not outside of them** −

function create(){

var counter =0;

return{

increment:function(){

counter++;

},

print:function(){

console.log(counter);

}

}

}

var c = create();

c.increment();

c.print();// ==> 1

This pattern allows you to create objects with methods that operate on data that isn't visible to the outside world.

**It should be noted that data hiding is the very basis of object-oriented programming.**

**Proxy Pattern**

**A proxy is an object that can be used to control access to another object.**

*It implements the same interface as this other object and passes on any method invocations to* it. This other object is often called the **real subject**.

A proxy can be instantiated in place of this real subject and allow it to be accessed remotely.

We can save jQuery's setArray method in a closure and overwrites it as follows −

(function(){

// log all calls to setArray

var proxied = jQuery.fn.setArray;

jQuery.fn.setArray =function(){

console.log(this, arguments);

return proxied.apply(this, arguments);

};

})();

**The above wraps its code in a function to hide the *proxied* variable.**

The proxy then logs all calls to the method and **delegates** the call to the original method.

Using *apply(this, arguments)* guarantees that the caller won't be able to notice the difference between the **original and the proxied method.**

Built-in Functions

JavaScript comes along with a useful set of built-in functions. These methods can be used to manipulate Strings, Numbers and Dates.

Following are important JavaScript functions −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | **charAt()**  Returns the character at the specified index. |
| 2 | **concat()**  Combines the text of two strings and returns a new string. |
| 3 | **forEach()**  Calls a function for each element in the array. |
| 4 | **indexOf()**  Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found. |
| 5 | **length()**  Returns the length of the string. |
| 6 | **pop()**  Removes the last element from an array and returns that element. |
| 7 | **push()**  Adds one or more elements to the end of an array and returns the new length of the array. |
| 8 | **reverse()**  Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first. |
| 9 | **sort()**  Sorts the elements of an array. |
| 10 | **substr()**  Returns the characters in a string beginning at the specified location through the specified number of characters. |
| 11 | **toLowerCase()**  Returns the calling string value converted to lower case. |
| 12 | **toString()**  Returns the string representation of the number's value. |
| 13 | **toUpperCase()**  Returns the calling string value converted to uppercase. |

**The Document Object Model**

The Document Object Model is a tree structure of various elements of HTML as follows −

[Live Demo](http://tpcg.io/s1QZ3X)

<html>

<head>

<title>The jQuery Example</title>

</head>

<body>

<div>

<p>This is a paragraph.</p>

<p>This is second paragraph.</p>

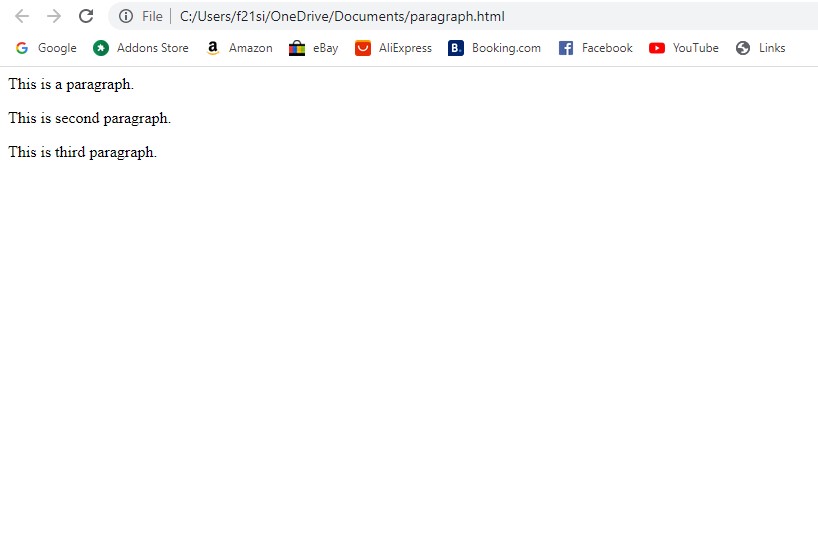
<p>This is third paragraph.</p>

</div>

</body>

</html>

This will produce following result –



Following are the important points about the above tree structure −

* The <html> is the ancestor of all the other elements; in other words, all the other elements are descendants of <html>.
* The <head> and <body> elements are not only descendants, but children of <html>, as well.
* Likewise, in addition to being the ancestor of <head> and <body>, <html> is also their parent.
* The <p> elements are children (and descendants) of <div>, descendants of <body> and <html>, and siblings of each other <p> elements.

While learning jQuery concepts, it will be helpful to have understanding on DOM, if you are not aware of DOM this is a good time to review foundational topics on that subject!

The jQuery library harnesses the power of Cascading Style Sheets **(CSS) selectors**to let us quickly and easily access elements or groups of elements in the Document Object Model (DOM).

A jQuery Selector is a function which makes use of expressions to find out matching elements from a DOM based on the given criteria.

Simply you can say, selectors are used to select one or more HTML elements using jQuery.

Once an element is selected then we can perform various operations on that selected element.

## The $() factory function:

## A factor is a method that programmatically creates OBJECTS.

jQuery selectors start with the dollar sign and parentheses − **$()**.The factory function **$()** makes use of following **three building blocks while selecting elements in a given document** −

|  |  |
| --- | --- |
| **Sr.No.** | **Selector & Description** |
| 1 | **Tag Name**  Represents a tag name available in the DOM. For example **$('p')** selects all paragraphs <p> in the document. |
| 2 | **Tag ID**  Represents a tag available with the given ID in the DOM. For example **$('#some-id')** selects the single element in the document that has an ID of some-id. |
| 3 | **Tag Class**  Represents a tag available with the given class in the DOM. For example **$('.some-class')** selects all elements in the document that have a class of some-class. |

All the above items can be used either on their own or in combination with other selectors.

All the jQuery selectors are based on the same principle except some tweaking.

**NOTE** − The factory function **$()** is a synonym of **jQuery()** function. So in case you are using any other JavaScript library where **$** sign is conflicting with some thing else then you can replace **$** sign by **jQuery** name and you can use function **jQuery()** instead of **$()**.

### Example

Following is a simple example which makes use of Tag Selector. This would select all the elements with a tag name **p**.

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

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

});

</script>

</head>

<body>

<div>

<pclass="myclass">This is a paragraph.</p>

<pid="myid">This is second paragraph.</p>

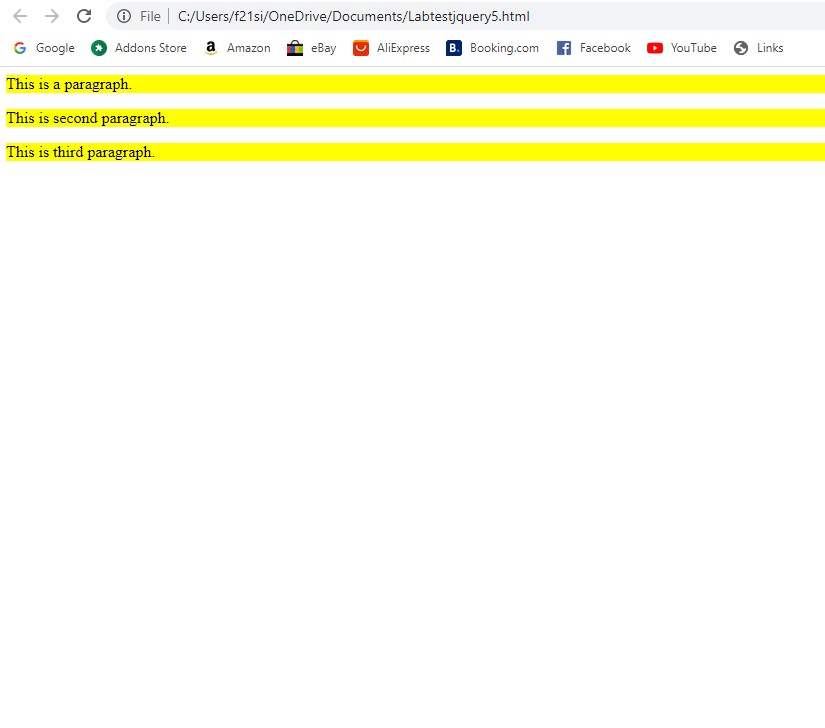
<p>This is third paragraph.</p>

</div>

</body>

</html>

This will produce following result –



## How to Use Selectors?

The selectors are very useful and would be required at every step while using jQuery.

They get the exact element that you want from your HTML document.

Following table lists down few basic selectors and explains them with examples.

|  |  |
| --- | --- |
| **Sr.No.** | **Selector & Description** |
| 1 | [Name](https://www.tutorialspoint.com/jquery/selector-element-name.htm)  Selects all elements which match with the given element **Name**. |
| 2 | [#ID](https://www.tutorialspoint.com/jquery/selector-element-id.htm)  Selects a single element which matches with the given **ID**. |
| 3 | [.Class](https://www.tutorialspoint.com/jquery/selector-element-class.htm)  Selects all elements which match with the given **Class**. |
| 4 | [Universal (\*)](https://www.tutorialspoint.com/jquery/selector-universal.htm)  Selects all elements available in a DOM. |
| 5 | [Multiple Elements E, F, G](https://www.tutorialspoint.com/jquery/selector-multiple-elements.htm)  Selects the combined results of all the specified selectors **E, F** or **G**. |

## Selectors Examples

Similar to above syntax and examples, following examples would give you understanding on using different type of other useful selectors −

Here, you have different type of other useful selectors −

|  |  |
| --- | --- |
| **Sr.No.** | **Selector & Description** |
| 1 | **$('\*')**  This selector selects all elements in the document. |
| 2 | **$("p > \*")**  This selector selects all elements that are children of a paragraph element. |
| 3 | **$("#specialID")**  This selector function gets the element with id="specialID". |
| 4 | **$(".specialClass")**  This selector gets all the elements that have the class of *specialClass*. |
| 5 | **$("li:not(.myclass)")**  Selects all elements matched by <li> that do not have class = "myclass". |
| 6 | **$("a#specialID.specialClass")**  This selector matches links with an id of *specialID* and a class of *specialClass*. |
| 7 | **$("p a.specialClass")**  This selector matches links with a class of *specialClass* declared within <p> elements. |
| 8 | **$("ul li:first")**  This selector gets only the first <li> element of the <ul>. |
| 9 | **$("#container p")**  Selects all elements matched by <p> that are descendants of an element that has an id of *container*. |
| 10 | **$("li > ul")**  Selects all elements matched by <ul> that are children of an element matched by <li> |
| 11 | **$("strong + em")**  Selects all elements matched by <em> that immediately follow a sibling element matched by <strong>. |
| 12 | **$("p ~ ul")**  Selects all elements matched by <ul> that follow a sibling element matched by <p>. |
| 13 | **$("code, em, strong")**  Selects all elements matched by <code> or <em> or <strong>. |
| 14 | **$("p strong, .myclass")**  Selects all elements matched by <strong> that are descendants of an element matched by <p> as well as all elements that have a class of *myclass*. |
| 15 | **$(":empty")**  Selects all elements that have no children. |
| 16 | **$("p:empty")**  Selects all elements matched by <p> that have no children. |
| 17 | **$("div[p]")**  Selects all elements matched by <div> that contain an element matched by <p>. |
| 18 | **$("p[.myclass]")**  Selects all elements matched by <p> that contain an element with a class of *myclass*. |
| 19 | **$("a[@rel]")**  Selects all elements matched by <a> that have a rel attribute. |
| 20 | **$("input[@name = myname]")**  Selects all elements matched by <input> that have a name value exactly equal to *myname.* |
| 21 | **$("input[@name^=myname]")**  Selects all elements matched by <input> that have a name value beginning with *myname*. |
| 22 | **$("a[@rel$=self]")**  Selects all elements matched by <a> that have **rel** attribute value ending with *self*. |
| 23 | **$("a[@href\*=domain.com]")**  Selects all elements matched by <a> that have an href value containing domain.com. |
| 24 | **$("li:even")**  Selects all elements matched by <li> that have an even index value. |
| 25 | **$("tr:odd")**  Selects all elements matched by <tr> that have an odd index value. |
| 26 | **$("li:first")**  Selects the first <li> element. |
| 27 | **$("li:last")**  Selects the last <li> element. |
| 28 | **$("li:visible")**  Selects all elements matched by <li> that are visible. |
| 29 | **$("li:hidden")**  Selects all elements matched by <li> that are hidden. |
| 30 | **$(":radio")**  Selects all radio buttons in the form. |
| 31 | **$(":checked")**  Selects all checked box in the form. |
| 32 | **$(":input")**  Selects only form elements (input, select, textarea, button). |
| 33 | **$(":text")**  Selects only text elements (input[type = text]). |
| 34 | **$("li:eq(2)")**  Selects the third <li> element. |
| 35 | **$("li:eq(4)")**  Selects the fifth <li> element. |
| 36 | **$("li:lt(2)")**  Selects all elements matched by <li> element before the third one; in other words, the first two <li> elements. |
| 37 | **$("p:lt(3)")**  selects all elements matched by <p> elements before the fourth one; in other words the first three <p> elements. |
| 38 | **$("li:gt(1)")**  Selects all elements matched by <li> after the second one. |
| 39 | **$("p:gt(2)")**  Selects all elements matched by <p> after the third one. |
| 40 | **$("div/p")**  Selects all elements matched by <p> that are children of an element matched by <div>. |
| 41 | **$("div//code")**  Selects all elements matched by <code>that are descendants of an element matched by <div>. |
| 42 | **$("//p//a")**  Selects all elements matched by <a> that are descendants of an element matched by <p> |
| 43 | **$("li:first-child")**  Selects all elements matched by <li> that are the first child of their parent. |
| 44 | **$("li:last-child")**  Selects all elements matched by <li> that are the last child of their parent. |
| 45 | **$(":parent")**  Selects all elements that are the parent of another element, including text. |
| 46 | **$("li:contains(second)")**  Selects all elements matched by <li> that contain the text second. |

You can use all the above selectors with any HTML/XML element in generic way.

For example if selector **$("li:first")** works for <li> element then **$("p:first")** would also work for <p> element.

Some of the most basic components we can manipulate when it comes to DOM elements are the properties and attributes assigned to those elements.

Most of these attributes are available through JavaScript as DOM node properties. Some of the more common properties are −

* className
* tagName
* id
* href
* title
* rel
* src

Consider the following HTML markup for an image element −

<img id = "imageid" src = "image.gif" alt = "Image" class = "myclass"

title = "This is an image"/>

In this element's markup, the tag name is img, and the markup for id, src, alt, class, and title represents the element's attributes, each of which consists of a name and a value.

jQuery gives us the means to easily manipulate an element's attributes and gives us access to the element so that we can also change its properties.

Get Attribute Value

The **attr()** method can be used to either fetch the value of an attribute from the first element in the matched set or set attribute values onto all matched elements.

Example

Following is a simple example which fetches title attribute of <em> tag and set <div id = "divid"> value with the same value −

[Live Demo](http://tpcg.io/h16fH2)

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

var title = $("em").attr("title");

$("#divid").text(title);

});

</script>

</head>

<body>

<div>

<emtitle="Bold and Brave">This is first paragraph.</em>

<pid="myid">This is second paragraph.</p>

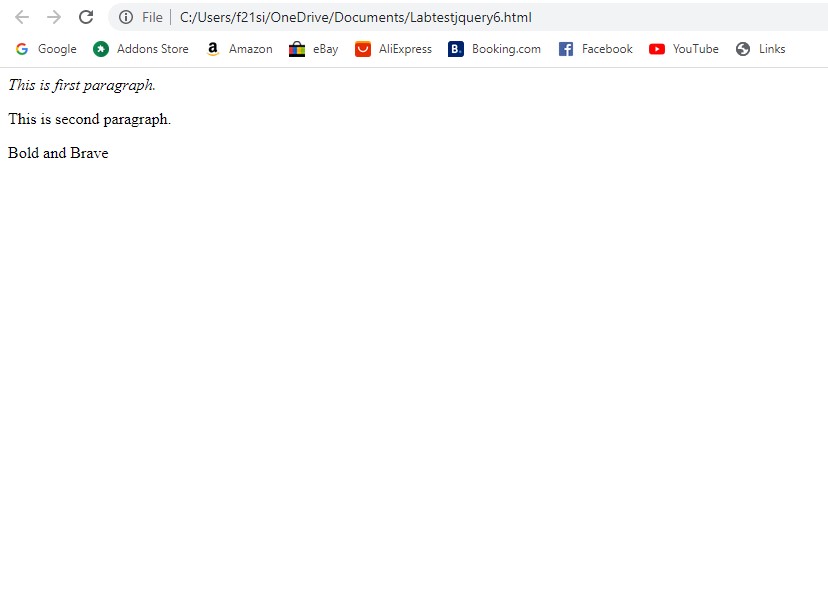
<divid="divid"></div>

</div>

</body>

</html>

This will produce following result –



Set Attribute Value

The **attr(name, value)** method can be used to set the named attribute onto all elements in the wrapped set using the passed value.

Example

Following is a simple example which set **src** attribute of an image tag to a correct location −

[Live Demo](http://tpcg.io/944kkG)

<html>

<head>

<title>The jQuery Example</title>

<basehref="http://www.google.com"/>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("#myimg").attr("src","/jquery/images/jquery.jpg");

});

</script>

</head>

<body>

<div>

<imgid="myimg"src="/images/jquery.jpg"alt="Sample image"/>

</div>

</body>

</html>

Applying CSS Stylesheets

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.

Example

Following is a simple example which sets **class** attribute of a para <p> tag −

Resume here...

<html>

<head>

<title>The jQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("em").addClass("selected");

$("#myid").addClass("highlight");

});

</script>

<style>

.selected { color:red;}

.highlight { background:yellow;}

</style>

</head>

<body>

<emtitle="Bold and Brave">This is first paragraph.</em>

<pid="myid">This is second paragraph.</p>

</body>

</html>

Attribute Methods

Following table lists down few useful methods which you can use to manipulate attributes and properties −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | [attr( properties )](https://www.tutorialspoint.com/jquery/attr-properties.htm)  Set a key/value object as properties to all matched elements. |
| 2 | [attr( key, fn )](https://www.tutorialspoint.com/jquery/attr-key-function.htm)  Set a single property to a computed value, on all matched elements. |
| 3 | [removeAttr( name )](https://www.tutorialspoint.com/jquery/attr-remove-attribute.htm)  Remove an attribute from each of the matched elements. |
| 4 | [hasClass( class )](https://www.tutorialspoint.com/jquery/attr-has-class.htm)  Returns true if the specified class is present on at least one of the set of matched elements. |
| 5 | [removeClass( class )](https://www.tutorialspoint.com/jquery/attr-remove-class.htm)  Removes all or the specified class(es) from the set of matched elements. |
| 6 | [toggleClass( class )](https://www.tutorialspoint.com/jquery/attr-toggle-class.htm)  Adds the specified class if it is not present, removes the specified class if it is present. |
| 7 | [html( )](https://www.tutorialspoint.com/jquery/attr-html.htm)  Get the html contents (innerHTML) of the first matched element. |
| 8 | [html( val )](https://www.tutorialspoint.com/jquery/attr-html-val.htm)  Set the html contents of every matched element. |
| 9 | [text( )](https://www.tutorialspoint.com/jquery/attr-text.htm)  Get the combined text contents of all matched elements. |
| 10 | [text( val )](https://www.tutorialspoint.com/jquery/attr-text-val.htm)  Set the text contents of all matched elements. |
| 11 | [val( )](https://www.tutorialspoint.com/jquery/attr-val.htm)  Get the input value of the first matched element. |
| 12 | [val( val )](https://www.tutorialspoint.com/jquery/attr-val-val.htm)  Set the value attribute of every matched element if it is called on <input> but if it is called on <select> with the passed <option> value then passed option would be selected, if it is called on check box or radio box then all the matching check box and radiobox would be checked. |

Examples

Similar to above syntax and examples, following examples would give you understanding on using various attribute methods in different situation −

Here is a complete list of attribute methods in different situation −

|  |  |
| --- | --- |
| **Sr.No.** | **Selector & Description** |
| 1 | **$("#myID").attr("custom")**  This would return value of attribute *custom* for the first element matching with ID myID. |
| 2 | **$("img").attr("alt", "Sample Image")**  This sets the **alt** attribute of all the images to a new value "Sample Image". |
| 3 | **$("input").attr({ value: "", title: "Please enter a value" });**  Sets the value of all <input> elements to the empty string, as well as sets The jQuery Example to the string *Please enter a value*. |
| 4 | **$("a[href^=https://]").attr("target","\_blank")**  Selects all links with an href attribute starting with *https://* and set its target attribute to *\_blank*. |
| 5 | **$("a").removeAttr("target")**  This would remove *target* attribute of all the links. |
| 6 | **$("form").submit(function() {$(":submit",this).attr("disabled", "disabled");});**  This would modify the disabled attribute to the value "disabled" while clicking Submit button. |
| 7 | **$("p:last").hasClass("selected")**  This return true if last <p> tag has associated class*selected*. |
| 8 | **$("p").text()**  Returns string that contains the combined text contents of all matched <p> elements. |
| 9 | **$("p").text("<i>Hello World</i>")**  This would set "<I>Hello World</I>" as text content of the matching <p> elements. |
| 10 | **$("p").html()**  This returns the HTML content of the all matching paragraphs. |
| 11 | **$("div").html("Hello World")**  This would set the HTML content of all matching <div> to *Hello World*. |
| 12 | **$("input:checkbox:checked").val()**  Get the first value from a checked checkbox. |
| 13 | **$("input:radio[name=bar]:checked").val()**  Get the first value from a set of radio buttons. |
| 14 | **$("button").val("Hello")**  Sets the value attribute of every matched element <button>. |
| 15 | **$("input").val("on")**  This would check all the radio or check box button whose value is "on". |
| 16 | **$("select").val("Orange")**  This would select Orange option in a dropdown box with options Orange, Mango and Banana. |
| 17 | **$("select").val("Orange", "Mango")**  This would select Orange and Mango options in a dropdown box with options Orange, Mango and Banana. |

jQuery is a very powerful tool which 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**

Consider a simple document with the following HTML content

<html>

<head>

<title>The JQuery Example</title>

</head>

<body>

<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>

</body>

</html>

This will produce following result



* Above every list has its own index, and can be located directly by using **eq(index)** method as below example.
* Every child element starts its index from zero, thus, *list item 2* would be accessed by using **$("li").eq(1)** and so on.

Example

Following is a simple example which adds the color to second list item.

<html>

<head>

<title>The JQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("li").eq(2).addClass("selected");

});

</script>

<style>

.selected { color:red;}

</style>

</head>

<body>

<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>

</body>

</html>

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(s). The *selector* can be written using any selector syntax.

Example

Following is a simple example which applies color to the lists associated with middle class −

[Live Demo](http://tpcg.io/9KunND)

<html>

<head>

<title>The JQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("li").filter(".middle").addClass("selected");

});

</script>

<style>

.selected { color:red;}

</style>

</head>

<body>

<div>

<ul>

<liclass="top">list item 1</li>

<liclass="top">list item 2</li>

<liclass="middle">list item 3</li>

<liclass="middle">list item 4</li>

<liclass="bottom">list item 5</li>

<liclass="bottom">list item 6</li>

</ul>

</div>

</body>

</html>

Locating Descendant Elements

The **find( selector )** method can be used to locate all the descendant elements of a particular type of elements. The *selector* can be written using any selector syntax.

Example

Following is an example which selects all the <span> elements available inside different <p> elements −

[Live Demo](http://tpcg.io/HwIVAB)

<html>

<head>

<title>The JQuery Example</title>

<scripttype="text/javascript"

src="https://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">

</script>

<scripttype="text/javascript"language="javascript">

$(document).ready(function(){

$("p").find("span").addClass("selected");

});

</script>

<style>

.selected { color:red;}

</style>

</head>

<body>

<p>This is 1st paragraph and <span>THIS IS RED</span></p>

<p>This is 2nd paragraph and <span>THIS IS ALSO RED</span></p>

</body>

</html>

## JQuery DOM Filter Methods

Following table lists down useful methods which you can use to filter out various elements from a list of DOM elements −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [eq( index )](https://www.tutorialspoint.com/jquery/traversal-eq.htm)  Reduce the set of matched elements to a single element. |
| 2 | [filter( selector )](https://www.tutorialspoint.com/jquery/traversal-filter.htm)  Removes all elements from the set of matched elements that do not match the specified selector(s). |
| 3 | [filter( fn )](https://www.tutorialspoint.com/jquery/traversal-filter-fn.htm)  Removes all elements from the set of matched elements that do not match the specified function. |
| 4 | [is( selector )](https://www.tutorialspoint.com/jquery/traversal-is.htm)  Checks the current selection against an expression and returns true, if at least one element of the selection fits the given selector. |
| 5 | [map( callback )](https://www.tutorialspoint.com/jquery/traversal-map.htm)  Translate a set of elements in the jQuery object into another set of values in a jQuery array (which may, or may not contain elements). |
| 6 | [not( selector )](https://www.tutorialspoint.com/jquery/traversal-not.htm)  Removes elements matching the specified selector from the set of matched elements. |
| 7 | [slice( start, [end] )](https://www.tutorialspoint.com/jquery/traversal-slice.htm)  Selects a subset of the matched elements. |

JQuery DOM Traversing Methods

Following table lists down other useful methods which you can use to locate various elements in a DOM −

|  |  |
| --- | --- |
| **Sr.No.** | **Methods & Description** |
| 1 | [add( selector )](https://www.tutorialspoint.com/jquery/traversal-add.htm)  Adds more elements, matched by the given selector, to the set of matched elements. |
| 2 | [andSelf( )](https://www.tutorialspoint.com/jquery/traversal-andself.htm)  Add the previous selection to the current selection. |
| 3 | [children( [selector])](https://www.tutorialspoint.com/jquery/traversal-children.htm)  Get a set of elements containing all of the unique immediate children of each of the matched set of elements. |
| 4 | [closest( selector )](https://www.tutorialspoint.com/jquery/traversal-closest.htm)  Get a set of elements containing the closest parent element that matches the specified selector, the starting element included. |
| 5 | [contents( )](https://www.tutorialspoint.com/jquery/traversal-contents.htm)  Find all the child nodes inside the matched elements (including text nodes), or the content document, if the element is an iframe. |
| 6 | [end( )](https://www.tutorialspoint.com/jquery/traversal-end.htm)  Revert the most recent 'destructive' operation, changing the set of matched elements to its previous state. |
| 7 | [find( selector )](https://www.tutorialspoint.com/jquery/traversal-find.htm)  Searches for descendant elements that match the specified selectors. |
| 8 | [next( [selector] )](https://www.tutorialspoint.com/jquery/traversal-next.htm)  Get a set of elements containing the unique next siblings of each of the given set of elements. |
| 9 | [nextAll( [selector] )](https://www.tutorialspoint.com/jquery/traversal-nextall.htm)  Find all sibling elements after the current element. |
| 10 | [offsetParent( )](https://www.tutorialspoint.com/jquery/traversal-offsetparent.htm)  Returns a jQuery collection with the positioned parent of the first matched element. |
| 11 | [parent( [selector] )](https://www.tutorialspoint.com/jquery/traversal-parent.htm)  Get the direct parent of an element. If called on a set of elements, parent returns a set of their unique direct parent elements. |
| 12 | [parents( [selector] )](https://www.tutorialspoint.com/jquery/traversal-parents.htm)  Get a set of elements containing the unique ancestors of the matched set of elements (except for the root element). |
| 13 | [prev( [selector] )](https://www.tutorialspoint.com/jquery/traversal-prev.htm)  Get a set of elements containing the unique previous siblings of each of the matched set of elements. |
| 14 | [prevAll( [selector] )](https://www.tutorialspoint.com/jquery/traversal-prevall.htm)  Find all sibling elements in front of the current element. |
| 15 | [siblings( [selector] )](https://www.tutorialspoint.com/jquery/traversal-siblings.htm)  Get a set of elements containing all of the unique siblings of each of the matched set of elements. |

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.

Using JQuery library developers can enhance their websites without worrying about browsers and their versions as long as the browsers have JavaScript enabled.

Most of the JQuery CSS Methods do not modify the content of the jQuery object and they are used to apply CSS properties on DOM elements.

Apply CSS Properties

This is very simple to apply any CSS property using JQuery method **css( PropertyName, PropertyValue )**.

Here is the syntax for the method −

**selector**.css( PropertyName, PropertyValue );

Here you can pass *PropertyName* as a javascript string and based on its value, *PropertyValue* could be string or integer.

### Example

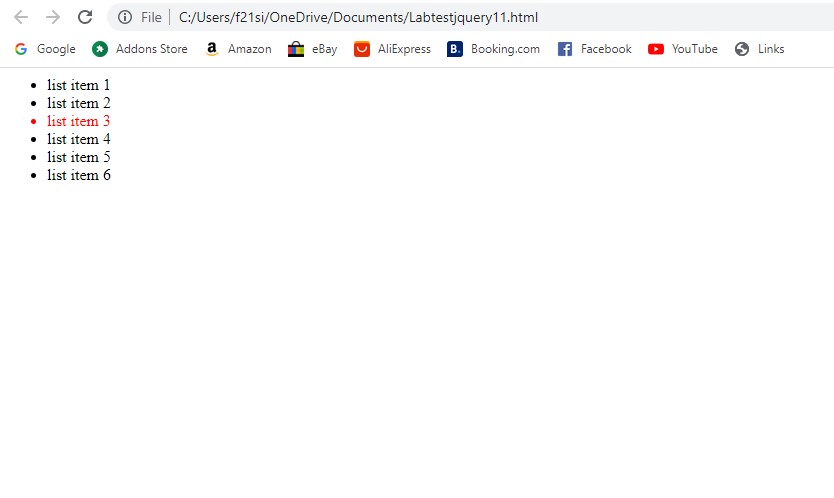
Following is an example which adds font color to the second list item.

Enter this code:

Text

Description automatically generated

**Screen shot your result:**

****

## Apply Multiple CSS Properties

You can apply multiple CSS properties using a single JQuery method **CSS( {key1:val1, key2:val2....)**. You can apply as many properties as you like in a single call.

Here is the syntax for the method −

**selector**.css( {key1:val1, key2:val2....keyN:valN})

Here you can pass key as property and val as its value as described above.

### Example

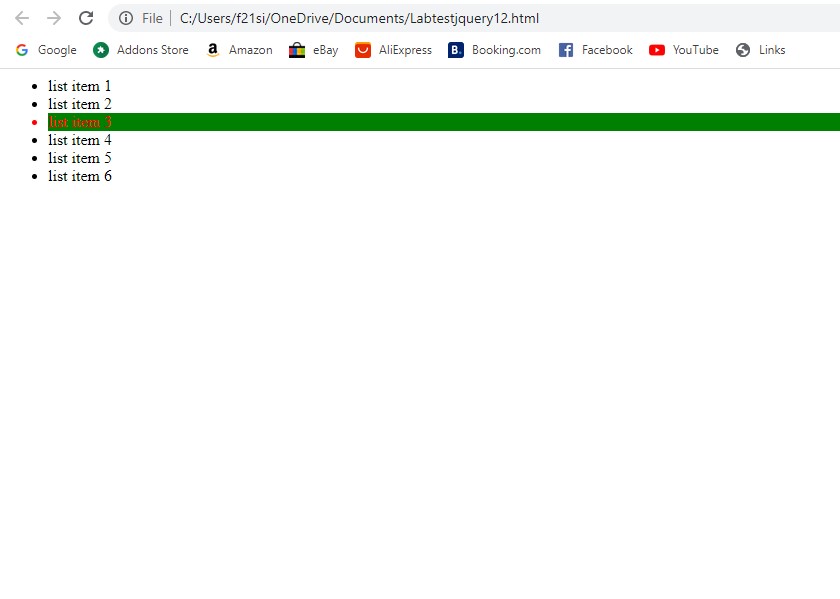
Following is an example which adds font color as well as background color to the second list item.

Enter this code:

Text

Description automatically generated

Screen shot your result –



**Setting Element Width & Height**

The **width( val )** and **height( val )** method can be used to set the width and height respectively of any element.

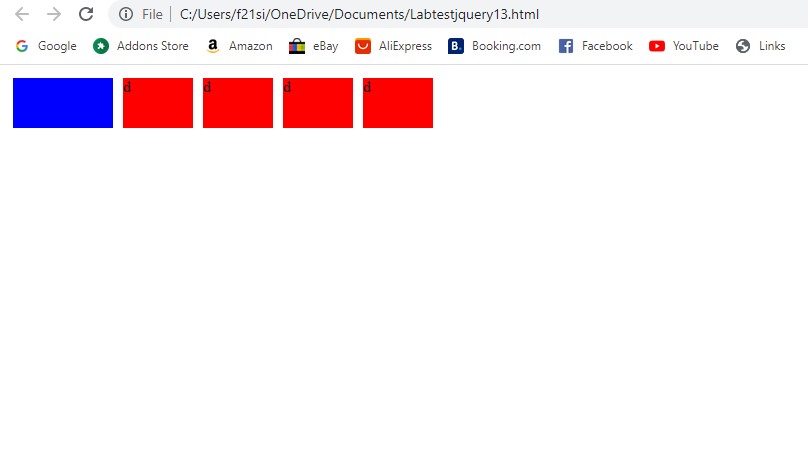
### Example

Following is a simple example which sets the width of first division element where as rest of the elements have width set by style sheet

Enter this code:Text

Description automatically generated

Screen shot your result –



## JQuery CSS Methods

Following table lists down all the methods which you can use to play with CSS properties −

|  |  |
| --- | --- |
| **Sr.No.** | **Method & Description** |
| 1 | [css( name )](https://www.tutorialspoint.com/jquery/css-name.htm)  Return a style property on the first matched element. |
| 2 | [css( name, value )](https://www.tutorialspoint.com/jquery/css-name-value.htm)  Set a single style property to a value on all matched elements. |
| 3 | [css( properties )](https://www.tutorialspoint.com/jquery/css-properties.htm)  Set a key/value object as style properties to all matched elements. |
| 4 | [height( val )](https://www.tutorialspoint.com/jquery/css-height-val.htm)  Set the CSS height of every matched element. |
| 5 | [height( )](https://www.tutorialspoint.com/jquery/css-height.htm)  Get the current computed, pixel, height of the first matched element. |
| 6 | [innerHeight( )](https://www.tutorialspoint.com/jquery/css-innerheight.htm)  Gets the inner height (excludes the border and includes the padding) for the first matched element. |
| 7 | [innerWidth( )](https://www.tutorialspoint.com/jquery/css-innerwidth.htm)  Gets the inner width (excludes the border and includes the padding) for the first matched element. |
| 8 | [offset( )](https://www.tutorialspoint.com/jquery/css-offset.htm)  Get the current offset of the first matched element, in pixels, relative to the document. |
| 9 | [offsetParent( )](https://www.tutorialspoint.com/jquery/css-offsetparent.htm)  Returns a jQuery collection with the positioned parent of the first matched element. |
| 10 | [outerHeight( [margin] )](https://www.tutorialspoint.com/jquery/css-outerheight.htm)  Gets the outer height (includes the border and padding by default) for the first matched element. |
| 11 | [outerWidth( [margin] )](https://www.tutorialspoint.com/jquery/css-outerwidth.htm)  Get the outer width (includes the border and padding by default) for the first matched element. |
| 12 | [position( )](https://www.tutorialspoint.com/jquery/css-position.htm)  Gets the top and left position of an element relative to its offset parent. |
| 13 | [scrollLeft( val )](https://www.tutorialspoint.com/jquery/css-scrollleft-val.htm)  When a value is passed in, the scroll left offset is set to that value on all matched elements. |
| 14 | [scrollLeft( )](https://www.tutorialspoint.com/jquery/css-scrollleft.htm)  Gets the scroll left offset of the first matched element. |
| 15 | [scrollTop( val )](https://www.tutorialspoint.com/jquery/css-scrolltop-val.htm)  When a value is passed in, the scroll top offset is set to that value on all matched elements. |
| 16 | [scrollTop( )](https://www.tutorialspoint.com/jquery/css-scrolltop.htm)  Gets the scroll top offset of the first matched element. |
| 17 | [width( val )](https://www.tutorialspoint.com/jquery/css-width-val.htm)  Set the CSS width of every matched element. |
| 18 | [width( )](https://www.tutorialspoint.com/jquery/css-width.htm)  Get the current computed, pixel, width of the first matched element. |