JQUERY

web application library

tutorialspoint

SIMPLYEASYLEARNING

www.tutorialspoint.com

https://www.facebook.com/tutorialspointindia

https://twitter.com/tutorialspoint

About the Tutorial

jQuery is a fast and concise JavaScript library created by John Resig in 2006. jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for Rapid Web Development.

Audience

This tutorial is designed for software programmers who wants to learn the basics of jQuery and its programming concepts in simple and easy ways. This tutorial will give you enough understanding on components of jQuery with suitable examples.

Prerequisites

Before proceeding with this tutorial, you should have a basic understanding of HTML, CSS, JavaScript, Document Object Model (DOM) and any text editor. As we are going to develop web based application using jQuery, it will be good if you have understanding on how internet and web based applications work

Copyright & Disclaimer

Table of Contents

1. OVERVIEW

What is jQuery?

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 traversing, event handling, animating, and Ajax interactions for rapid web development. jQuery is a JavaScript toolkit designed to simplify various tasks by writing less code. Here is the list of important core features supported by jQuery:

- **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 feature-rich 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+
- **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, insert 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:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"  src="/jquery/jquery-2.1.3.min.js"></script>

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

   </head>

   <body>

      <h1>Hello</h1>

   </body>
</html>
```

This will produce the following result –

```
Hello, World!
```

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.

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>

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

   <body>

      <h1>Hello</h1>

   </body>
</html>
```

This will produce the following result:

```
Hello, World!
```

How to Call a 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:

```
<html>
<head>
<title>The jQuery Example</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>

   <script type="text/javascript" language="javascript">
   // <![CDATA[
   $(document).ready(function() {
      $("div").click(function() {
        alert("Hello world!");
      });
   });
   // ]]>
   </script>

</head>
<body>
<div id="newdiv">
Click on this to see a dialogue box.
</div>
</body>
</html>
```

This will produce the following result:

```
Click on this to see a dialogue box.
```

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:

```
<html>
<head>
<title>The jQuery Example</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript"
   src="/jquery/custom.js"></script>
</head>
<body>
<div id="newdiv">
Click on this to see a dialogue box.
</div>
</body>
</html>
```

This will produce the following result:

```
Click on this to see a dialogue box.
```

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 Method for more detail.

jQuery noConflict() Method

Many JavaScript libraries use $ as a function or variable name, just as jQuery does. In jQuery's case, $ is just an alias for jQuery, so all the functionality is available without using $.

Run **$.noConflict()** method to give control of the $ variable back to whichever library first implemented it. This helps us to make sure that jQuery doesn't conflict with the $ object of other libraries.

Here is a simple way of avoiding any conflict:

```
// Import other Library
```

```
// Import jQuery Library
$.noConflict();
// Code that uses other library's $ can follow here.
```

This technique is especially effective in conjunction with the .ready() method's ability to alias the jQuery object, as within the .ready() we can use $ if we wish without fear of conflicts later:

```
// Import other library
// Import jQuery
$.noConflict();
jQuery(document).ready(function($) {
   // Code that uses jQuery's $ can follow here.
});
// Code that uses other library's $ can follow here.
```

What is Next ?

Do not worry too much if you did not understand the above examples. You are going to grasp them very soon in subsequent chapters. In the next chapter, we would try to cover few basic concepts which are coming from conventional JavaScript.

2. 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 there is an empty string, it 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. An 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 shows 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. For example:

```
function func() {
   return arguments.callee;
}
func();                 // ==> func
```

Context

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:** 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 saves 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 the important JavaScript functions:

Method	Description
charAt()	Returns the character at the specified index.
concat()	Combines the text of two strings and returns a new string.
forEach()	Calls a function for each element in the array.
indexOf()	Returns the index within the calling String object of the first occurrence of the specified value, or -1 if not found.
length()	Returns the length of the string.

pop()	Removes the last element from an array and returns that element.
push()	Adds one or more elements to the end of an array and returns the new length of the array.
reverse()	Reverses the order of the elements of an array -- the first becomes the last, and the last becomes the first.
sort()	Sorts the elements of an array.
substr()	Returns the characters in a string beginning at the specified location through the specified number of characters.
toLowerCase()	Returns the calling string value converted to lower case.
toString()	Returns the string representation of the number's value.
toUpperCase()	Returns the calling string value converted to uppercase.

A complete list of **JavaScript** built-in function is available here – Built-in Functions.

The Document Object Model

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

```
<html>
<head>
   <title>the title</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 the following result:

```
This is a paragraph.

This is second paragraph.

This is third paragraph
```

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 , 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, then I would suggest you to go through our simple tutorial on DOM Tutorial.

3. SELECTORS

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.

The $() Factory Function

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

S.N.	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 title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>

   <script type="text/javascript" language="javascript">
   $(document).ready(function() {
      var pars = $("p");
      for( i=0; i<pars.length; i++ ){
         alert("Found paragraph: " + pars[i].innerHTML);
      }
   });
   </script>
</head>
<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>
</html>
```

This will produce the the following result:

This is a paragraph.

This is second paragraph.

This is third paragraph.

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.

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

jQuery - Element Name Selector

Description

The element selector selects all the elements that have a tag name of T.

Syntax

Here is the simple syntax to use this selector –

```
$('tagname')
```

Parameters

Here is the description of all the parameters used by this selector –

- **tagname –** Any standard HTML tag name like div, p, em, img, li etc.

Returns

Like any other jQuery selector, this selector also returns an array filled with the found elements.

Example

- **$('p')** – Selects all elements with a tag name of **p** in the document.
- **$('div')** – Selects all elements with a tag name of **div** in the document.

Following example would select all the divisions and will apply yellow color to their background –

```
<html>
   <head>
      <title>The Selecter Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            /* This would select all the divisions */
            $("div").css("background-color", "yellow");
         });
      </script>
   </head>

   <body>

      <div class="big" id="div1">
         <p>This is first division of the DOM.</p>
      </div>

      <div class="medium" id="div2">
         <p>This is second division of the DOM.</p>
      </div>

      <div class="small" id="div3">
         <p>This is third division of the DOM</p>
       </div>
```

```
   </body>

</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

jQuery - Element ID Selector

Description

The element ID selector selects a single element with the given id attribute.

Syntax

Here is the simple syntax to use this selector –

```
$('#elementid')
```

Parameters

Here is the description of all the parameters used by this selector –

- **Elementid:** This would be an element ID. If the id contains any special characters like periods or colons you have to escape those characters with backslashes.

Returns

Like any other jQuery selector, this selector also returns an array filled with the found element.

Example

- **$('#myid')** – Selects a single element with the given id myid.
- **$('div#yourid')** – Selects a single division with the given id yourid.

Following example would select second division and will apply yellow color to its background as below:

```
<html>
   <head>
      <title>The Selecter Example</title>
```

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

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            /* This would select second division only*/
            $("#div2").css("background-color", "yellow");
         });
      </script>

   </head>

   <body>

      <div class="big" id="div1">
         <p>This is first division of the DOM.</p>
      </div>

      <div class="medium" id="div2">
         <p>This is second division of the DOM.</p>
      </div>

      <div class="small" id="div3">
         <p>This is third division of the DOM</p>
      </div>

   </body>

</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

jQuery - Element Class Selector

Description

The element class selector selects all the elements which match with the given class of the elements.

Syntax

Here is the simple syntax to use this selector:

```
$('.classid')
```

Parameters

Here is the description of all the parameters used by this selector –

- **classid –** This is class ID available in the document.

Returns

Like any other jQuery selector, this selector also returns an array filled with the found elements.

Example

- **$('.big')** – Selects all the elements with the given class ID **big**.
- **$('p.small')** – Selects all the paragraphs with the given class ID **small**.
- **$('.big.small')** – Selects all the elements with a class of **big** and **small**.

Following example would select all divisions with class .big and will apply yellow color to its background

```
<html>
   <head>
      <title>The Selecter Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            /* This would select second division only*/
            $(".big").css("background-color", "yellow");
         });
      </script>
```

```
   </head>

   <body>

      <div class="big" id="div1">
         <p>This is first division of the DOM.</p>
      </div>

      <div class="medium" id="div2">
         <p>This is second division of the DOM.</p>
      </div>

      <div class="small" id="div3">
         <p>This is third division of the DOM</p>
      </div>

   </body>

</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

jQuery - Universal Selector

Description

The universal selector selects all the elements available in the document.

Syntax

Here is the simple syntax to use this selector –

```
$('*')
```

Parameters

Here is the description of all the parameters used by this selector –

***** – A symbolic star.

Returns

Like any other jQuery selector, this selector also returns an array filled with the found elements.

Example

- **$('*')** selects all the elements available in the document.

Following example would select all the elements and will apply yellow color to their background. Try to understand that this selector will select every element including head, body etc.

```
<html>
   <head>
      <title>The Selecter Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            /* This would select all the elements */
            $("*").css("background-color", "yellow");
         });
      </script>

   </head>

   <body>

      <div class="big" id="div1">
         <p>This is first division of the DOM.</p>
      </div>

      <div class="medium" id="div2">
         <p>This is second division of the DOM.</p>
```

```
      </div>

      <div class="small" id="div3">
         <p>This is third division of the DOM</p>
      </div>

   </body>

</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

jQuery - Multiple Elements Selector

Description

This Multiple Elements selector selects the combined results of all the specified selectors E, F or G.

You can specify any number of selectors to combine into a single result. Here order of the DOM elements in the jQuery object aren't necessarily identical.

Syntax

Here is the simple syntax to use this selector –

```
$('E, F, G,....')
```

Parameters

Here is the description of all the parameters used by this selector –

- **E –** Any valid selector
- **F –** Any valid selector
- **G –** Any valid selector

Returns

Like any other jQuery selector, this selector also returns an array filled with the found elements.

Example

- **$('div, p') –** selects all the elements matched by **div** or **p**.
- **$('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**.
- **$('p strong, #myid') –** selects a single elements matched by **strong** that is descendant of an element matched by **p** as well as element whose id is **myid**.

Following example would select elements with class ID **big** and element with ID **div3** and will apply yellow color to its background –

```
<html>
   <head>
      <title>The Selecter Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $(".big, #div3").css("background-color", "yellow");
         });
      </script>
   </head>

   <body>

      <div class="big" id="div1">
         <p>This is first division of the DOM.</p>
      </div>

      <div class="medium" id="div2">
         <p>This is second division of the DOM.</p>
      </div>

      <div class="small" id="div3">
         <p>This is third division of the DOM</p>
      </div>
```

```
   </body>


</html>
```

This will produce the following result:

This is first division of the DOM.

This is second division of the DOM.

This is third division of the DOM

Selectors Examples

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

S.N.	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 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 element of the .
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 that are children of an element matched by
11	$("strong + em") Selects all elements matched by that immediately follow a sibling element matched by .
12	$("p ~ ul") Selects all elements matched by that follow a sibling element matched by <p>.
13	$("code, em, strong") Selects all elements matched by <code> or or .
14	$("p strong, .myclass") Selects all elements matched by 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 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 a href value containing domain.com.
24	$("li:even") Selects all elements matched by 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 element.
27	$("li:last") Selects the last element.
28	$("li:visible") Selects all elements matched by that are visible.

29	$("li:hidden") Selects all elements matched by that are hidden.
30	$(":radio") Selects all radio buttons in the form.
31	$(":checked") Selects all checked boxes 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 element.
35	$("li:eq(4)") Selects the fifth element.
36	$("li:lt(2)") Selects all elements matched by element before the third one; in other words, the first two 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 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 .
41	$("div//code") Selects all elements matched by <code>that are descendants of an element matched by .
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 that are the first child of their parent.
44	$("li:last-child") Selects all elements matched by 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 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.

4. JQUERY ATTRIBUTES

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="myImage" src="image.gif" alt="An image"
class="someClass" 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 tag and set value with the same value:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {
      var title = $("em").attr("title");
```

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

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

This will produce the following result:

```
This is first paragraph.
This is second paragraph.

Bold and Brave
```

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:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

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

```
   </script>
</head>
<body>
   <div>
      <img id="myimg" src="/wongpath.jpg" alt="Sample image" />
   </div>
</body>
</html>
```

This will produce the following result:

Applying Styles

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:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

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

```
   </script>
   <style>
      .selected { color:red; }
      .highlight { background:yellow; }
  </style>
</head>
<body>
   <em title="Bold and Brave">This is first paragraph.</em>
   <p id="myid">This is second paragraph.</p>
</body>
</html>
```

This will produce the following result:

Attribute Methods

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

S.N.	Methods & Description
1	attr(properties) Set a key/value object as properties to all matched elements.
2	attr(key, fn) Set a single property to a computed value, on all matched elements.
3	removeAttr(name) Remove an attribute from each of the matched elements.
4	hasClass(class)

	Returns true if the specified class is present on at least one of the set of matched elements.
5	removeClass(class) Removes all or the specified class(es) from the set of matched elements.
6	toggleClass(class) Adds the specified class if it is not present, removes the specified class if it is present.
7	html( ) Get the html contents (innerHTML) of the first matched element.
8	html(val) Set the html contents of every matched element.
9	text( ) Get the combined text contents of all matched elements.
10	text(val) Set the text contents of all matched elements.
11	val( ) Get the input value of the first matched element.
12	val(val) 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.

attr(properties) Method

The **attr(properties)** method set a key/value object as properties to all matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.attr({property1:value1, property2:value2})
```

Parameters

Here is the description of all the parameters used by this method:

- **property:** This is the CSS property of the matched element.
- **value:** This is the value of the property to be set.

Example

Following example would change the properties of an image tag:

```
<html>
<head>
<title>The Selecter Example</title>
<script type="text/javascript" src="/jquery/jquery-1.3.2.min.js">
</script>

<script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("img").attr({
          src: "/images/jquery.jpg",
          title: "jQuery",
          alt: "jQuery Logo"
        });

   });

</script>
</head>
<body>

<div class="divcl" id="divid">
  <p>Following is the logo of jQuery</p>
```

```
   <img src="Wrong SRC" title="None" alt="None" />
</div>


</body>
</html>
```

This will produce the following result:

attr(key, func) Method

The **attr(key, func)** method sets a single property to a computed value, on all matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.attr( key, func )
```

Parameters

Here is the description of all the parameters used by this method:

- **key:** The name of the property to set.
- **func:** A function returning the value to set. This function would have one argument which is index of current element.

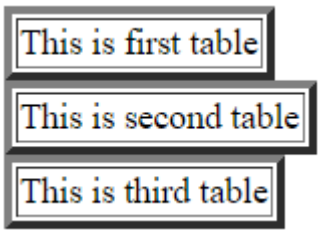
Example

The following example would create border for each table:

```
<html>
<head>
<title>The Selecter Example</title>
<script type="text/javascript" src="/jquery/jquery-1.3.2.min.js">
</script>
```

```
<script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("table").attr("border", function(arr) {
          return arr;
        })
   });
</script>
</head>
<body>

<table>
<tr><td>This is first table</td></tr>
</table>

<table>
<tr><td>This is second table</td></tr>
</table>

<table>
<tr><td>This is third table</td></tr>
</table>

</body>
</html>
```

This would produce the following result:



removeAttr(name) Method

The **removeAttr(name)** method removes an attribute from each of the matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.removeAttr( name )
```

Parameters

Here is the description of all the parameters used by this method:

- **name:** The name of the property to be removed.

Example

Following example would remove border from each table:

```
<html>
<head>
<title>The Selecter Example</title>
<script type="text/javascript" src="/jquery/jquery-1.3.2.min.js">
</script>

<script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("table").removeAttr("border");

   });

</script>
```

```
</head>
<body>

<table border="2">
<tr><td>This is first table</td></tr>
</table>

<table border="3">
<tr><td>This is second table</td></tr>
</table>

<table border="4">
<tr><td>This is third table</td></tr>
</table>

</body>
</html>
```

This would produce the following result:

```
This is first table
This is second table
This is third table
```

hasClass(class) Method

The **hasClass(class)** method returns true if the specified class is present on at least one of the set of matched elements otherwise it returns false.

Syntax

Here is the simple syntax to use this method:

```
selector.hasClass( class )
```

Parameters

Here is the description of all the parameters used by this method:

- **class:** The name of CSS class.

Example

Following example would check which para has class red:

```
<html>
   <head>
      <title>The Selecter Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("#result1").text( $("p#pid1").hasClass("red") );
            $("#result2").text( $("p#pid2").hasClass("red") );
         });
      </script>

      <style>
         .red { color:red; }
         .green { color:green; }
      </style>

   </head>

   <body>

      <p class="red" id="pid1">This is first paragraph.</p>
      <p class="green" id="pid2">This is second paragraph.</p>

      <div id="result1"></div>
      <div id="result2"></div>

   </body>

</html>
```

This would produce the following result:

```
This is first paragraph.
This is second paragraph.


true
false
```

removeClass(class) Method

The **removeClass(class)** method removes all or the specified class(es) from the set of matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.removeClass( class )
```

Parameters

Here is the description of all the parameters used by this method:

- **class:** The name of CSS class.

Example

Following example would remove class red from the first para:

```
<html>
   <head>
      <title>The Selecter Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
      </script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("p#pid1").removeClass("red");
         });
      </script>

      <style>
         .red { color:red; }
         .green { color:green; }
```

```
      </style>

   </head>

   <body>
      <p class="red" id="pid1">This is first paragraph.</p>
      <p class="green" id="pid2">This is second paragraph.</p>
   </body>

</html>
```

This would produce the following result:

```
This is first paragraph.

This is second paragraph.
```

toggleClass(class) Method

The **toggleClass(class)** method adds the specified class if it is not present, removes the specified class if it is present.

Syntax

Here is the simple syntax to use this method:

```
selector.toggleClass( class )
```

Parameters

Here is the description of all the parameters used by this method:

- **class:** The name of CSS class.

Example

Following example would remove a class with one click and in second click it would again add the same class:

```
<html>
   <head>
      <title>The Selecter Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js">
```

```
      </script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("p#pid").click(function () {
               $(this).toggleClass("red");
            });
         });
      </script>

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

   </head>

   <body>
      <p class="green">Click following line to see the result</p>
      <p class="red" id="pid">This is first paragraph.</p>
   </body>

</html>
```

With the first click it would generate following result, in the second click it would revert to its original form as shown above:

```
Click the following line to see the result

This is first paragraph.
```

html( ) Method

The **html( )** method gets the html contents of the first matched element. This property is not available on XML documents but it works for XHTML documents.

Syntax

Here is the simple syntax to use this method:

```
selector.html( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA

Example

Following example would get HTML content of first paragraph and would display it in second paragraph. Please check description of **html(val)** method as well.

```
<html>
<head>
<title>The Selecter Example</title>
<script type="text/javascript" src="/jquery/jquery-1.3.2.min.js">
</script>

<script type="text/javascript" language="javascript">

   $(document).ready(function() {
      var content = $("p").html();
      $("p#pid2").html( content );
   });

</script>
<style>
  .red { color:red; }
  .green { color:green; }
</style>

</head>
<body>

<p class="green" id="pid1">This is first paragraph.</p>
<p class="red" id="pid2">This is second paragraph.</p>

</body>
</html>
```

This would produce the following result.

```
This is first paragraph.


This is first paragraph.
```

html(val) Method

The **html(val)** method sets the html contents of every matched element. This property is not available on XML documents but it works for XHTML documents.

Syntax

Here is the simple syntax to use this method:

```
selector.html( val )
```

Parameters

Here is the description of all the parameters used by this method:

- **val**: Any string

Example

Following example would get HTML content of first paragraph and would display it in second paragraph. Please check description of **html()** method as well.

```
<html>
<head>
<title>The Selecter Example</title>
<script type="text/javascript" src="/jquery/jquery-1.3.2.min.js">
</script>

<script type="text/javascript" language="javascript">

   $(document).ready(function() {
      var content = $("p").html();
      $("p#pid2").html( content );
   });

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

```
  .green { color:green; }
</style>

</head>
<body>

<p class="green" id="pid1">This is first paragraph.</p>
<p class="red" id="pid2">This is second paragraph.</p>

</body>
</html>
```

This would produce the following result.

```
This is first paragraph.
This is first paragraph.
```

text( ) Method

The **text( )** method gets the combined text contents of all matched elements. This method works for both on XML and XHTML documents.

Syntax

Here is the simple syntax to use this method:

```
selector.text( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA

Example

Following example would find the text in the first paragraph stripping out the html, then set the html of the second paragraph to show it is just text

```
<html>
<head>
<title>The Selecter Example</title>
<script type="text/javascript" src="/jquery/jquery-1.3.2.min.js">
</script>
```

```
<script type="text/javascript" language="javascript">

   $(document).ready(function() {
      var content = $("p#pid1").text();
      $("p#pid2").html(content);
   });

</script>
<style>
  .red { color:red; }
  .green { color:green; }
</style>

</head>
<body>

<p class="green" id="pid1">This is <i>first paragraph</i>.</p>
<p class="red" id="pid2">This is second paragraph.</p>

</body>
</html>
```

This would produce the following result.

```
This is first paragraph.
This is first paragraph.
```

text(val) Method

The **text(val)** method sets the text contents of all matched elements. This method is similar to html(val) but escapes all HTML entities.

Syntax

Here is the simple syntax to use this method:

```
selector.text( val )
```

Parameters

Here is the description of all the parameters used by this method:

- **val**: Any string

Example

Following example would set the HTML content of the first paragraph in the second paragraph but it escapes all the HTML tag.

```
<html>
<head>
<title>The Selecter Example</title>
<script type="text/javascript" src="/jquery/jquery-1.3.2.min.js">
</script>

<script type="text/javascript" language="javascript">

   $(document).ready(function() {
      var content = $("p#pid1").html();
      $("p#pid2").text(content);
   });

</script>
<style>
  .red { color:red; }
  .green { color:green; }
</style>

</head>
<body>

<p class="green" id="pid1">This is <i>first paragraph</i>.</p>
<p class="red" id="pid2">This is second paragraph.</p>

</body>
</html>
```

This would produce the following result.

```
This is first paragraph.
```

```
This is <i>first paragraph</i>.
```

val( ) Method

The **val( )** method gets the input value of the first matched element.

Syntax

Here is the simple syntax to use this method:

```
selector.val( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA

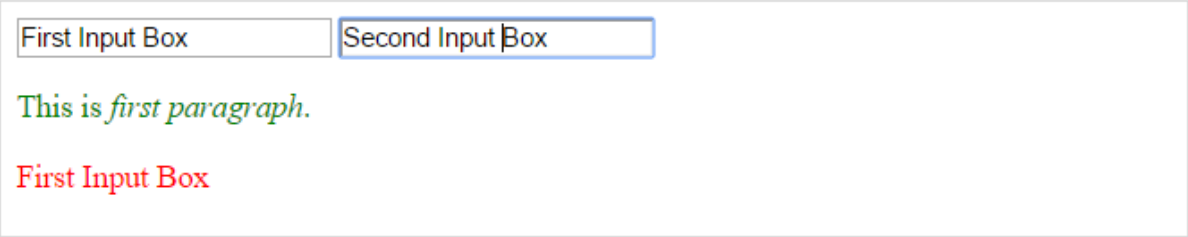
Example

Following example would set the HTML content of the first input box in the second paragraph:

```
<html>
<head>
<title>The Selecter Example</title>
<script type="text/javascript" src="/jquery/jquery-1.3.2.min.js">
</script>

<script type="text/javascript" language="javascript">

   $(document).ready(function() {
      var content = $("input").val();
      $("p#pid2").text(content);
   });

</script>
<style>
  .red { color:red; }
  .green { color:green; }
</style>

</head>
```

```
<body>

<input type="text" value="First Input Box"/>
<input type="text" value="Second Input Box"/>
<p class="green" id="pid1">This is <i>first paragraph</i>.</p>
<p class="red" id="pid2">This is second paragraph.</p>

</body>
</html>
```

This would produce the following result.



val(val) Method

The **val(val)** method sets the input value of every matched element.

If this method is called on radio buttons, checkboxes, or select options then it would check, or select them at the passed value.

Syntax

Here is the simple syntax to use this method:

```
selector.val( val )
```

Parameters

Here is the description of all the parameters used by this method:

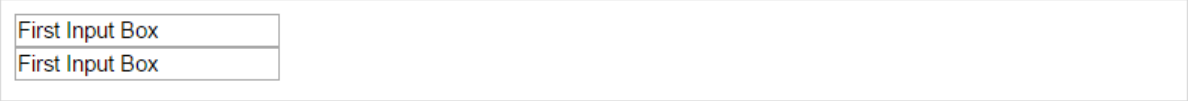
- **val**: If it is called on <input> but if it is called on <select> with the passed <option> value then passed option would be selected, if it is called on check box or radio box then all the matching check box and radiobox would be checked.

Example

Following example would set the value attribute of the second input with the value content of the first input:

```
<html>
<head>
<title>The Selecter Example</title>
<script type="text/javascript" src="/jquery/jquery-1.3.2.min.js">
</script>

<script type="text/javascript" language="javascript">

   $(document).ready(function() {
      var content = $("input").val();
      $("input").val( content );
   });

</script>
</head>
<body>

<input type="text" value="First Input Box"/><br/>
<input type="text" value="Second Input Box"/>

</body>
</html>
```

This would produce the following result.



Examples

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

S.N.	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^=http://]").attr("target","_blank") Selects all links with href attribute starting with *http://* 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("Hello World") 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 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.

5. DOM TRAVERSING

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 title</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 the following result:

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

- 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 title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="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>
```

This will produce the following result

- 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(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:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {
      $("li").filter(".middle").addClass("selected");
   });

   </script>
   <style>
      .selected { color:red; }
  </style>
</head>
<body>
   <div>
   <ul>
     <li class="top">list item 1</li>
     <li class="top">list item 2</li>
```

```
      <li class="middle">list item 3</li>
      <li class="middle">list item 4</li>
      <li class="bottom">list item 5</li>
      <li class="bottom">list item 6</li>
   </ul>
   </div>
</body>
</html>
```

This will produce the following result:

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

Locating Descendent Elements

The **find(selector)** method can be used to locate all the descendent 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 elements available inside different <p> elements:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="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>
```

This will produce the following result

```
This is 1st paragraph and THIS IS RED


This is 2nd paragraph and THIS IS ALSO RED
```

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:

S.N.	Method & Description
1	**eq(index)** Reduce the set of matched elements to a single element.
2	**filter(selector)** Removes all elements from the set of matched elements that do not match the specified selector(s).
3	**filter(fn)** Removes all elements from the set of matched elements that do not match the specified function.
4	**is(selector)** 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)**

	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)** Removes elements matching the specified selector from the set of matched elements.
7	**slice(start, [end])** Selects a subset of the matched elements.

eq(index) Method

The **eq(index)** method reduces the set of matched elements to a single element.

Syntax

Here is the simple syntax to use this method:

```
selector.eq( index )
```

Parameters

Here is the description of all the parameters used by this method:

- **index:** This is the position of the element in the set of matched elements, starting at 0 and going to length - 1.

Example

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

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="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>
```

This would produce the following result:

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

filter(selector) Method

The **filter(selector)** method filters all elements from the set of matched elements that do not match the specified selector(s).

Syntax

Here is the simple syntax to use this method:

```
selector.filter( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** It could be a comma-separated list of expressions to apply multiple filters at once (e.g. filter(".class1, .class2")).

Example

Following is an example showing a simple usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {
      $("li").filter(".middle").addClass("selected");
   });

   </script>
   <style>
      .selected { color:red; }
  </style>
</head>
<body>
   <div>
   <ul>
     <li class="top">list item 1</li>
     <li class="top">list item 2</li>
     <li class="middle">list item 3</li>
     <li class="middle">list item 4</li>
     <li class="bottom">list item 5</li>
     <li class="bottom">list item 6</li>
   </ul>
   </div>
</body>
</html>
```

This would produce the following result:

- list item 1

```
• list item 2
• list item 3
• list item 4
• list item 5
• list item 6
```

filter(fn) Method

The **filter(fn)** method filters all elements from the set of matched elements that do not match the specified function.

Syntax

Here is the simple syntax to use this method:

```
selector.filter( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **fn:** The function is called with a context equal to the current element just like **$.each**. If the function returns false, then the element is removed otherwise the element is kept.

Example

Following is an example showing a simple usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

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

   </script>
```

```
   <style>
      .selected { color:red; }
  </style>
</head>
<body>
   <div>
   <ul>
     <li class="top">list item 1</li>
     <li class="top">list item 2</li>
     <li class="middle">list item 3</li>
     <li class="middle">list item 4</li>
     <li class="bottom">list item 5</li>
     <li class="bottom">list item 6</li>
   </ul>
   </div>
</body>
</html>
```

This would produce the following result:

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

is(selector) Method

The **is(selector)** method checks the current selection against an expression and returns true, if at least one element of the selection fits the given selector.

If no element fits, or the selector is not valid, then the response will be 'false'.

Syntax

Here is the simple syntax to use this method:

```
element.is( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** The expression with which to filter.

Example

Following is an example showing a simple usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

      $("li").click(function () {
         if ($(this).is(":first-child")) {
            $("p").text("This is list item 1");
         } else if ($(this).is(".middle0,.middle1")) {
            $("p").text("This is middle class list");
         } else if ($(this).is(":contains('item 5')")) {
            $("p").text("It's 5th list");
         }
      });
   });

   </script>
   <style>
      .selected { color:red; }
  </style>
</head>
<body>
   <div>
   <span>Click any list item below:</span>
   <ul>
     <li class="top0">list item 1</li>
     <li class="top1">list item 2</li>
```

```
      <li class="middle0">list item 3</li>
      <li class="middle1">list item 4</li>
      <li class="bottom0">list item 5</li>
      <li class="bottom1">list item 6</li>
   </ul>
   <p>FILLER</p>
   </div>
</body>
</html>
```

This would produce the following result:

```
Click any list item below:


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


FILLER
```

map(callback) Method

The **map(callback)** method translates a set of elements in the jQuery object into another set of values in a jQuery array which may, or may not contain elements.

You could use this method to build lists of values, attributes, css values - or even perform special, custom, selector transformations.

Syntax

Here is the simple syntax to use this method:

```
selector.map( callback )
```

Parameters

Here is the description of all the parameters used by this method:

- **callback:** The function to execute on each element in the set.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function(){
      var mappedItems = $("li").map(function (index) {
      var replacement = $("<li>").text($(this).text()).get(0);
      if (index == 0) {
        // make the first item all caps
        $(replacement).text($(replacement).text().toUpperCase());
      } else if (index == 1 || index == 3) {
        // delete the second and fourth items
        replacement = null;
      } else if (index == 2) {
        // make two of the third item and add some text
        replacement = [replacement,$("<li>").get(0)];
        $(replacement[0]).append("<b> - A</b>");
        $(replacement[1]).append("Extra <b> - B</b>");
      }

      // replacement will be an dom element, null,
      // or an array of dom elements
      return replacement;
    });
    $("#results").append(mappedItems);

  });
  </script>
  <style>
     body { font-size:16px; }
     ul { float:left; margin:0 30px; color:blue; }
```

```
    #results { color:red; }
  </style>
</head>
<body>
  <ul>
    <li>First</li>
    <li>Second</li>
    <li>Third</li>
    <li>Fourth</li>
    <li>Fifth</li>
  </ul>
  <ul id="results">
  </ul>
</body>
</html>
```

This will produce the following result:

- First
- Second
- Third
- Fourth
- Fifth

- FIRST
- Third - **A**
- Extra - **B**
- Fifth

not(selector) Method

The **not(selector)** method filters out all the elements matching the specified selector from the set of matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.not( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** It could be a comma-separated list of selectors to apply multiple filters at once (e.g. not(".class1, .class2")).

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {
      $("li").not(".middle").addClass("selected");
   });

   </script>
   <style>
      .selected { color:red; }
  </style>
</head>
<body>
   <div>
   <ul>
     <li class="top">list item 1</li>
     <li class="top">list item 2</li>
     <li class="middle">list item 3</li>
     <li class="middle">list item 4</li>
     <li class="bottom">list item 5</li>
     <li class="bottom">list item 6</li>
   </ul>
   </div>
</body>
</html>
```

This would produce the following result:

```
list item 1
list item 2
list item 3
list item 4
```

```
list item 5
list item 6
```

slice(start, end) Method

The **slice(start, end)** method selects a subset of the matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.slice( start, end )
```

Parameters

Here is the description of all the parameters used by this method:

- **start:** Where to start the subset. The first element is at zero. Can be negative to start from the end of the selection.
- **end:** Where to end the subset excluding end element. If unspecified, ends at the end of the selection.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {
      $("li").slice(2, 5).addClass("selected");
   });

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

```
   <div>
   <ul>
     <li class="above">list item 0</li>
     <li class="top">list item 1</li>
     <li class="top">list item 2</li>
     <li class="middle">list item 3</li>
     <li class="middle">list item 4</li>
     <li class="bottom">list item 5</li>
     <li class="bottom">list item 6</li>
      <li class="below">list item 7</li>
   </ul>
   </div>
</body>
</html>
```

This would produce the following result:

```
list item 0
list item 1
list item 2
list item 3
list item 4
list item 5
list item 6
list item 7
```

JQuery DOM Traversing Methods

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

S.N.	Methods & Description
1	**add(selector)** Adds more elements, matched by the given selector, to the set of matched elements.
2	**andSelf( )**

	Add the previous selection to the current selection.
3	**children([selector])** Get a set of elements containing all the unique immediate children of each matched set of elements.
4	**closest(selector)** Get a set of elements containing the closest parent element that matches the specified selector, the starting element included.
5	**contents( )** Find all the child nodes inside the matched elements (including text nodes), or the content document, if the element is an iframe.
6	**end( )** Revert the most recent 'destructive' operation, changing the set of matched elements to its previous state.
7	**find(selector)** Searches for descendent elements that match the specified selectors.
8	**next([selector])** Get a set of elements containing the unique next siblings of each given set of elements.
9	**nextAll([selector])** Find all sibling elements after the current element.
10	**offsetParent( )** Returns a jQuery collection with the positioned parent of the first matched element.
11	**parent([selector])** 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])**

	Get a set of elements containing the unique ancestors of the matched set of elements (except for the root element).
13	**prev([selector])** Get a set of elements containing the unique previous siblings of each matched set of elements.
14	**prevAll([selector])** Find all sibling elements in front of the current element.
15	**siblings([selector])** Get a set of elements containing all of the unique siblings of each o matched set of elements.

add(selector) Method

The **add(selector)** method adds more elements, matched by the given selector, to the set of matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.add( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** It could be a comma-separated list of selectors to select elements to be added. (e.g. add(".class1, .class2")).

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
```

```
   $(document).ready(function() {
      $(".top").add(".middle").addClass("selected");
   });

   </script>
   <style>
      .selected { color:red; }
  </style>
</head>
<body>
   <div>
   <ul>
     <li class="above">list item 0</li>
     <li class="top">list item 1</li>
     <li class="top">list item 2</li>
     <li class="middle">list item 3</li>
     <li class="middle">list item 4</li>
     <li class="bottom">list item 5</li>
     <li class="bottom">list item 6</li>
     <li class="below">list item 7</li>
   </ul>
   </div>
</body>
</html>
```

This would produce the following result:

```
list item 0
list item 1
list item 2
list item 3
list item 4
list item 5
list item 6
list item 7
```

andSelf( ) Method

The **andSelf( )** method adds the previous selection to the current selection.

The method is useful when you have multiple trasversals in your script and then adding something that was matched before the last traversal.

Syntax

Here is the simple syntax to use this method:

```
selector.andSelf( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA.

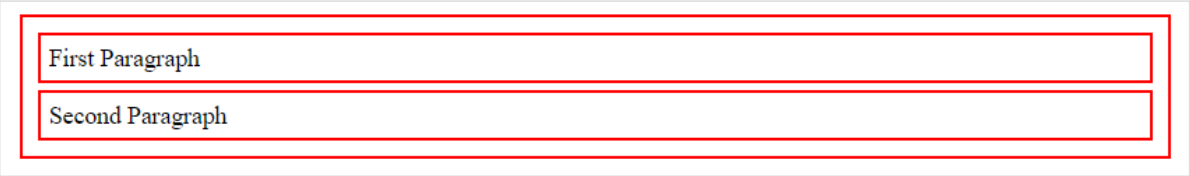
Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
 <script>
  $(document).ready(function(){

    $("div").find("p").andSelf().addClass("border");

  });
  </script>
  <style>
   p, div { margin:5px; padding:5px; }
  .border { border: 2px solid red; }
  .background { background:yellow; }
  </style>
</head>
<body>
  <div>
    <p>First Paragraph</p>
```

```
    <p>Second Paragraph</p>
  </div>
</body>
</html>
```

Here border would be added to previous selection which is a division and then second selection which is paragraphs, as shown below –



children([selector]) Method

The **children([selector])** method gets a set of elements containing all of the unique immediate children of each of the matched set of elements.

Syntax

Here is the simple syntax to use this method:

```
selector.children( [selector] )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector**: This is an optional argument to filter out all the children. If not supplied then all the childerns are selected.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
 <script>
  $(document).ready(function(){

     $("div").children(".selected").addClass("blue");
```

```
  });
  </script>
  <style>
  .blue { color:blue; }
  </style>
</head>
<body>
  <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>
</body>
</html>
```

This would produce following result. This would apply blue color to all children with a class "selected" of each div.

```
Hello
Hello Again

And Again
And One Last Time
```

closest(selector) Method

The **closest(selector)** method works by first looking at the current element to see if it matches the specified expression, if so it just returns the element itself. If it doesn't match then it will continue to traverse up the document, parent by parent, until an element is found that matches the specified expression. If no matching element is found then none will be returned.

Syntax

Here is the simple syntax to use this method:

```
selector.children( [selector] )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector**: This is the selector to be used to filter the elements.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
 <script>
  $(document).ready(function(){

    $(document).bind("click", function (e) {
      $(e.target).closest("li").toggleClass("highlight");
    });

  });
  </script>
  <style>
  .highlight { color:red;
          background: yellow;
        }
  </style>
</head>
<body>
   <div>
   <p>Click any item below to see the result:</p>
   <ul>
     <li class="top">list item 1</li>
     <li class="top">list item 2</li>
     <li class="middle">list item 3</li>
     <li class="middle">list item 4</li>
     <li class="bottom">list item 5</li>
     <li class="bottom">list item 6</li>
```

```
   </ul>
   </div>
</body>
</html>
```

This will produce the the following result:

```
Click any item below to see result:
list item 1
list item 2
list item 3
list item 4
list item 5
list item 6
```

contents( ) Method

The **contents( )** method finds all the child nodes inside the matched elements (including text nodes), or the content document, if the element is an iframe.

Syntax

Here is the simple syntax to use this method:

```
selector.contents( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA

Example

Consider you have a blank html file index-blank.htm which we would use in an iframe.

Try the following example which shows how you can access the objects in an iframe from a parent window. This operation has become possible just because of **contents()** method.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
<script>
```

```
  $(document).ready(function(){
    var $content = $("iframe").contents();
    $content.find("body").append("I'm in an iframe!");
  });
  </script>

</head>
<body>
  <iframe src="/jquery/index-blank.htm" width="300" height="100">
  </iframe>
</body>
</html>
```

This will produce the following result:

end( ) Method

The **end( )** method reverts the most recent **destructive** operation, changing the set of matched elements to its previous state right before the destructive operation.

Syntax

Here is the simple syntax to use this method:

```
operations.end( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA

Example

Following is a simple example showing the usage of this method. This selects all paragraphs, finds span elements inside these, and reverts the selection back to the paragraphs.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script>
   $(document).ready(function(){
      $("p").find("span").end().css("border", "2px red solid");
   });
  </script>
  <style>
      p{
         margin:10px;
         padding:10px;
      }
  </style>
</head>
<body>
  <p><span>Hello</span>, how are you?</p>
</body>
</html>
```

This will produce the following result:

find(selector) Method

The **find(selector)** method searches for descendent elements that match the specified selector.

Syntax

Here is the simple syntax to use this method:

```
selector.find( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** The selector can be written using CSS 1-3 selector syntax.

Example

Following is a simple example showing the usage of this method.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {
      $("p").find("span").addClass("selected");
   });

   </script>
   <style>
      .selected { color:red; }
  </style>
</head>
<body>
   <div>
      <p><span>Hello</span>, how are you?</p>
      <p>Me? I'm <span>good</span>.</p>
   </div>
</body>
</html>
```

This would produce the following result.

```
Hello, how are you?


Me? I'm good.
```

next([selector]) Method

The **next([selector])** method gets a set of elements containing the unique next siblings of each of the given set of elements.

Syntax

Here is the simple syntax to use this method:

```
selector.next( [selector] )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** The optional selector can be written using CSS 1-3 selector syntax. If we supply a selector expression, the element is unequivocally included as part of the object. If we do not supply one, the element would be tested for a match before it was included.

Example

Following is a simple example showing the usage of this method. Try this example without passing selector in next() method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

     $(document).ready(function(){
       $("p").next(".selected").addClass("hilight");
     });

  </script>
  <style>
      .hilight { background:yellow; }
  </style>

</head>
<body>
  <p>Hello</p>
```

```
  <p class="selected">Hello Again</p>
  <div><span>And Again</span></div>
</body>
</html>
```

This would produce the following result.

nextAll([selector]) Method

The **nextAll([selector])** method finds all sibling elements after the current element.

Syntax

Here is the simple syntax to use this method:

```
selector.nextAll( [selector] )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** The optional selector can be written using CSS 1-3 selector syntax. If we supply a selector then result would be filtered out.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

     $(document).ready(function(){
       $("div:first").nextAll().addClass("hilight");
```

```
    });

  </script>
  <style>
      .hilight { background:yellow; }
  </style>

</head>
<body>
   <div>first</div>
   <div>sibling<div>child</div></div>
   <div>sibling</div>
   <div>sibling</div>
</body>
</html>
```

This would produce the following result.

offsetParent() Method

The **offsetParent()** method returns a positioned parent (e.g. relative, absolute) of the first selected element.

Syntax

Here is the simple syntax to use this method:

```
selector.offsetParent()
```

Parameters

Here is the description of all the parameters used by this method:

- NA:

Example

Following is a simple example showing the usage of this method.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

     $(document).ready(function(){
          $("p").offsetParent().addClass('hilight');
     });

  </script>
  <style>
      .hilight { background:yellow; }
  </style>

</head>
<body>
     <scan>Top Element</scan>
     <div style="position:relative;">
        <div>sibling<div>child</div></div>
        <p>sibling</p>
        <scan>sibling</scan>
    <div>
</body>
</html>
```

This would produce the following result.

parent([selector]) Method

The **parent([selector])** method gets the direct parent of an element. If called on a set of elements, parent returns a set of their unique direct parent elements.

Syntax

Here is the simple syntax to use this method:

```
selector.parent( [selector] )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** This is optional selector to filter the parent with.

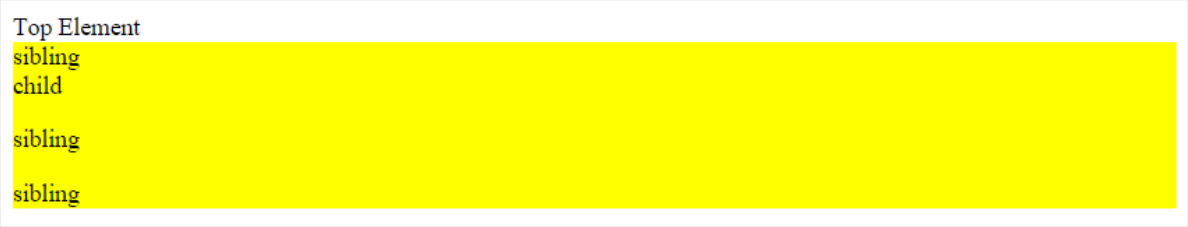
Example

Following is a simple example showing the usage of this method.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

     $(document).ready(function(){
          $("p").parent().addClass('hilight');
     });

  </script>
  <style>
      .hilight { background:yellow; }
  </style>

</head>
<body>
     <scan>Top Element</scan>
     <div">
        <div>sibling<div>child</div></div>
        <p>sibling</p>
        <scan>sibling</scan>
    <div>
```

```
</body>
</html>
```

This would produce the following result.



parents([selector]) Method

The **parents([selector])** method gets a set of elements containing the unique ancestors of the matched set of elements except for the root element.

Syntax

Here is the simple syntax to use this method:

```
selector.parents( [selector] )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** This is optional selector to filter the ancestors with.

Example

Following is a simple example showing the usage of this method. Try this example by passing parent selector like ".top".

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

     $(document).ready(function(){
       var parentEls = $("p").parents()
                               .map(function () {
                                     return this.tagName;
                               }).get().join(", ");
```

```
        $("b").append("<strong>" + parentEls + "</strong>");
     });

  </script>
  <style>
      .hilight { background:yellow; }
  </style>

</head>
<body>
      <scan>Top Element</scan>
     <div>
        <div class="top">Top division
            <p class="first">First Sibling</p>
            <scan>Second sibling</scan>
            <p class="third">Third sibling</p>
        </div>
       <b>My parents are: </b>
    <div>
</body>
</html>
```

This would produce the following result.

```
Top Element
Top division

First Sibling

Second sibling
Third sibling

Parents of <p> elements are: DIV, DIV, BODY, HTML
```

prev([selector]) Method

The **prev([selector])** method gets the immediately preceding sibling of each element in the set of matched elements, optionally filtered by a selector.

Syntax

Here is the simple syntax to use this method:

```
selector.prev( [selector] )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** This is optional selector to filter the previous elements with.

Example

Following is a simple example showing the usage of this method. Try this example without passing selector in prev() method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

     $(document).ready(function(){
       $("p").prev(".selected").addClass("hilight");
     });

  </script>
  <style>
      .hilight { background:yellow; }
  </style>

</head>
<body>
  <div><span>Hello</span></div>
  <p class="selected">Hello Again</p>
  <p>And Again</p>
</body>
</html>
```

This would produce the following result.

prevAll([selector]) Method

The **prevAll([selector])** method finds all sibling elements in front of the current element.

Syntax

Here is the simple syntax to use this method:

```
selector.prevAll( [selector] )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** This is optional selector to filter the previous elements with.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

     $(document).ready(function(){
       $("div:last").prevAll().addClass("hilight");
     });

  </script>
  <style>
      .hilight { background:yellow; }
  </style>
```

```
</head>
<body>
   <div>first</div>
   <div>sibling<div>child</div></div>
   <div>sibling</div>
   <div>sibling</div>
</body>
</html>
```

This would produce the following result.

siblings([selector]) Method

The **siblings([selector])** method gets a set of elements containing all of the unique siblings of each of the matched set of elements.

Syntax

Here is the simple syntax to use this method:

```
selector.siblings( [selector] )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** This is optional selector to filter the sibling elements with.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
```

```
   <script type="text/javascript" language="javascript">

     $(document).ready(function(){
       $("p").siblings('.selected').addClass("hilight");
     });

  </script>
  <style>
      .hilight { background:yellow; }
  </style>

</head>
<body>
  <div><span>Hello</span></div>
  <p class="selected">Hello Again</p>
  <p>And Again</p>
</body>
</html>
```

This would produce the following result.

6. CSS SELECTOR 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.

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.

```
• list item 1
• list item 2
• list item 3
• list item 4
• list item 5
• list item 6
```

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.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

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

   </script>
</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 the the following result:

- list item 1
- list item 2
- list item 3
- list item 4

```
• list item 5
• list item 6
```

Setting Element Width & Height

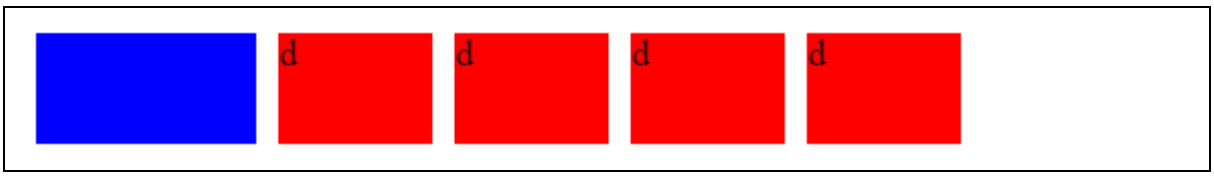
The **width(val)** and **height(val)** method can be used to set the width and height respectively of any element.

Example

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

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {
      $("div:first").width(100);
      $("div:first").css("background-color", "blue");
   });

   </script>
   <style>
   div{ width:70px; height:50px; float:left; margin:5px;
      background:red; cursor:pointer; }
  </style>
</head>
<body>
  <div></div>
  <div>d</div>
  <div>d</div>
  <div>d</div>
  <div>d</div>
</body>
</html>
```

This will produce the following result:



JQuery CSS Methods

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

S.No.	Method & Description
1	css(name) Return a style property on the first matched element.
2	css(name, value) Set a single style property to a value on all matched elements.
3	css(properties) Set a key/value object as style properties to all matched elements.
4	height(val) Set the CSS height of every matched element.
5	height( ) Get the current computed, pixel, height of the first matched element.
6	innerHeight( ) Gets the inner height (excludes the border and includes the padding) for the first matched element.
7	innerWidth( ) Gets the inner width (excludes the border and includes the padding) for the first matched element.
8	offset( ) Get the current offset of the first matched element, in pixels, relative to the document.

9	offsetParent( ) Returns a jQuery collection with the positioned parent of the first matched element.
10	outerHeight([margin]) Gets the outer height (includes the border and padding by default) for the first matched element.
11	outerWidth([margin]) Get the outer width (includes the border and padding by default) for the first matched element.
12	position( ) Gets the top and left position of an element relative to its offset parent.
13	scrollLeft(val) When a value is passed in, the scroll left offset is set to that value on all matched elements.
14	scrollLeft( ) Gets the scroll left offset of the first matched element.
15	scrollTop(val) When a value is passed in, the scroll top offset is set to that value on all matched elements.
16	scrollTop( ) Gets the scroll top offset of the first matched element.
17	width(val) Set the CSS width of every matched element.
18	width( ) Get the current computed, pixel, width of the first matched element.

css(name) Method

The **css(name)** method returns a style property on the first matched element.

Syntax

Here is the simple syntax to use this method:

```
selector.css( name )
```

Parameters

Here is the description of all the parameters used by this method:

- **name:** The name of the property to access.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var color = $(this).css("background-color");
      $("#result").html("That div is <span style='color:" +
                        color + ";'>" + color + "</span>.");
    });

   });

   </script>
   <style>
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
```

```
   <div style="background-color:blue;"></div>
   <div style="background-color:rgb(15,99,30);"></div>
   <div style="background-color:#123456;"></div>
   <div style="background-color:#f11;"></div>
</body>
</html>
```

This will produce the the following result:

```
Click on any square:
```

css(name, value) Method

The **css(name, value)** method sets a single style property to a value on all matched elements..

Syntax

Here is the simple syntax to use this method:

```
selector.css( name, value )
```

Parameters

Here is the description of all the parameters used by this method:

- **name:** The name of the property to be set.
- **value:** The value of the property.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
```

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

     $("div").click(function () {
      var color = $(this).css("background-color");
      $("#result").html("That div is <span>" + color + "</span>.");
      $("#result").css("color", color);
    });

   });

   </script>
   <style>
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div style="background-color:blue;"></div>
   <div style="background-color:rgb(15,99,30);"></div>
   <div style="background-color:#123456;"></div>
   <div style="background-color:#f11;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square:
```

css(properties) Method

The **css(properties)** method sets a key/value object as style properties to all matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.css( properties )
```

The above syntax can be written as follows:

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

Parameters

Here is the description of all the parameters used by this method:

- **properties:** Key/value pairs to set as style properties.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var color = $(this).css("background-color");
      $("#result").html("That div is <span>" + color + "</span>.");
      $("#result").css({'color': color,
                        'font-weight' : 'bold',
                        'background-color': 'gray'});
    });

   });

   </script>
   <style>
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
```

```
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div style="background-color:blue;"></div>
   <div style="background-color:rgb(15,99,30);"></div>
   <div style="background-color:#123456;"></div>
   <div style="background-color:#f11;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square:
```

height(val) Method

The **height(val)** method sets the CSS height of every matched element.

Syntax

Here is the simple syntax to use this method:

```
selector.height( val )
```

Parameters

Here is the description of all the parameters used by this method:

- **val:** This is height of the element. If no explicit unit was specified (like 'em' or '%') then "px" is concatenated to the value.

Example

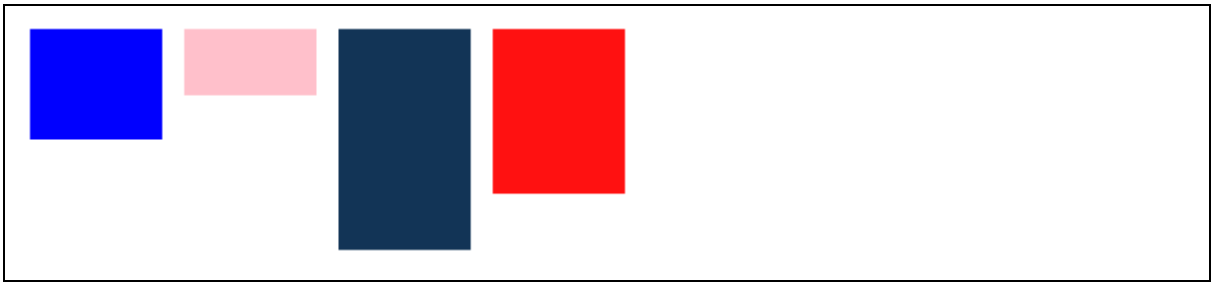
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
```

```
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var color = $(this).css("background-color");
      var height = $(this).height();
      $("#result").html("That div is <span>" +color+ "</span>.");
      $("#result").css({'color': color, 'background-color':'gray'});
      $("#result").height( height );
    });

   });

   </script>
   <style>
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div style="background-color:blue; height:50px;"></div>
   <div style="background-color:pink;height:30px;"></div>
   <div style="background-color:#123456;height:100px;"></div>
   <div style="background-color:#f11; height:75px;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square:
```



height( ) Method

The **height( )** method gets the current computed, pixel, height of the first matched element.

Syntax

Here is the simple syntax to use this method:

```
selector.height( )
```

This method is able to find the height of the window and document as follows:

```
$(window).height();   // returns height of browser viewport
$(document).height(); // returns height of HTML document
```

Parameters

Here is the description of all the parameters used by this method:

- **NA:**

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var color = $(this).css("background-color");
```

```
      var height = $(this).height();
      $("#result").html("That div is <span>" +color+ "</span>.");
      $("#result").css({'color': color,
                        'background-color':'gray'});
      $("#result").height( height );
    });

   });

   </script>
   <style>
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div style="background-color:blue; height:50px;"></div>
   <div style="background-color:pink;height:30px;"></div>
   <div style="background-color:#123456;height:100px;"></div>
   <div style="background-color:#f11; height:75px;"></div>
</body>
</html>
```

This will produce the following result:

Click on any square:

innerHeight( ) Method

The **innerHeight( )** method gets the inner height (excludes the border and includes the padding) for the first matched element.

Syntax

Here is the simple syntax to use this method:

```
selector.innerHeight( )
```

Parameters

Here is the description of all the parameters used by this method:

- **NA:**

Example

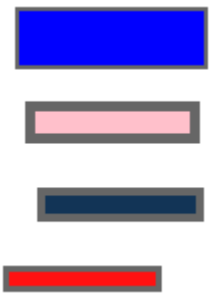
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var color = $(this).css("background-color");
      var height = $(this).innerHeight();
      $("#result").html("Inner Height is <span>" +
                        height + "</span>.");
      $("#result").css({'color': color,
                       'background-color':'gray'});
      $("#result").height( height );
    });

   });

   </script>
   <style>
      #div1{ margin:10px;padding:12px;
             border:2px solid #666;
```

```
            width:60px;}
      #div2 { margin:15px;padding:5px;
              border:5px solid #666;
              width:60px;}
      #div3 { margin:20px;padding:4px;
              border:4px solid #666;
              width:60px;}
      #div4 { margin:5px;padding:3px;
              border:3px solid #666;
              width:60px;}
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div id="div1" style="background-color:blue;"></div>
   <div id="div2" style="background-color:pink;"></div>
   <div id="div3" style="background-color:#123456;"></div>
   <div id="div4" style="background-color:#f11;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square:
```



innerWidth( ) Method

The **innerWidth( )** method gets the inner width (excludes the border and includes the padding) for the first matched element.

Syntax

Here is the simple syntax to use this method:

```
selector.innerWidth( )
```

Parameters

Here is the description of all the parameters used by this method:

- **NA:**

Example

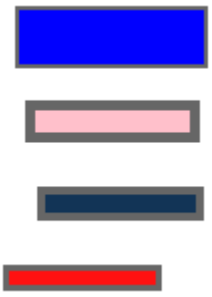
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var color = $(this).css("background-color");
      var width = $(this).innerWidth();
      $("#result").html("Inner Width is <span>" +
                         width + "</span>.");
      $("#result").css({'color': color,
                        'background-color':'gray'});
    });

   });

   </script>
   <style>
      #div1{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;}
      #div2 { margin:15px;padding:5px;
```

```
            border:5px solid #666;
            width:60px;}
      #div3 { margin:20px;padding:4px;
            border:4px solid #666;
            width:60px;}
      #div4 { margin:5px;padding:3px;
            border:3px solid #666;
            width:60px;}
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div id="div1" style="background-color:blue;"></div>
   <div id="div2" style="background-color:pink;"></div>
   <div id="div3" style="background-color:#123456;"></div>
   <div id="div4" style="background-color:#f11;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square:
```



offset( ) Method

The **offset( )** method gets the current offset of the first matched element, in pixels, relative to the document.

The returned object contains two Float properties, top and left. Browsers usually round these values to the nearest integer pixel for actual positioning. The method works only with visible elements.

Syntax

Here is the simple syntax to use this method:

```
selector.offset( )
```

Parameters

Here is the description of all the parameters used by this method:

- **NA:**

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var offset = $(this).offset();
      $("#lresult").html("left offset: <span>" +
                         offset.left + "</span>.");

      $("#tresult").html("top offset: <span>" +
                         offset.top + "</span>.");
    });

   });

   </script>
   <style>
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
```

```
<body>
   <p>Click on any square:</p>
   <span id="lresult"> </span>
   <span id="tresult"> </span>
   <div  style="background-color:blue;"></div>
   <div  style="background-color:pink;"></div>
   <div  style="background-color:#123456;"></div>
   <div  style="background-color:#f11;"></div>
</body>
</html>
```

This will produce the the following result:

```
Click on any square:
```

offsetParent( ) Method

The **offsetParent( )** method returns a jQuery collection with the positioned parent of the first matched element.

This is the first parent of the element that has position (as in relative or absolute). This method only works with visible elements.

Syntax

Here is the simple syntax to use this method:

```
selector.offsetParent( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA:

Example

Following is a simple example showing the usage of this method:

```
<html>
```

```
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

    $(document).ready(function() {

     $("div").click(function () {
      var offset = $(this).offsetParent();
      $("#lresult").html("left offset: <span>" +
                         offset.offset().left + "</span>.");

      $("#tresult").html("top offset: <span>" +
                         offset.offset().top + "</span>.");
    });

   });

   </script>
   <style>
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="lresult"> </span>
   <span id="tresult"> </span>
   <div  style="background-color:blue;">
       <div  style="background-color:pink;"></div>
   </div>

   <div  style="background-color:#123456;">
      <div  style="background-color:#f11;"></div>
   </div>
```

```
</body>
</html>
```

This will produce the following result:

```
Click on any square:
```

outerHeight([margin]) Method

The **outerHeight([margin])** method gets the outer height (includes the border and padding by default) for the first matched element.

This method works for both visible and hidden elements. It is not supported for elements that are indirectly hidden by consequence of a parent being hidden.

Syntax

Here is the simple syntax to use this method:

```
selector.outerHeight( [margin] )
```

Parameters

Here is the description of all the parameters used by this method:

- **margin:** This optional argument when set to true the margin of the element will be included in the calculations.

Example

Following is a simple example showing the usage of this method:

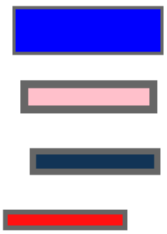
```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

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

```
     $("div").click(function () {
      var color = $(this).css("background-color");
      var height = $(this).outerHeight();
      $("#result").html("Outer Height is <span>" +
                          height + "</span>.");
      $("#result").css({'color': color,
                        'background-color':'gray'});
      $("#result").height( height );
    });

   });

   </script>
   <style>
      #div1{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;}
      #div2 { margin:15px;padding:5px;
              border:5px solid #666;
              width:60px;}
      #div3 { margin:20px;padding:4px;
              border:4px solid #666;
              width:60px;}
      #div4 { margin:5px;padding:3px;
              border:3px solid #666;
              width:60px;}
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div id="div1" style="background-color:blue;"></div>
   <div id="div2" style="background-color:pink;"></div>
   <div id="div3" style="background-color:#123456;"></div>
   <div id="div4" style="background-color:#f11;"></div>
</body>
```

```
</html>
```

This will produce the following result:

```
Click on any square:
```



outerWidth([margin]) Method

The **outerWidth([margin])** method gets the outer width (includes the border and padding by default) for the first matched element.

This method works for both visible and hidden elements. It is not supported for elements that are indirectly hidden by consequence of a parent being hidden.

Syntax

Here is the simple syntax to use this method:

```
selector.outerWidth( [margin] )
```

Parameters

Here is the description of all the parameters used by this method:

- **margin:** This optional argument when set to true, the margin of the element will be included in the calculations.

Example

Following is a simple example showing the usage of this method:

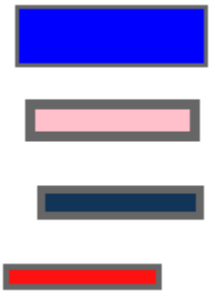
```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
```

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

     $("div").click(function () {
      var color = $(this).css("background-color");
      var width = $(this).outerWidth( true );
      $("#result").html("Outer Width is <span>" +
                         width + "</span>.");
      $("#result").css({'color': color,
                        'background-color':'gray'});
    });

   });

   </script>
   <style>
      #div1{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;}
      #div2 { margin:15px;padding:5px;
              border:5px solid #666;
              width:60px;}
      #div3 { margin:20px;padding:4px;
              border:4px solid #666;
              width:60px;}
      #div4 { margin:5px;padding:3px;
              border:3px solid #666;
              width:60px;}
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div id="div1" style="background-color:blue;"></div>
   <div id="div2" style="background-color:pink;"></div>
   <div id="div3" style="background-color:#123456;"></div>
   <div id="div4" style="background-color:#f11;"></div>
```

```
</body>
</html>
```

This will produce the following result:

```
Click on any square:
```



position( ) Method

The **position( )** method gets the top and left position of an element relative to its offset parent.

The returned object contains two integer properties, top and left. For accurate calculations make sure to use pixel values for margins, borders and padding. This method only works with visible elements.

Syntax

Here is the simple syntax to use this method:

```
selector.position( )
```

Parameters

Here is the description of all the parameters used by this method:

- **NA**

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
```

```
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var position = $(this).position();
      $("#lresult").html("left position: <span>" +
                         position.left + "</span>.");

      $("#tresult").html("top position: <span>" +
                         position.top + "</span>.");
    });

   });

   </script>
   <style>
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="lresult"> </span>
   <span id="tresult"> </span>
   <div  style="background-color:blue;"></div>
   <div  style="background-color:pink;"></div>
   <div  style="background-color:#123456;"></div>
   <div  style="background-color:#f11;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square:
```

scrollLeft(val) Method

The **scrollLeft(val)** method is used to set scroll left offset to the passed value on all matched elements. This method works for both visible and hidden elements.

Syntax

Here is the simple syntax to use this method:

```
selector.scrollLeft( val )
```

Parameters

Here is the description of all the parameters used by this method:

- **val:** A positive number representing the desired scroll left offset.

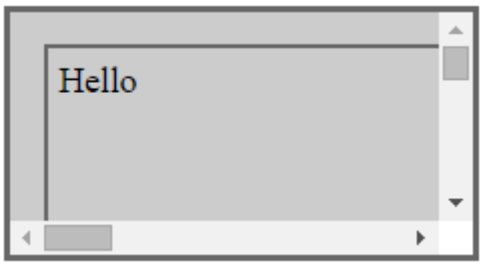
Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function(){
       $("div.demo").scrollLeft(300);
   });
   </script>
   <style>
      div.demo {
      background:#CCCCCC none repeat scroll 0 0;
      border:3px solid #666666;
      margin:5px;
      padding:5px;
```

```
      position:relative;
      width:200px;
      height:100px;
      overflow:auto;
  }
  p { margin:10px;
      padding:5px;
      border:2px solid #666;
      width:1000px;
      height:1000px;
  }
</style>
</head>
<body>
  <div class="demo"><p>Hello</p></div>
</body>
</html>
```

This will produce the following result:



scrollLeft( ) Method

The **scrollLeft( )** method gets the scroll left offset of the first matched element. This method works for both visible and hidden elements.

Syntax

Here is the simple syntax to use this method:

```
selector.scrollLeft( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA:

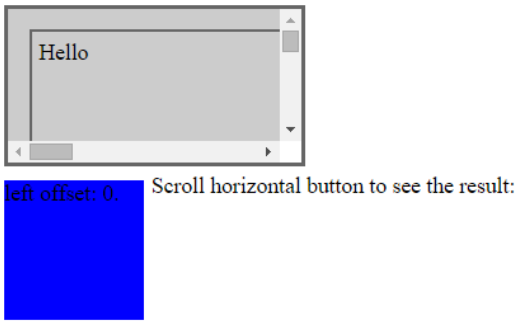
Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function(){
       $("div.demo").scrollLeft( 200 );

       $("div.demo").mousemove(function () {
          var left = $("div.demo").scrollLeft();
          $("#result").html("left offset: <span>" +
                        left + "</span>.");
       });

   });
   </script>
   <style>
      div.demo {
      background:#CCCCCC none repeat scroll 0 0;
      border:3px solid #666666;
      margin:5px;
      padding:5px;
      position:relative;
      width:200px;
      height:100px;
      overflow:auto;
  }
  div.result{
      margin:10px;
      width:100px;
```

```
      height:100px;
      margin:5px;
      float:left;
      background-color:blue;
  }
  p { margin:10px;
      padding:5px;
      border:2px solid #666;
      width:1000px;
      height:1000px;
  }
</style>
</head>
<body>
  <div class="demo"><p>Hello</p></div>
   <span>Scroll horizontal  button to see the result:</span>
  <div class="result"><span id="result"></span></div>
</body>
</html>
```

This will produce the following result:



scrollTop(val) Method

The **scrollTop(val)** method is used to set scroll top offset to the passed value on all matched elements. This method works for both visble and hidden elements.

Syntax

Here is the simple syntax to use this method:

```
selector.scrollTop( val )
```

Parameters

Here is the description of all the parameters used by this method:

- **val:** A positive number representing the desired scroll top offset.

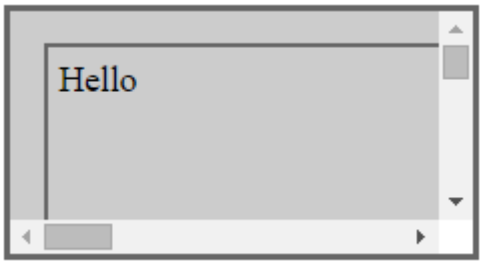
Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function(){
       $("div.demo").scrollTop(300);
   });
   </script>
   <style>
      div.demo {
      background:#CCCCCC none repeat scroll 0 0;
      border:3px solid #666666;
      margin:5px;
      padding:5px;
      position:relative;
      width:200px;
      height:100px;
      overflow:auto;
  }
  p { margin:10px;
      padding:5px;
      border:2px solid #666;
```

```
      width:1000px;
      height:1000px;
  }
</style>
</head>
<body>
  <div class="demo"><p>Hello</p></div>
</body>
</html>
```

This will produce the following result:



scrollTop( ) Method

The **scrollTop( )** method gets the scroll top offset of the first matched element. This method works for both visible and hidden elements.

Syntax

Here is the simple syntax to use this method:

```
selector.scrollTop( )
```

Parameters

Here is the description of all the parameters used by this method:

- **NA:**

Example

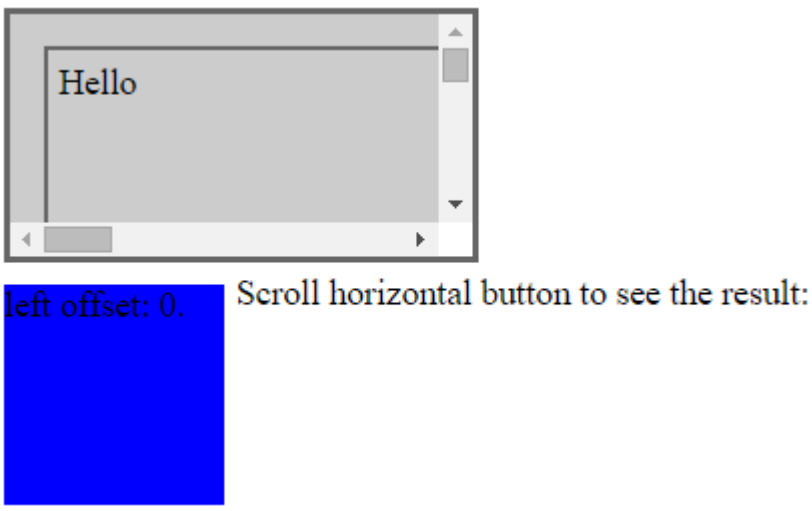
Following is a simple example showing the usage of this method:

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

```
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function(){
       $("div.demo").scrollTop( 200 );

       $("div.demo").mousemove(function () {
          var top = $("div.demo").scrollTop();
          $("#result").html("top offset: <span>" +
                        top + "</span>.");
       });

   });
   </script>
   <style>
      div.demo {
      background:#CCCCCC none repeat scroll 0 0;
      border:3px solid #666666;
      margin:5px;
      padding:5px;
      position:relative;
      width:200px;
      height:100px;
      overflow:auto;
  }
  div.result{
      margin:10px;
      width:100px;
      height:100px;
      margin:5px;
      float:left;
      background-color:blue;
  }
  p { margin:10px;
```

```
      padding:5px;
      border:2px solid #666;
      width:1000px;
      height:1000px;
  }
</style>
</head>
<body>
  <div class="demo"><p>Hello</p></div>
   <span>Scroll vertical button to see the result:</span>
  <div class="result"><span id="result"></span></div>
</body>
</html>
```

This will produce the following result:



width(val) Method

The **width(val)** method sets the CSS width of every matched element.

Syntax

Here is the simple syntax to use this method:

```
selector.width( val )
```

Parameters

Here is the description of all the parameters used by this method:

- **val:** This is width of the element. If no explicit unit was specified (like 'em' or '%') then "px" is concatenated to the value.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var color = $(this).css("background-color");
      var width = $(this).height();
      $("#result").html("That div is <span>" +color+ "</span>.");
      $("#result").css({'color': color, 'background-color':'gray'});
      $("#result").width( width );
    });

   });

   </script>
   <style>
```

```
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div style="background-color:blue; height:50px;"></div>
   <div style="background-color:pink;height:30px;"></div>
   <div style="background-color:#123456;height:100px;"></div>
   <div style="background-color:#f11; height:75px;"></div>
</body>
</html>
```

This will produce the following result:

Click on any square:

width( ) Method

The **width( )** method gets the current computed, pixel, width of the first matched element.

Syntax

Here is the simple syntax to use this method:

```
selector.width( )
```

This method is able to find the width of the window and document as follows:

```
$(window).width();   // returns width of browser viewport
$(document).width(); // returns width of HTML document
```

Parameters

Here is the description of all the parameters used by this method:

- **NA:**

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
      var color = $(this).css("background-color");
      var width = $(this).height();
      $("#result").html("That div is <span>" +color+ "</span>.");
      $("#result").css({'color': color,
                        'background-color':'gray'});
      $("#result").width( width );
    });

   });

   </script>
   <style>
      div { width:60px; height:60px; margin:5px; float:left; }
  </style>
</head>
<body>
   <p>Click on any square:</p>
   <span id="result"> </span>
   <div style="background-color:blue; height:50px;"></div>
   <div style="background-color:pink;height:30px;"></div>
   <div style="background-color:#123456;height:100px;"></div>
   <div style="background-color:#f11; height:75px;"></div>
</body>
```

```
</html>
```

This will produce the following result:

```
Click on any square:
```

7. DOM MANIPULATION

JQuery provides methods to manipulate DOM in efficient way. You do not need to write big code to modify the value of any element's attribute or to extract HTML code from a paragraph or division.

JQuery provides methods such as .attr(), .html(), and .val() which act as getters, retrieving information from DOM elements for later use.

Content Manipulation

The **html( )** method gets the html contents (innerHTML) of the first matched element. Here is the syntax for the method:

```
selector.html( )
```

Example

Following is an example which makes use of .html() and .text(val) methods. Here .html() retrieves HTML content from the object and then .text(val) method sets value of the object using passed parameter:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
```

```
      #division{ margin:10px;padding:12px;
                 border:2px solid #666;
                 width:60px;
               }
  </style>
</head>
<body>
   <p>Click on the square below:</p>
   <span id="result"> </span>
   <div id="division" style="background-color:blue;">
     This is Blue Square!!
   </div>
</body>
</html>
```

This will produce the following result:

```
Click on the square below:
```

DOM Element Replacement

You can replace a complete DOM element with the specified HTML or DOM elements. The **replaceWith(content)** method serves this purpose very well.

Here is the syntax for the method:

```
selector.replaceWith( content )
```

Here content is what you want to have instead of original element. This could be HTML or simple text.

Example

Following is an example which would replace division element with "<h1>JQuery is Great</h1>":

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
      #division{ margin:10px;padding:12px;
                 border:2px solid #666;
                 width:60px;
               }
  </style>
</head>
<body>
   <p>Click on the square below:</p>
   <span id="result"> </span>
   <div id="division" style="background-color:blue;">
     This is Blue Square!!
   </div>
</body>
</html>
```

This will produce the following result:

```
Click on the square below:
```

Removing DOM Elements

There may be a situation when you would like to remove one or more DOM elements from the document. JQuery provides two methods to handle the situation. The **empty( )** method remove all child nodes from the set of matched elements whereas the **remove(expr)** method removes all matched elements from the DOM. Here is the syntax for the method:

```
selector.remove( [ expr ])


or


selector.empty( )
```

You can pass optional paramter *expr* to filter the set of elements to be removed.

Example

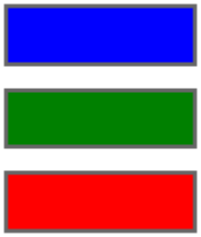
Following is an example where elements are being removed as soon as they are clicked:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });
```

```
   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <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>
</body>
</html>
```

This will produce the following result:

```
Click on any square below:
```



Inserting DOM Elements

There may be a situation when you would like to insert new one or more DOM elements in your existing document. JQuery provides various methods to insert elements at various locations. The **after(content)** method insert content after each of the matched elements where as the **before(content)** method inserts content before each of the matched elements. Here is the syntax for the method:

```
selector.after( content )
```

```
or


selector.before( content )
```

Here content is what you want to insert. This could be HTML or simple text.

Example

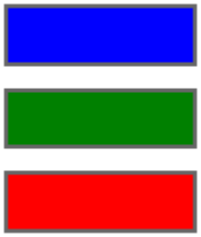
Following is an example where elements are being inserted just before the clicked element:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <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>
</body>
</html>
```

This will produce the following result:

```
Click on any square below:
```



DOM Manipulation Methods

Following table lists down all the methods which you can use to manipulate DOM elements:

S.No	Method	Description
1	after(content)	Insert content after each of the matched elements.
2	append(content)	Append content to the inside of every matched element.
3	appendTo(selector)	Append all of the matched elements to another, specified, set of elements.
4	before(content)	Insert content before each of the matched elements.
5	clone(bool)	Clone matched DOM Elements, and all their event handlers, and select the clones.
6	clone( )	Clone matched DOM Elements and select the clones.
7	empty( )	Remove all child nodes from the set of matched elements.

8	html(val)	Set the html contents of every matched element.
9	html( )	Get the html contents (innerHTML) of the first matched element.
10	insertAfter(selector)	Insert all of the matched elements after another, specified, set of elements.
11	insertBefore(selector)	Insert all of the matched elements before another, specified, set of elements.
12	prepend(content)	Prepend content to the inside of every matched element.
13	prependTo(selector)	Prepend all of the matched elements to another, specified, set of elements.
14	remove(expr)	Removes all matched elements from the DOM.
15	replaceAll(selector)	Replaces the elements matched by the specified selector with the matched elements.
16	replaceWith(content)	Replaces all matched elements with the specified HTML or DOM elements.
17	text(val)	Set the text contents of all matched elements.
18	text( )	Get the combined text contents of all matched elements.
19	wrap(elem)	Wrap each matched element with the specified element.
20	wrap(html)	Wrap each matched element with the specified HTML content.
21	wrapAll(elem)	Wrap all the elements in the matched set into a single wrapper element.

22	wrapAll(html)	Wrap all the elements in the matched set into a single wrapper element.
23	wrapInner(elem)	Wrap the inner child contents of each matched element (including text nodes) with a DOM element.
24	wrapInner(html)	Wrap the inner child contents of each matched element (including text nodes) with an HTML structure.

after(content) Method

The **after(content)** method inserts content after each of the matched elements.

Here is the simple syntax to use this method:

```
selector.after( content )
```

Parameters

Here is the description of all the parameters used by this method:

- **content:** Content to insert after each target. This could be HTML or Text content

Example

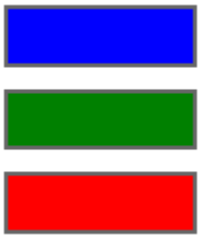
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });
```

```
   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



append(content) Method

The **append(content)** method appends content to the inside of every matched element.

Syntax

Here is the simple syntax to use this method

```
selector.append( content )
```

Parameters

Here is the description of all the parameters used by this method:

- **content:** Content to insert after each target. This could be HTML or Text content

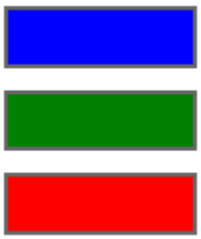
Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



appendTo(selector) Method

The **appendTo(selector)** method appends all of the matched elements to another, specified, set of elements.

Syntax

Here is the simple syntax to use this method:

```
selector.appendTo( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** This is the target to which the content will be appended.

Example

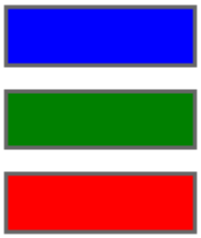
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       $(this).appendTo("#result");
    });

   });
```

```
   </script>
   <style>
      .div{ margin:10px;padding:12px;
            border:2px solid #666;
            width:60px;
          }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <p id="result"> THIS IS TEST </p>
   <hr />
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
THIS IS TEST
```



before(content) Method

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

Syntax

Here is the simple syntax to use this method:

```
selector.before( content )
```

Parameters

Here is the description of all the parameters used by this method:

- **content:** Content to insert before each target. This could be HTML or Text content

Example

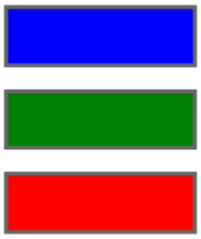
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
```

```
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



clone(bool) Method

The **clone(bool)** method clones matched DOM Elements, and all their event handlers, and select the clones.

This is useful for moving copies of the elements, and their events, to another location in the DOM.

Syntax

Here is the simple syntax to use this method:

```
selector.clone( bool )
```

Parameters

Here is the description of all the parameters used by this method:

- **bool:** Set to true to enable cloning of event handlers.

Example

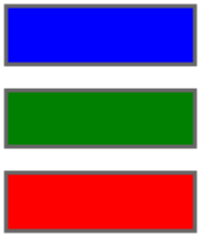
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
```

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

     $("div").click(function () {
       $(this).clone().insertAfter(this);
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

Click on any square below to see the result:



clone( ) Method

The **clone( )** method clones matched DOM Elements and select the clones. This is useful for moving copies of the elements to another location in the DOM.

Syntax

Here is the simple syntax to use this method:

```
selector.clone( )
```

Parameters

Here is the description of all the parameters used by this method:

- **NA:**

Example

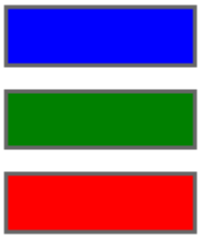
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       $(this).clone().insertAfter(this);
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
```

```
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



empty( ) Method

The **empty( )** method removes all child nodes from the set of matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.empty( )
```

Parameters

Here is the description of all the parameters used by this method:

- **NA:**

Example

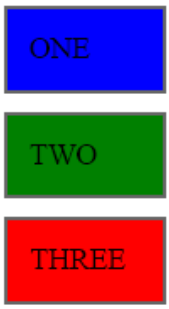
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
```

```
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;">ONE</div>
   <div class="div" style="background-color:green;">TWO</div>
   <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



html(val) Method

The **html(val)** method sets the html contents of every matched element. This property is not available on XML documents.

Syntax

Here is the simple syntax to use this method:

```
selector.html( val )
```

Parameters

Here is the description of all the parameters used by this method:

- **val:** This is the html content to be set.

Example

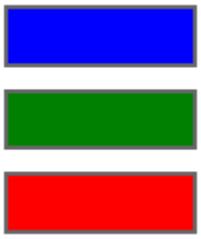
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       $(this).html( "<h1>Click another square</h1>");
    });

   });
```

```
   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



html( ) Method

The **html(val)** method gets the html contents (innerHTML) of the first matched element. This property is not available on XML documents.

Syntax

Here is the simple syntax to use this method:

```
selector.html( )
```

Parameters

Here is the description of all the parameters used by this method:

- **NA:**

Example

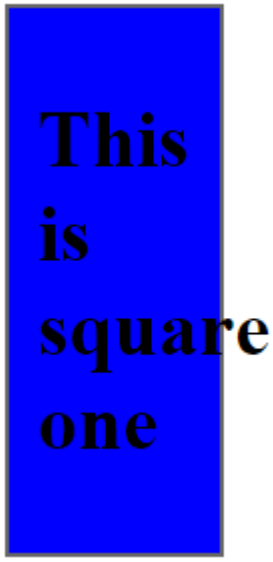
Following is a simple example showing the usage of this method:

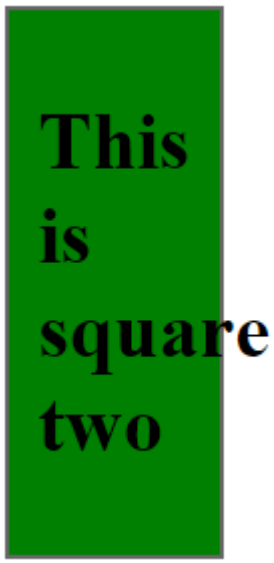
```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <p id="result"> THIS IS TEST </p>
   <div class="div" style="background-color:blue;">
       <h1>This is square one </h1>
   </div>
   <div class="div" style="background-color:green;">
```

```
        <h1>This is square two </h1>
    </div>
    <div class="div" style="background-color:red;">
        <h1>This is square three </h1>
    </div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
THIS IS TEST
```





insertAfter(selector) Method

The **insertAfter(selector)** method inserts all of the matched elements after another, specified, set of elements.

Syntax

Here is the simple syntax to use this method:

```
selector.insertAfter( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** Content after which the selected element(s) is inserted.

Example

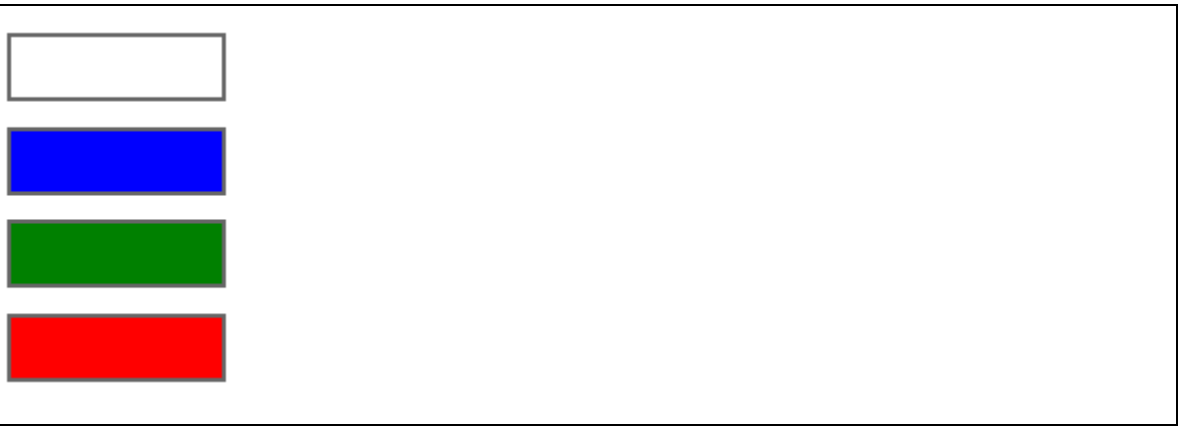
Following is a simple example showing the usage of this method. This inserts division element with ID of "source" after an element which is being clicked.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
```

```
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       $("#source").insertAfter(this);
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" id="source"></div>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



insertBefore(selector) Method

The **insertBefore(selector)** method inserts all of the matched elements before another, specified, set of elements.

Syntax

Here is the simple syntax to use this method:

```
selector.insertBefore( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** Content before which the selected element(s) is inserted.

Example

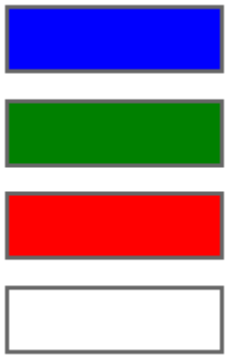
Following is a simple example showing the usage of this method. This inserts division element with ID of "source" before an element which is being clicked.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       $("#source").insertBefore(this);
```

```
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
   <div class="div" id="source"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



prepend(content) Method

The **prepend(content)** method prepends content to the inside of every matched element. Compare it with **append(content)** method.

Syntax

Here is the simple syntax to use this method:

```
selector.prepend( content )
```

Parameters

Here is the description of all the parameters used by this method:

- **content:** Content to insert after each target. This could be HTML or Text content

Example

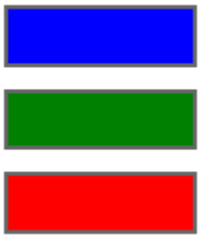
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

```
   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below:
```



prependTo(selector) Method

The **prependTo(selector)** method prepends all of the matched elements to another, specified, set of elements. Compare it with **appendTo(selector)** method.

Syntax

Here is he simple syntax to use this method:

```
selector.prependTo( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** This is the target to which the content will be prepended.

Example

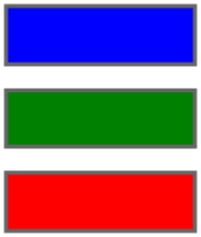
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       $(this).prependTo("#result");
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <p id="result"> THIS IS TEST </p>
   <hr />
   <div class="div" style="background-color:blue;"></div>
```

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

This will produce the following result:

```
Click on any square below:
```



remove(expr) Method

The remove(expr) method removes all matched elements from the DOM. This does NOT remove them from the jQuery object, allowing you to use the matched elements further. Compare it with **empty()** method.

Syntax

Here is the simple syntax to use this method:

```
selector.remove( expr )
```

Parameters

Here is the description of all the parameters used by this method:

- **expr:** This is an optional jQuery expression to filter the set of elements to be removed

Example

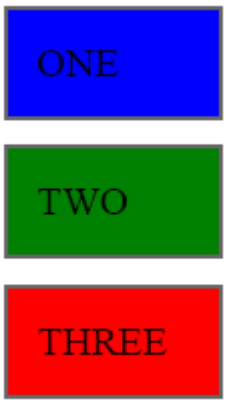
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
```

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

     $("div").click(function () {
       $(this).remove().appendTo("#result");
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <p id="result"> THIS IS TEST </p>
   <hr />
   <div class="div" style="background-color:blue;">ONE</div>
   <div class="div" style="background-color:green;">TWO</div>
   <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
THIS IS TEST
```



replaceAll(selector) Method

The **replaceAll(selector)** method replaces the elements matched by the specified selector with the matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.replaceAll( selector )
```

Parameters

Here is the description of all the parameters used by this method:

- **selector:** The elements to find and replace the matched elements with.

Example

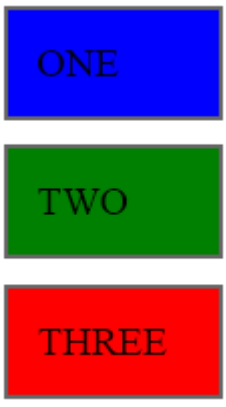
Following is a simple example showing the usage of this method. This replaces clicked one square by a new division:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

```
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;">ONE</div>
   <div class="div" style="background-color:green;">TWO</div>
   <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



replaceWith(content) Method

The **replaceWith(content)** method replaces all matched elements with the specified HTML or DOM elements. This returns the JQuery element that was just replaced, which has been removed from the DOM.

Syntax

Here is the simple syntax to use this method:

```
selector.replaceWith( content )
```

Parameters

Here is the description of all the parameters used by this method:

- **content :** Content to replace the matched elements with.

Example

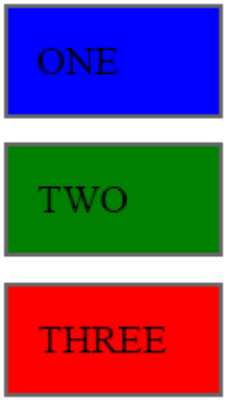
Following is a simple example showing the usage of this method. This replaces clicked one square by a new division. Compare the syntax with **replaceAll(selector)**method

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
```

```
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;">ONE</div>
   <div class="div" style="background-color:green;">TWO</div>
   <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



text(val) Method

The **text(val)** method gets the combined text contents of all matched elements. Similar to html(val), but escapes HTML (replace "<" and ">" with their HTML entities).

Syntax

Here is the simple syntax to use this method:

```
selector.text( val )
```

Parameters

Here is the description of all the parameters used by this method:

- **val:** This is the text value to be set.

Example

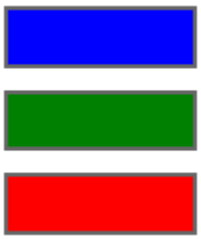
Following is a simple example showing the usage of this method:

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

```
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       $(this).text( "<h1>Click another square</h1>");
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



text( ) Method

The **html(val)** method gets the combined text contents of all matched elements.

The result is a string that contains the combined text contents of all matched elements. This method works on both HTML and XML documents.

Syntax

Here is the simple syntax to use this method:

```
selector.text( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA

Example

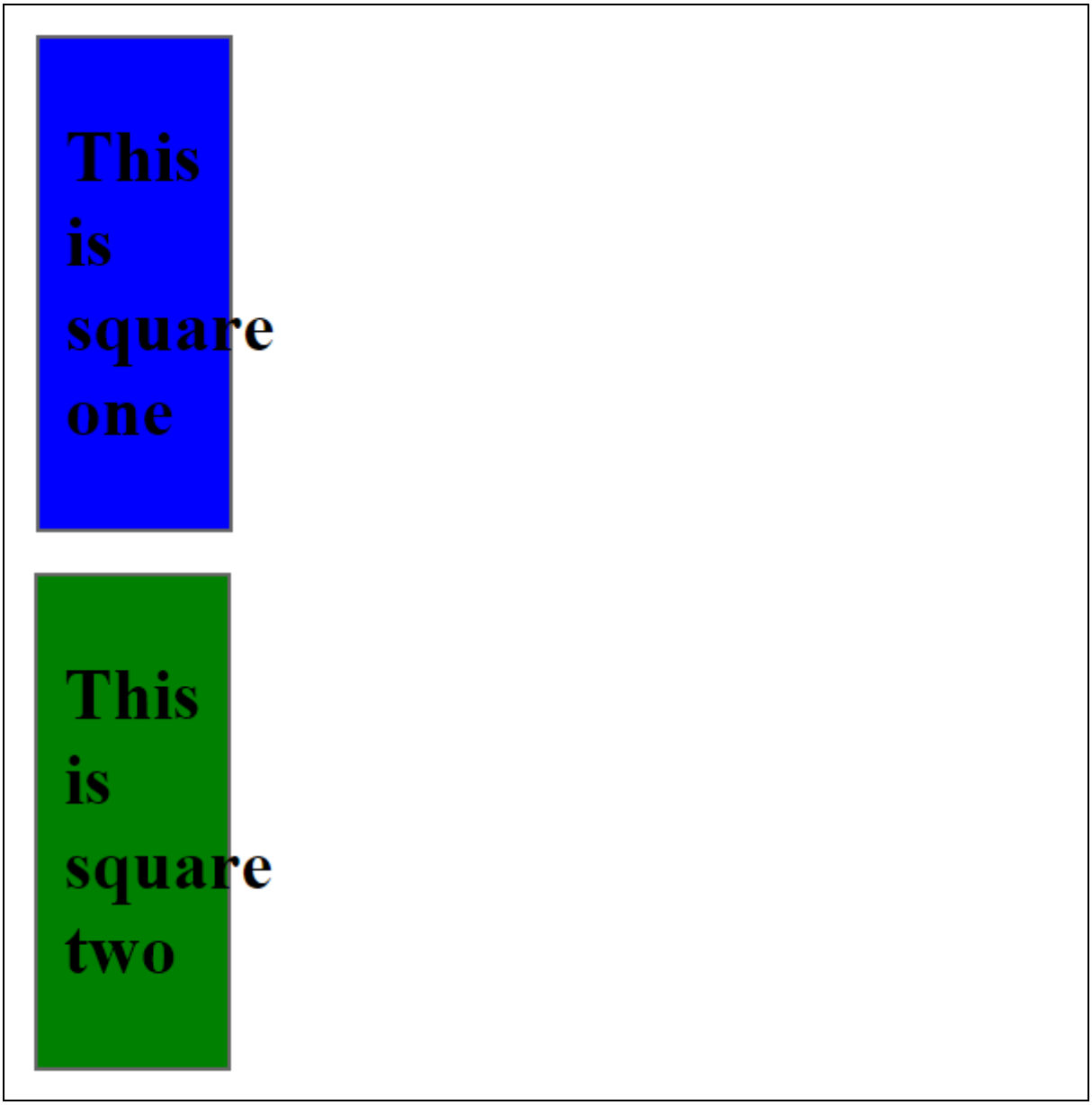
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

```
   });
   </script>
   <style>
      .div{ margin:10px;padding:12px;
            border:2px solid #666;
            width:60px;
          }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <p id="result"> THIS IS TEST </p>
   <div class="div" style="background-color:blue;">
       <h1>This is square one </h1>
   </div>
   <div class="div" style="background-color:green;">
       <h1>This is square two </h1>
   </div>
   <div class="div" style="background-color:red;">
       <h1>This is square three </h1>
   </div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
THIS IS TEST
```



wrap(elem) Method

The **wrap(elem)** method wraps each matched element with the specified element.

Syntax

Here is the simple syntax to use this method:

```
selector.wrap( elem )
```

Parameters

Here is the description of all the parameters used by this method:

- **elem :** A DOM element that will be wrapped around each target.

Example

Following is a simple example showing the usage of this method. This wraps destination square with a square when any square gets cli cked:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

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

```
    $("div").click(function () {
      var content = 'div class="div"';
      $("#destination").wrap(document.createElement(content));
    });

  });

  </script>
  <style>
     .div{ margin:10px;padding:12px;
            border:2px solid #666;
            width:60px;
          }
  </style>
</head>
<body>
  <p>Click on any square below to see the result:</p>
  <div class="div" id="destination">THIS IS TEST</div>
  <div class="div" style="background-color:blue;">ONE</div>
  <div class="div" style="background-color:green;">TWO</div>
  <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:

Click on any square below to see the result:

wrap(html) Method

The **wrap(html)** method wraps each matched element with the specified HTML content.

This wrapping process is most useful for injecting additional structure into a document, without ruining the original semantic qualities of a document.

Syntax

Here is the simple syntax to use this method:

```
selector.wrap( html )
```

Parameters

Here is the description of all the parameters used by this method:

- **elem :** A string of HTML that will be created on the fly and wrapped around each target.

Example

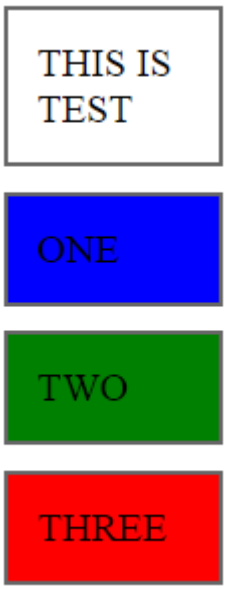
Following is a simple example showing the usage of this method. This wraps destination square with a sqaure when any square gets clicked:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       var content = '<div class="div"></div>';
       $("#destination").wrap( content );
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
```

```
            border:2px solid #666;
            width:60px;
          }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" id="destination">THIS IS TEST</div>
   <div class="div" style="background-color:blue;">ONE</div>
   <div class="div" style="background-color:green;">TWO</div>
   <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



wrapAll(elem) Method

The **wrapAll(elem)** method wraps all the elements in the matched set into a single wrapper element.

Here is the simple syntax to use this method:

```
selector.wrapAll( elem )
```

Parameters

Here is the description of all the parameters used by this method:

- **elem :** A DOM element that will be wrapped around the target.

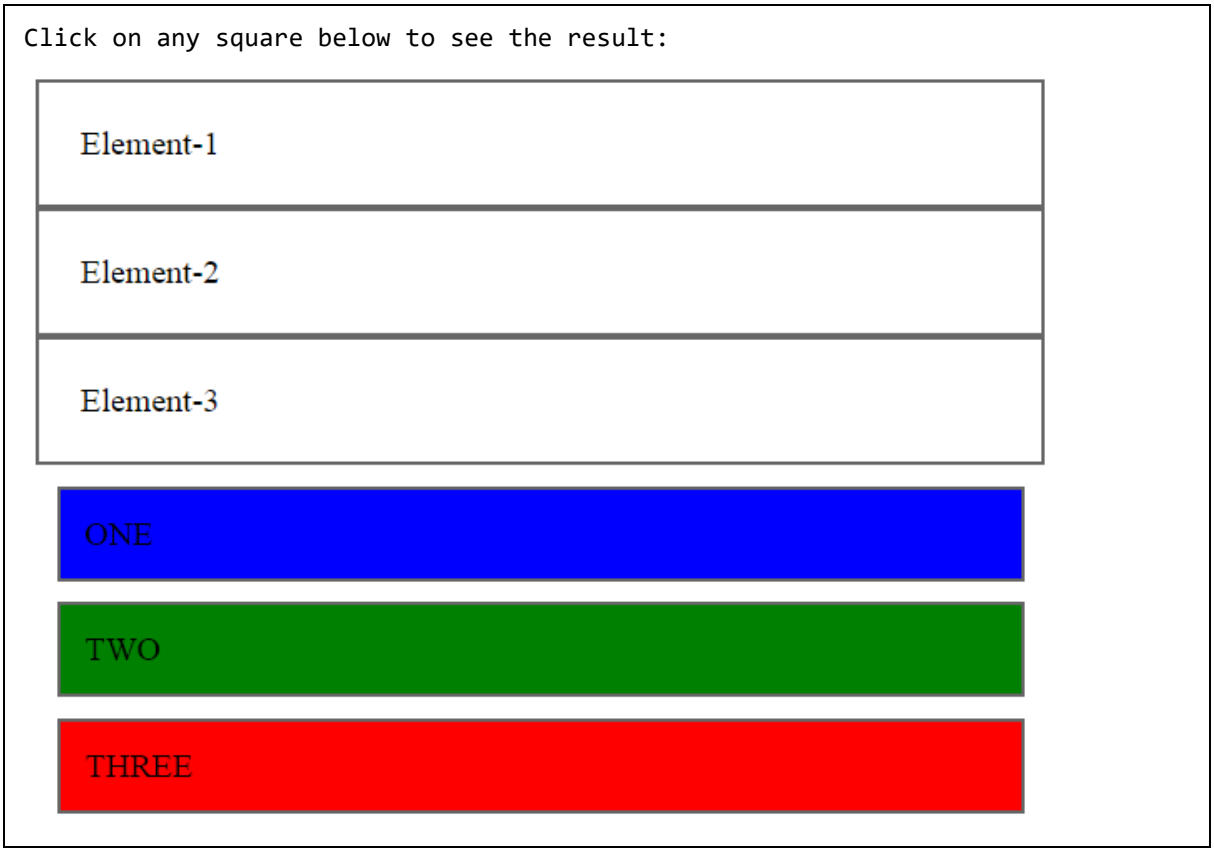
Example

Following is a simple example showing the usage of this method. This wraps all the squares with a new square:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       var content = "<div class='div'>";
       $("div").wrapAll( document.createElement(content) );
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" id="destination">THIS IS TEST</div>
   <div class="div" style="background-color:blue;">ONE</div>
   <div class="div" style="background-color:green;">TWO</div>
```

```
   <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:



wrapAll(html) Method

The **wrapAll(html)** method wraps all the elements in the matched set into a single wrapper element.

Here is the simple syntax to use this method:

```
selector.wrapAll( html )
```

Parameters

Here is the description of all the parameters used by this method:

- **html:** A string of HTML that will be created on the fly and wrapped around each target.

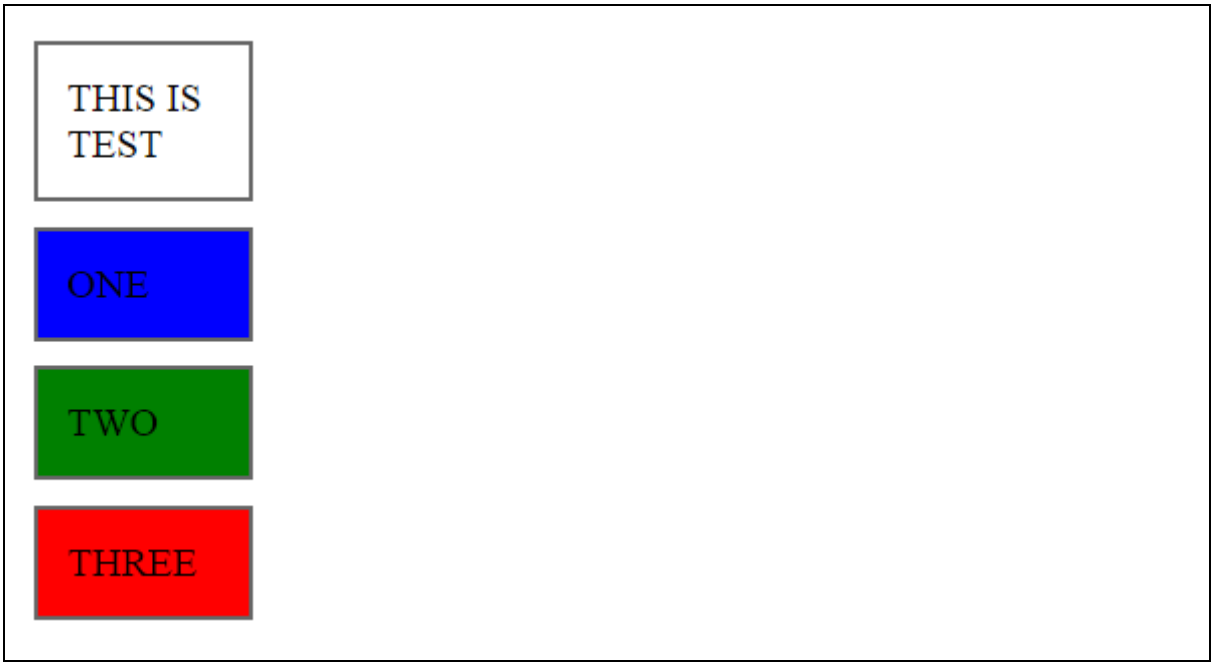
Example

Following is a simple example showing the usage of this method. This wraps all the squares with a new square:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $("div").click(function () {
       var content = "<div class='div'></div>";
       $("div").wrapAll( content );
    });
   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" id="destination">THIS IS TEST</div>
   <div class="div" style="background-color:blue;">ONE</div>
   <div class="div" style="background-color:green;">TWO</div>
   <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



wrapInner(elem) Method

The **wrapInner(elem)** method wraps the inner child contents of each matched element (including text nodes) with a DOM element.

Here is the simple syntax to use this method:

```
selector.wrapInner( elem )
```

Parameters

Here is the description of all the parameters used by this method:

- **html :** A DOM element that will be wrapped around the target.

Example

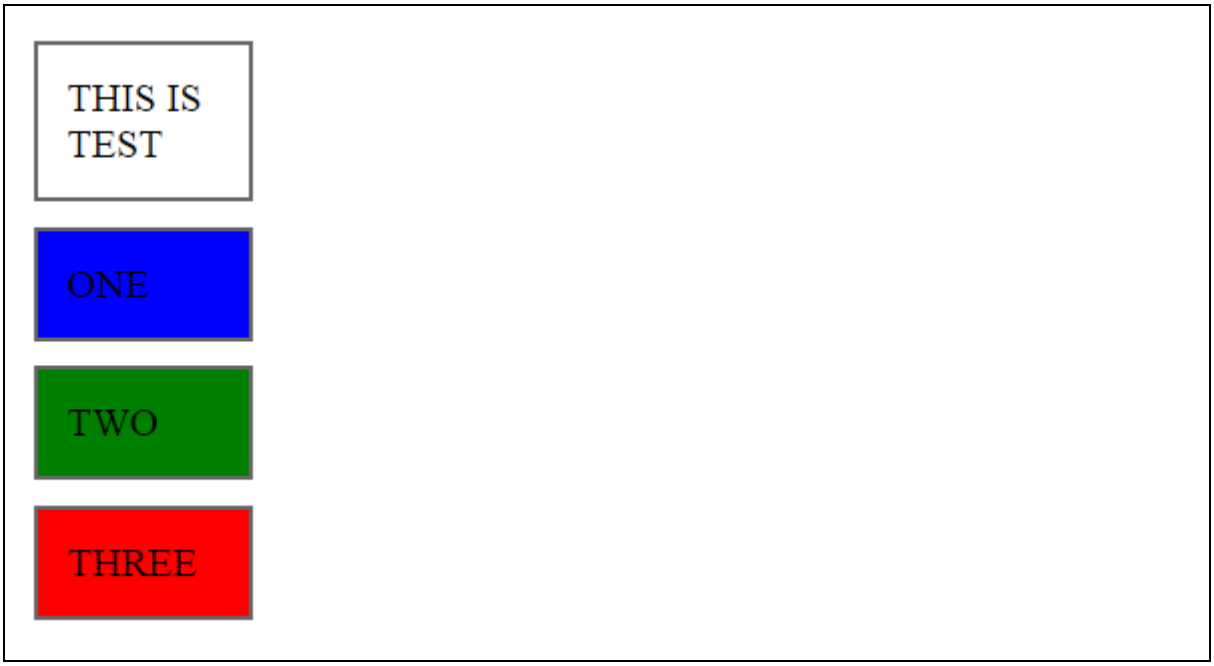
Following is a simple example showing the usage of this method. This selects all divisions and wraps a bold tag around the content of clicked one square:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

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

```
    $("div").click(function () {
      $(this).wrapInner(document.createElement( "b" ));
    });

  });

  </script>
  <style>
     .div{ margin:10px;padding:12px;
            border:2px solid #666;
            width:60px;
          }
  </style>
</head>
<body>
  <p>Click on any square below to see the result:</p>
  <div class="div" id="destination">THIS IS TEST</div>
  <div class="div" style="background-color:blue;">ONE</div>
  <div class="div" style="background-color:green;">TWO</div>
  <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



wrapInner(html) Method

The **wrapInner(html)** method wraps the inner child contents of each matched element (including text nodes) with an HTML structure.

Here is the simple syntax to use this method:

```
selector.wrapInner( html )
```

Parameters

Here is the description of all the parameters used by this method:

- **html :** A string of HTML that will be created on the fly and wrapped around the target.

Example

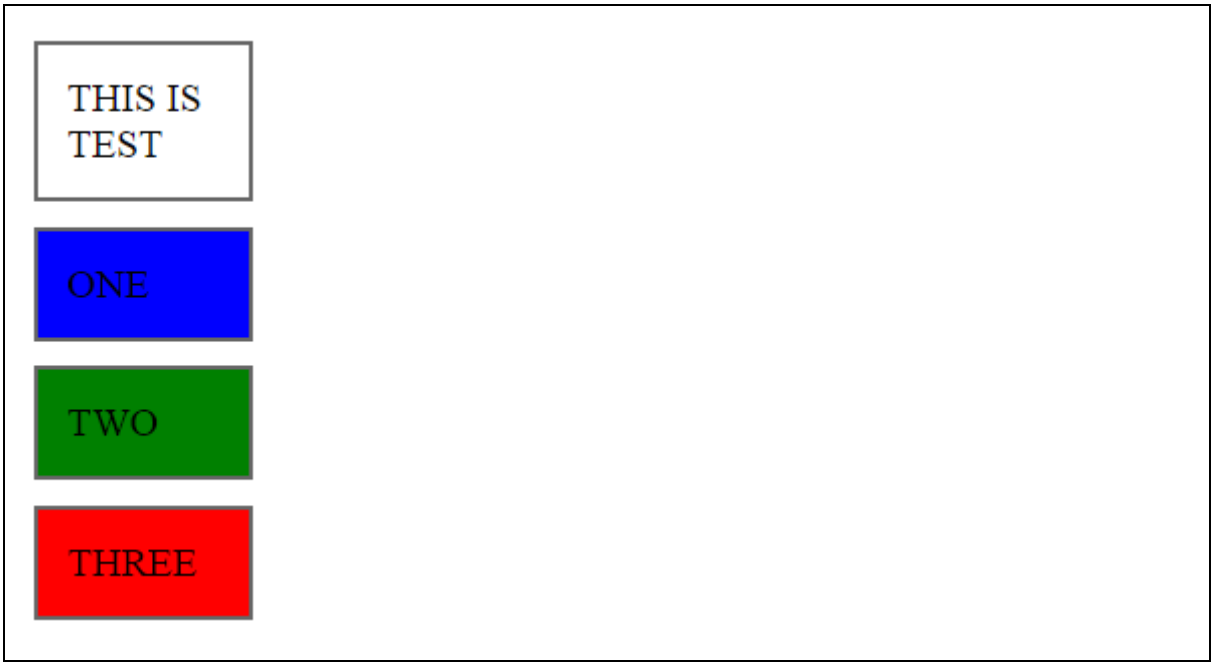
Following is a simple example showing the usage of this method. This selects all divisions and wraps a bold tag around the content of clicked one square:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
```

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

     $("div").click(function () {
       var content = "<b></b>";
       $(this).wrapInner( content );
    });

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" id="destination">THIS IS TEST</div>
   <div class="div" style="background-color:blue;">ONE</div>
   <div class="div" style="background-color:green;">TWO</div>
   <div class="div" style="background-color:red;">THREE</div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



8. EVENTS HANDLING

We have the ability to create dynamic web pages by using events. Events are actions that can be detected by your Web Application.

Following are the examples events:

- A mouse click
- A web page loading
- Taking mouse over an element
- Submitting an HTML form
- A keystroke on your keyboard, etc.

When these events are triggered, you can then use a custom function to do pretty much whatever you want with the event. These custom functions call Event Handlers.

Binding Event Handlers

Using the jQuery Event Model, we can establish event handlers on DOM elements with the **bind()** method as follows:

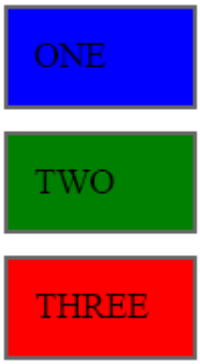
```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $('div').bind('click', function( event ){
               alert('Hi there!');
            });
         });
      </script>

      <style>
         .div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
      </style>

   </head>
```

```
   <body>

      <p>Click on any square below to see the result:</p>

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

   </body>

</html>
```

This code will cause the division element to respond to the click event; when a user clicks inside this division thereafter, the alert will be shown.

This will produce the following result:

```
Click on any square below to see the result:
```



The full syntax of the bind() command is as follows:

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

Following is the description of the parameters:

- **eventType:** A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.
- **eventData:** This optional parameter is a map of data that will be passed to the event handler.
- **handler:** A function to execute each time the event is triggered.

Removing Event Handlers

Typically, once an event handler is established, it remains in effect for the remainder of the life of the page. There may be a need when you would like to remove event handler. jQuery provides the **unbind()** command to remove an exiting event handler. The syntax of unbind() is as follows:

```
selector.unbind(eventType, handler)


or


selector.unbind(eventType)
```

Following is the description of the parameters:

- **eventType:** A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.
- **handler:** If provided, identifies the specific listener that's to be removed.

Event Types

The following are cross platform and recommended event types which you can bind using JQuery:

S. NO.	Event Type & Description
1	**blur** Occurs when the element loses focus.
2	**change** Occurs when the element changes.
3	**click** Occurs when a mouse click.
4	**dblclick** Occurs when a mouse double-click.
5	**Error** Occurs when there is an error in loading or unloading etc.

6	**Focus** Occurs when the element gets focus.
7	**keydown** Occurs when key is pressed.
8	**keypress** Occurs when key is pressed and released.
9	**keyup** Occurs when key is released.
10	**Load** Occurs when document is loaded.
11	**mousedown** Occurs when mouse button is pressed.
12	**mouseenter** Occurs when mouse enters in an element region.
13	**mouseleave** Occurs when mouse leaves an element region.
14	**Mousemove** Occurs when mouse pointer moves.
15	**Mouseout** Occurs when mouse pointer moves out of an element.
16	**Mouseover** Occurs when mouse pointer moves over an element.
17	**Mouseup** Occurs when mouse button is released.

18	**Resize** Occurs when window is resized.
19	**Scroll** Occurs when window is scrolled.
20	**Select** Occurs when a text is selected.
21	**Submit** Occurs when form is submitted.
22	**Unload** Occurs when documents is unloaded.

The Event Object

The callback function takes a single parameter; when the handler is called the JavaScript event object will be passed through it.

The event object is often unnecessary and the parameter is omitted, as sufficient context is usually available when the handler is bound to know exactly what needs to be done when the handler is triggered, however there are certain attributes which you would need to be accessed.

The Event Attributes

The following event properties/attributes are available and safe to access in a platform independent manner:

S. NO.	Property & Description
1	**altKey** Set to true if the Alt key was pressed when the event was triggered, false if not. The Alt key is labeled Option on most Mac keyboards.
2	**ctrlKey** Set to true if the Ctrl key was pressed when the event was triggered, false if not.

3	**data** The value, if any, passed as the second parameter to the bind() command when the handler was established.
4	**keyCode** For keyup and keydown events, this returns the key that was pressed.
5	**metaKey** Set to true if the Meta key was pressed when the event was triggered, false if not. The Meta key is the Ctrl key on PCs and the Command key on Macs.
6	**pageX** For mouse events, specifies the horizontal coordinate of the event relative from the page origin.
7	**pageY** For mouse events, specifies the vertical coordinate of the event relative from the page origin.
8	**relatedTarget** For some mouse events, identifies the element that the cursor left or entered when the event was triggered.
9	**screenX** For mouse events, specifies the horizontal coordinate of the event relative from the screen origin.
10	**screenY** For mouse events, specifies the vertical coordinate of the event relative from the screen origin.
11	**shiftKey** Set to true if the Shift key was pressed when the event was triggered, false if not.
12	**target** Identifies the element for which the event was triggered.

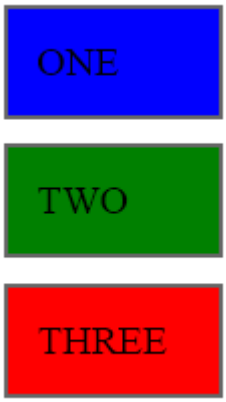
13	**Timestamp** The timestamp (in milliseconds) when the event was created.
14	**type** For all events, specifies the type of event that was triggered (for example, click).
15	**which** For keyboard events, specifies the numeric code for the key that caused the event, and for mouse events, specifies which button was pressed (1 for left, 2 for middle, 3 for right).

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $('div').bind('click', function( event ){
               alert('Event type is ' + event.type);
               alert('pageX : ' + event.pageX);
               alert('pageY : ' + event.pageY);
               alert('Target : ' + event.target.innerHTML);
            });
         });
      </script>

      <style>
         .div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
      </style>

   </head>

   <body>
```

```
      <p>Click on any square below to see the result:</p>

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

   </body>

</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



The Event Methods

There is a list of methods which can be called on an Event Object:

SNo	Method	Description
1	preventDefault()	Prevents the browser from executing the default action.
2	isDefaultPrevented()	Returns whether event.preventDefault() was ever called on this event object.
3	stopPropagation()	Stops the bubbling of an event to parent elements, preventing any parent handlers from being notified of the event.

4	isPropagationStopped()	Returns whether event.stopPropagation() was ever called on this event object.
5	stopImmediatePropagation()	Stops the rest of the handlers from being executed.
6	isImmediatePropagationStopped()	Returns whether event.stopImmediatePropagation() was ever called on this event object.

preventDefault() Method

The **preventDefault()** method prevents the browser from executing the default action.

You can use the method **isDefaultPrevented** to know whether this method was ever called (on that event object).

Syntax

Here is the simple syntax to use this method:

```
event.preventDefault()
```

Parameters

Here is the description of all the parameters used by this method:

- **NA**

Example

Following is a simple example showing the usage of this method. This example demonstrate how you can stop the browser from changing the page to the href of any anchors.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

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

```
      $("a").click(function(event){
         event.preventDefault();
         alert( "Default behavior is disabled!" );
      });

   });
   </script>
</head>
<body>
   <span>Click the following link and it won't work:</span>
   <a href="http://www.google.com">GOOGLE Inc.</a>
</body>
</html>
```

This will produce the following result:

```
Click the following link and it won't work: GOOGLE Inc.
```

isDefaultPrevented() Method

The **isDefaultPrevented()** method checks whether event.preventDefault() was ever called on this event object.

This method returns true in case preventDefault() has been called otherwise it returns false.

Syntax

Here is the simple syntax to use this method:

```
event.isDefaultPrevented()
```

Parameters

Here is the description of all the parameters used by this method:

- **NA**

Example

Following is a simple example showing the usage of this method. This example demonstrate how you can stop the browser from changing the page to the href of any anchors.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {


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

         if ( event.isDefaultPrevented() ){
            alert( "Default behavior is disabled - 1" );
         }else{
            alert( "Default behavior is enabled - 1" );
         }
         event.preventDefault();
         if ( event.isDefaultPrevented() ){
            alert( "Default behavior is disabled - 2" );
         }else{
            alert( "Default behavior is enabled - 2" );
         }

      });

   });
   </script>
</head>
<body>
   <span>Click the following link and it won't work:</span>
   <a href="http://www.google.com">GOOGLE Inc.</a>
</body>
</html>
```

This will produce the following result:

```
Click the following link and it won't work: GOOGLE Inc.
```

stopPropagation() Method

The **stopPropagation()** method stops the bubbling of an event to parent elements, preventing any parent handlers from being notified of the event.

You can use the method **event.isPropagationStopped()** to know whether this method was ever called (on that event object).

Syntax

Here is the simple syntax to use this method:

```
event.stopPropagation()
```

Parameters

Here is the description of all the parameters used by this method:

- **NA**

Example

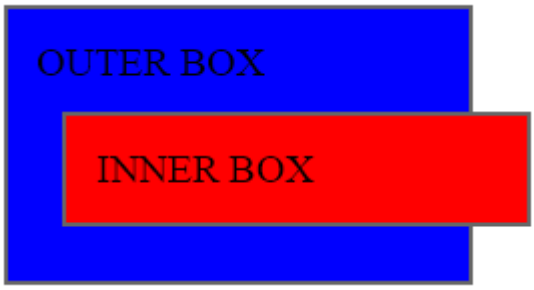
Following is a simple example showing the usage of this method. This example demonstrate how you can prevent other event handlers from being called:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

      $("div").click(function(event){
          alert("This is : " + $(this).text());
          // Comment the following to see the difference
          event.stopPropagation();
      });

   });
```

```
   </script>
   <style>
      div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:160px;
           }
  </style>
</head>
<body>
   <p>Click on any box to see the effect:</p>
   <div id="div1" style="background-color:blue;">
       OUTER BOX
       <div id="div2" style="background-color:red;">
             INNER BOX
      </div>
  </div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



isPropagationStopped() Method

The **isPropagationStopped()** method checks whether event.stopPropagation() was ever called on this event object.

This method returns true in case **event.stopPropagation()** method has been already called, otherwise it returns false.

Syntax

Here is the simple syntax to use this method:

```
event.isPropagationStopped()
```

Parameters

Here is the description of all the parameters used by this method:

- **NA**

Example

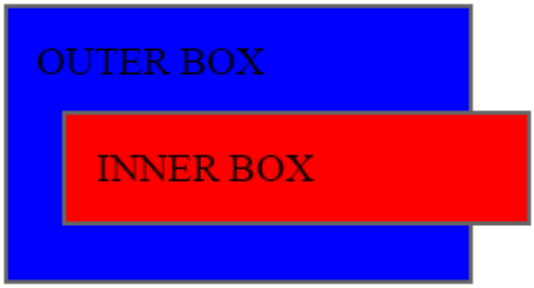
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

      $("div").click(function(event){
          alert("This is : " + $(this).text());
          if ( event.isPropagationStopped() ){
             alert( "Event bubbling is disabled - 1" );
          }else{
             alert( "Event bubbling is enabled - 1" );
          }
          event.stopPropagation();
          if ( event.isPropagationStopped() ){
             alert( "Event bubbling is disabled - 2" );
          }else{
             alert( "Event bubbling is enabled - 2" );
          }
      });

   });
   </script>
   <style>
      div{ margin:10px;padding:12px;
```

```
            border:2px solid #666;
            width:160px;
          }
  </style>
</head>
<body>
   <p>Click on any box to see the effect:</p>
   <div id="div1" style="background-color:blue;">
       OUTER BOX
       <div id="div2" style="background-color:red;">
            INNER BOX
      </div>
  </div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



stopImmediatePropagation() Method

The **stopImmediatePropagation()** method stops the rest of the handlers from being executed. This method also stops the bubbling by calling event.stopPropagation(). You can use **event.isImmediatePropagationStopped()** to know whether this method was ever called (on that event object).

Syntax

Here is the simple syntax to use this method:

```
event.stopImmediatePropagation()
```

Parameters

Here is the description of all the parameters used by this method:

- **NA**

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

      $("div").click(function(event){
          alert("1 - This is : " + $(this).text());
          // Comment the following to see the effect.
          event.stopImmediatePropagation();
      });

      // This won't be executed.
      $("div").click(function(event){
          alert("2 - This is : " + $(this).text());
      });

   });
   </script>
   <style>
      div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:160px;
           }
  </style>
</head>
<body>
   <p>Click on any box to see the result:</p>
```

```
   <div id="div1" style="background-color:blue;">
       BOX 1
  </div>
  <div id="div2" style="background-color:red;">
         BOX 2
  </div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```

isImmediatePropagationStopped() Method

The **isImmediatePropagationStopped()** method checks whether event.stopImmediatePropagation() was ever called on this event object. This method returns true in case **event.stopImmediatePropagation()** method has already been called, otherwise it returns false:

Syntax

Here is the simple syntax to use this method:

```
event.isImmediatePropagationStopped()
```

Parameters

Here is the description of all the parameters used by this method:

- **NA**

Example

Following is a simple example showing the usage of this method:

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

```
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

          if ( event.isImmediatePropagationStopped() ){
             alert( "Event bubbling is disabled - 1" );
          }else{
             alert( "Event bubbling is enabled - 1" );
          }
          event.stopImmediatePropagation();
          if ( event.isImmediatePropagationStopped() ){
             alert( "Event bubbling is disabled - 2" );
          }else{
             alert( "Event bubbling is enabled - 2" );
          }

      });

   });
   </script>
   <style>
      div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:160px;
           }
  </style>
</head>
<body>
   <p>Click on any box to see the result:</p>
   <div id="div1" style="background-color:blue;">
       BOX 1
```

```
  </div>
  <div id="div2" style="background-color:red;">
       BOX 2
  </div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```

BOX 1

BOX 2

Event Manipulation Methods

Following table lists down important event-related methods:

S.No.	Method & Description
1	bind(type, [data], fn) Binds a handler to one or more events (like click) for each matched element. Can also bind custom events.
2	off(events [, selector] [, handler(eventObject)]) This does the opposite of live, it removes a bound live event.
3	hover(over, out) Simulates hovering for example moving the mouse on, and off, an object.
4	on(events [, selector] [, data], handler)

	Binds a handler to an event (like click) for all current – and future – matched element. Can also bind custom events.
5	one(type, [data], fn) Binds a handler to one or more events to be executed once for each matched element.
6	ready(fn) Binds a function to be executed whenever the DOM is ready to be traversed and manipulated.
7	trigger(event, [data]) Trigger an event on every matched element.
8	triggerHandler(event, [data]) Triggers all bound event handlers on an element.
9	unbind([type], [fn]) This does the opposite of bind, it removes bound events from each of the matched elements.

bind(type, [data], fn) Method

The **bind(type, [data], fn)** method binds a handler to one or more events (like click) for each matched element. Can also bind custom events.

Possible event values: blur, focus, load, resize, scroll, unload, click etc.

Syntax

Here is the simple syntax to use this method:

```
selector.bind( type, [data], fn )
```

Parameters

Here is the description of all the parameters used by this method:

- **type:** One or more event types separated by a space.
- **data:** This is optional paramter and represents additional data passed to the event handler as event.data.
- **fn:** A function to bind to the event on each of the set of matched elements.

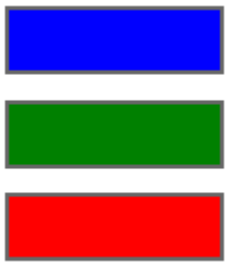
Example

Following is a simple example showing the usage of this method. Here it binds click event with each element:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



off(events [, selector] [, handler(eventObject)]) Method

The **off(events [, selector] [, handler(eventObject)])** method does the opposite of **on()** method, it removes a bound live event.

Syntax

Here is the simple syntax to use this method −

```
selector.on( event, selector, handler )
```

Parameter

Here is the description of all the parameters used by this method −

- **events −** Event types separated by spaces.
- **selector −** A Selector String
- **handler −** A function to bind to the event on each of the set of matched elements

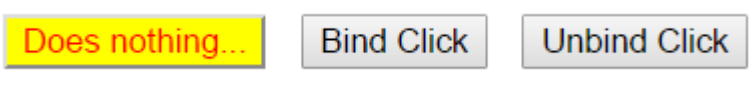
Example

Following is a simple example showing the usage of this method.

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {

            function aClick() {
               $("div").show().fadeOut("slow");
            }
```

```
            $("#bind").click(function () {
               $("#theone").on("click", aClick).text("Can Click!");
            });

            $("#unbind").click(function () {
               $("#theone").off("click", aClick).text("Does nothing...");
            });
         });
      </script>

      <style>
         button { margin:5px; }
         button#theone { color:red; background:yellow; }
      </style>

   </head>

   <body>

      <button id="theone">Does nothing...</button>
      <button id="bind">Bind Click</button>
      <button id="unbind">Unbind Click</button>

      <div style="display:none;">Click!</div>

   </body>

</html>
```

This will produce the following result:



hover(over, out) Method

The **hover(over, out)** method simulates hovering (moving the mouse on, and off, an object). This is a custom method which provides an 'in' to a frequent task.

Syntax

Here is the simple syntax to use this method:

```
selector.hover( over, out )
```

Parameters

Here is the description of all the parameters used by this method:

- **over:** The callback function to fire when the mouse is moved over a matched element.
- **out:** The callback function to fire when the mouse is moved off of a matched element.

Example

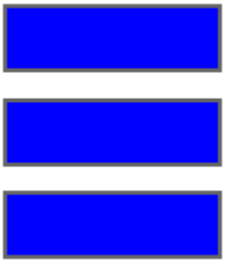
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

     $('div').hover(
         function () {
           $(this).css({"background-color":"red"});
         },
         function () {
           $(this).css({"background-color":"blue"});
         }
     );

   });
```

```
   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Move mouse on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:blue;"></div>
</body>
</html>
```

This will produce the following result:

```
Move mouse on any square below to see the result:
```



on(events [, selector] [, data], handler) Method

The **on(events [, selector] [, data], handler)** method binds a handler to an event (like click) for all current – and future – matched element. Can also bind custom events. Possible event values – blur, focus, load, resize, scroll, unload, click etc.

Syntax

Here is the simple syntax to use this method –

```
selector.on( event, selector, data, handler )
```

Parameter

Here is the description of all the parameters used by this method –

- **events –** Event types separated by spaces.
- **selector –** A Selector String
- **data –** Data to be passed to the event handler in event.data
- **handler –** A function to bind to the event on each of the set of matched elements

Example

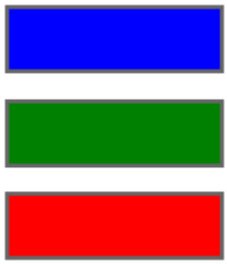
Following is a simple example showing the usage of this method. Here it binds click event with each element –

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $('div').on('click', function( event ){
               alert('Hi there!');
            });
         });
      </script>

      <style>
         .div{ margin:10px;padding:12px; border:2px solid #666; width:60px;}
      </style>

   </head>

   <body>

      <p>Click on any square below to see the result:</p>

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

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

   </body>

</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



one(type, [data], fn) Method

The **one(type, [data], fn)** method binds a handler to one or more events to be executed once for each matched element. The handler is executed only once for each element. Otherwise, the same rules as described in bind() apply.

Possible event values: blur, focus, load, resize, scroll, unload, click etc.

Syntax

Here is the simple syntax to use this method:

```
selector.one( type, [data], fn )
```

Parameters

Here is the description of all the parameters used by this method:

- **type:** An event type.
- **data:** This is optional paramter and represents additional data passed to the event handler as event.data.
- **fn:** A function to bind to the event on each of the set of matched elements.

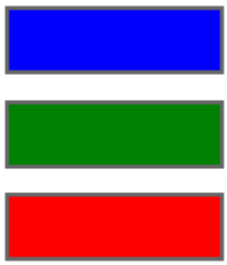
Example

Following is a simple example showing the usage of this method. Here it binds click event with each element. Try to click any square two times, it won't react unlike bind() method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

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

   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <p>Click on any square below to see the result:</p>
   <div class="div" style="background-color:blue;"></div>
   <div class="div" style="background-color:green;"></div>
   <div class="div" style="background-color:red;"></div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



ready(fn) Method

The **ready(fn)** method binds a function to be executed whenever the DOM is ready to be traversed and manipulated. This method is a replacement for using window.onload

Syntax

Here is the simple syntax to use this method:

```
selector.ready( fn )
```

Parameters

Here is the description of all the parameters used by this method:

- **fn:** The function to be executed when the DOM is ready.

Example

Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {
      $("div").text("The DOM is now loaded...");
   });

   </script>
   <style>
      .div{ margin:10px;padding:12px;
            border:2px solid #666;
```

```
            width:200px;
          }
  </style>
</head>
<body>
   <div class="div" style="background-color:blue;"></div>
</body>
</html>
```

This will produce the following result:

trigger(event, [data]) Method

The **trigger(event, [data])** method triggers an event on every matched element.

Triggered events aren't limited to browser-based events, you can also trigger custom events registered with bind.

Syntax

Here is the simple syntax to use this method:

```
selector.trigger( event, [data] )
```

Parameters

Here is the description of all the parameters used by this method:

- **event:** An event object or type to trigger.
- **data :** This is an optional parameters and represents additional data to pass as arguments (after the event object) to the event handler.

Example

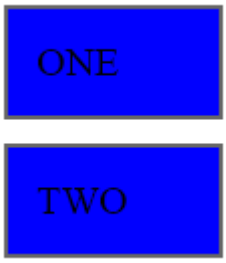
Following is a simple example showing the usage of this method. Here you would trigger a click event on square TWO by clicking on square ONE:

```
<html>
<head>
<title>the title</title>
```

```
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

      $("#div1").click( function () {
           $("#div2").trigger('click');
      });

      $("#div2").click( function () {
           alert( "Square TWO is clicked");
      });

   });
   </script>
   <style>
      div{ margin:10px;padding:12px;
             border:2px solid #666;
             width:60px;
           }
  </style>
</head>
<body>
   <span>Click square ONE to see the result:</span>
   <div id="div1" style="background-color:blue;">ONE</div>
   <div id="div2" style="background-color:blue;">TWO</div>
</body>
</html>
```

This will produce the following result:

```
Click on any square below to see the result:
```



triggerHandler(event, [data]) Method

The **triggerHandler(event, [data])** method triggers all bound event handlers on an element (for a specific event type) WITHOUT executing the browser's default actions, bubbling, or live events.

This method behaves very similarly to the trigger method, with two major exceptions:

- **First:** No default browser actions are triggered, the triggered event does not bubble, and live events aren't triggered.
- **Second:** The event is only triggered on the first element within the jQuery collection.

This method returns the return value of the triggered handler instead of a chainable jQuery object.

Syntax

Here is the simple syntax to use this method:

```
selector.triggerHandler( event, [data] )
```

Parameters

Here is the description of all the parameters used by this method:

- **event:** An event object or type to trigger.
- **data :** This is an optional parameters and represents additional data to pass as arguments (after the event object) to the event handler.

Example

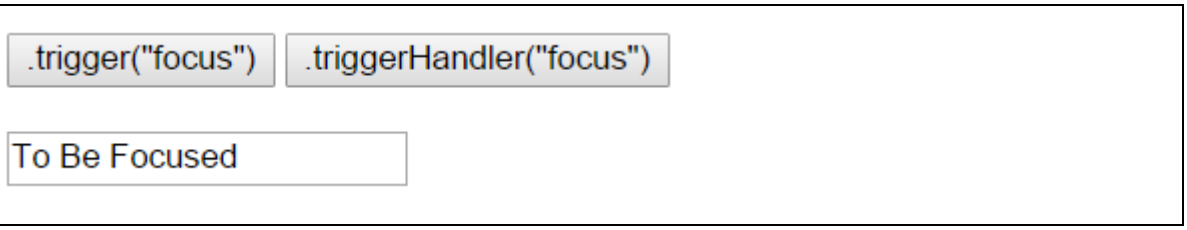
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
```

```
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

      $("#old").click(function(){
         $("input").trigger("focus");
      });

      $("#new").click(function(){
         $("input").triggerHandler("focus");
      });

      $("input").focus(function(){
       $("<span>Focused!</span>").appendTo("body").fadeOut(1000);
      });

   });
   </script>
</head>
<body>
   <button id="old">.trigger("focus")</button>
   <button id="new">.triggerHandler("focus")</button><br/><br/>
   <input type="text" value="To Be Focused"/>
</body>
</html>
```

This will produce the following result:

Click on any square below to see the result:



unbind([type], [fn]) Method

The **unbind([type], [fn])** method does the opposite of bind, it removes bound events from each of the matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.unbind( [type], [fn] )
```

Parameters

Here is the description of all the parameters used by this method:

- **type:** One or more event types separated by a space.
- **fn:** A function to unbind from the event on each of the set of matched elements.

Example

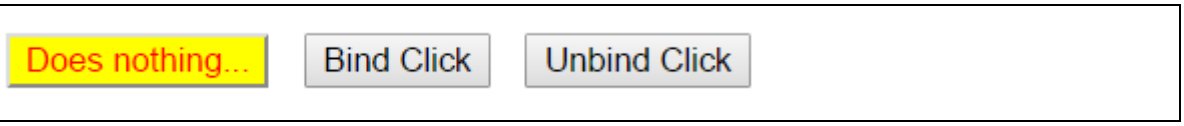
Following is a simple example showing the usage of this method:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">

   $(document).ready(function() {

       function aClick() {
         $("div").show().fadeOut("slow");
       }
       $("#bind").click(function () {
         $("#theone").click(aClick)
                   .text("Can Click!");
       });
       $("#unbind").click(function () {
         $("#theone").unbind('click', aClick)
                   .text("Does nothing...");
       });
   });

   </script>
   <style>
```

```
      button { margin:5px; }
      button#theone { color:red; background:yellow; }
  </style>
</head>
<body>
  <button id="theone">Does nothing...</button>
  <button id="bind">Bind Click</button>
  <button id="unbind">Unbind Click</button>
  <div style="display:none;">Click!</div>
</body>
</html>
```

This will produce the following result:

Click on any square below to see the result:



Event Helper Methods

jQuery also provides a set of event helper functions which can be used either to trigger an event to bind any event types mentioned above.

Trigger Methods

Following is an example which would triggers the blur event on all paragraphs:

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

Binding Methods

Following is an example which would bind a **click** event on all the :

```
$("div").click( function () {
   // do something here
});
```

Here is a complete list of all the support methods provided by jQuery:

S.No.	Method & Description

1	**blur( )** Triggers the blur event of each matched element.
2	**blur(fn)** Bind a function to the blur event of each matched element.
3	**change( )** Triggers the change event of each matched element.
4	**change(fn)** Binds a function to the change event of each matched element.
5	**click( )** Triggers the click event of each matched element.
6	**click(fn)** Binds a function to the click event of each matched element.
7	**dblclick( )** Triggers the dblclick event of each matched element.
8	**dblclick(fn)** Binds a function to the dblclick event of each matched element.
9	**error( )** Triggers the error event of each matched element.
10	**error(fn)** Binds a function to the error event of each matched element.
11	**focus( )** Triggers the focus event of each matched element.
12	**focus(fn)** Binds a function to the focus event of each matched element.

13	**keydown( )** Triggers the keydown event of each matched element.
14	**keydown(fn)** Bind a function to the keydown event of each matched element.
15	**keypress( )** Triggers the keypress event of each matched element.
16	**keypress(fn)** Binds a function to the keypress event of each matched element.
17	**keyup( )** Triggers the keyup event of each matched element.
18	**keyup(fn)** Bind a function to the keyup event of each matched element.
20	**load(fn)** Binds a function to the load event of each matched element.
21	**mousedown(fn)** Binds a function to the mousedown event of each matched element.
22	**mouseenter(fn)** Bind a function to the mouseenter event of each matched element.
23	**mouseleave(fn)** Bind a function to the mouseleave event of each matched element.
24	**mousemove(fn)** Bind a function to the mousemove event of each matched element.
25	**mouseout(fn)** Bind a function to the mouseout event of each matched element.

26	**mouseover(fn)** Bind a function to the mouseover event of each matched element.
27	**mouseup(fn)** Bind a function to the mouseup event of each matched element.
28	**resize(fn)** Bind a function to the resize event of each matched element.
29	**scroll(fn)** Bind a function to the scroll event of each matched element.
30	**select( )** Trigger the select event of each matched element.
31	**select(fn)** Bind a function to the select event of each matched element.
32	**submit( )** Trigger the submit event of each matched element.
33	**submit(fn)** Bind a function to the submit event of each matched element.
34	**unload(fn)** Binds a function to the unload event of each matched element.

9. AJAX

AJAX is an acronym standing for Asynchronous JavaScript and XML and this technology helps us to load data from the server without a browser page refresh.

If you are new with AJAX, I would recommend you go through our Ajax Tutorial before proceeding further.

JQuery is a great tool which provides a rich set of AJAX methods to develop next generation web application.

Loading Simple Data

This is very easy to load any static or dynamic data using JQuery AJAX. JQuery provides **load()** method to do the job:

Syntax

Here is the simple syntax for **load()** method:

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

Here is the description of all the parameters:

- **URL:** The URL of the server-side resource to which the request is sent. It could be a CGI, ASP, JSP, or PHP script which generates data dynamically or out of a database.

- **data:** This optional parameter represents 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.

Example

Consider the following HTML file with a small JQuery coding:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
   $(document).ready(function() {
```

```
      $("#driver").click(function(event){
          $('#stage').load('/jquery/result.html');
      });
   });
   </script>
</head>
<body>
   <p>Click on the button to load result.html file:</p>
   <div id="stage" style="background-color:blue;">
          STAGE
   </div>
   <input type="button" id="driver" value="Load Data" />
</body>
</html>
```

Here **load()** initiates an Ajax request to the specified URL **/jquery/result.html** file. After loading this file, all the content would be populated inside tagged with ID *stage*. Assuming, our /jquery/result.html file has just one HTML line:

```
   <h1>THIS IS RESULT...</h1>
```

When you click the given button, then result.html file gets loaded.

```
Click on the button to load result.html file
```

STAGE

Load Data

Getting JSON Data

There would be a situation when server would return JSON string against your request. JQuery utility function **getJSON()** parses the returned JSON string and makes the resulting string available to the callback function as first parameter to take further action.

Syntax

Here is the simple syntax for **getJSON()** method:

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

Here is the description of all the parameters:

- **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.

Example

Consider the following HTML file with a small JQuery coding:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
   $(document).ready(function() {
      $("#driver").click(function(event){
          $.getJSON('/jquery/result.json', function(jd) {
             $('#stage').html('<p> Name: ' + jd.name + '</p>');
             $('#stage').append('<p>Age : ' + jd.age+ '</p>');
             $('#stage').append('<p> Sex: ' + jd.sex+ '</p>');
          });
      });
   });
   </script>
</head>
<body>
   <p>Click on the button to load result.html file:</p>
   <div id="stage" style="background-color:blue;">
          STAGE
   </div>
   <input type="button" id="driver" value="Load Data" />
</body>
</html>
```

Here JQuery utility method **getJSON()** initiates an Ajax request to the specified URL**/jquery/result.json** file. After loading this file, all the content would be passed to the callback function which finally would be populated inside tagged with ID *stage*. Assuming, our /jquery/result.json file has following json formatted content:

```
{
"name": "Zara Ali",
"age" : "67",
"sex": "female"
}
```

When you click the given button, then result.html file gets loaded.

```
Click on the button to load result.html file
```

STAGE

Load Data

Passing Data to the Server

Many times you collect input from the user and you pass that input to the server for further processing. JQuery AJAX made it easy enough to pass collected data to the server using **data** parameter of any available Ajax method.

Example

This example demonstrates how user can pass input to a web server script which would send the same result back and we would print it:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
   $(document).ready(function() {
      $("#driver").click(function(event){
          var name = $("#name").val();
          $("#stage").load('/jquery/result.php', {"name":name} );
      });
   });
   </script>
</head>
<body>
   <p>Enter your name and click on the button:</p>
   <input type="input" id="name" size="40" /><br />
```

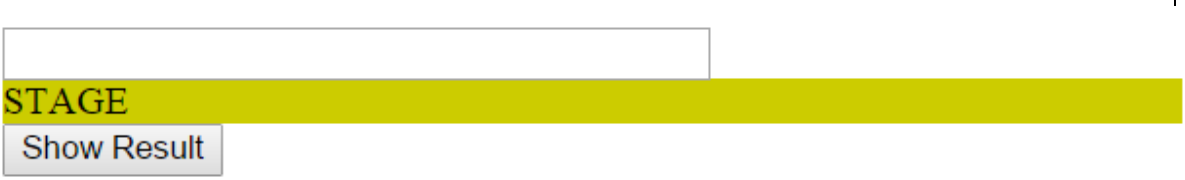
```
   <div id="stage" style="background-color:blue;">
          STAGE
   </div>
   <input type="button" id="driver" value="Show Result" />
</body>
</html>
```

Here is the code written in **result.php** script:

```
<?php
if( $_REQUEST["name"] )
{
   $name = $_REQUEST['name'];
   echo "Welcome ". $name;
}
?>
```

Now you can enter any text in the given input box and then click "Show Result" button to see what you have entered in the input box.

```
Enter your name and click on the button
```



JQuery AJAX Methods

You have seen basic concept of AJAX using JQuery. Following table lists down all important JQuery AJAX methods which you can use based your programming need:

S.No.	Methods & Description
1	jQuery.ajax(options) Load a remote page using an HTTP request.
2	jQuery.ajaxSetup(options) Setup global settings for AJAX requests.

3	jQuery.get(url, [data], [callback], [type]) Load a remote page using an HTTP GET request.
4	jQuery.getJSON(url, [data], [callback]) Load JSON data using an HTTP GET request.
5	jQuery.getScript(url, [callback]) Loads and executes a JavaScript file using an HTTP GET request.
6	jQuery.post(url, [data], [callback], [type]) Load a remote page using an HTTP POST request.
7	load(url, [data], [callback]) Load HTML from a remote file and inject it into the DOM.
8	serialize( ) Serializes a set of input elements into a string of data.
9	serializeArray( ) Serializes all forms and form elements like the .serialize() method but returns a JSON data structure for you to work with.

jQuery.ajax(options) Method

The **jQuery.ajax(options)** method loads a remote page using an HTTP request. $.ajax() returns the XMLHttpRequest that it creates. In most cases you won't need that object to manipulate directly, but it is available if you need to abort the request manually.

Syntax

Here is the simple syntax to use this method:

```
$.ajax( options )
```

Parameters

Here is the description of all the parameters used by this method:

- **options**: A set of key/value pairs that configure the Ajax request. All options are optional.

Here is the list of option which could be used as key/value pairs. Except URL, rest of the parameters are optional:

S.No.	Option & Description
1	**async** A Boolean indicating whether to perform the request asynchronously. The default value is true.
2	**beforeSend** A callback function that is executed before the request is sent.
3	**complete** A callback function that executes whenever the request finishes.
4	**contentType** A string containing a MIME content type to set for the request. The default value is application/x-www-form-urlencoded.
5	**data** A map or string that is sent to the server with the request.
6	**dataFilter** A function to be used to handle the raw responsed data of XMLHttpRequest. This is a pre-filtering function to sanitize the response.
7	**dataType** A string defining the type of data expected back from the server (xml, html, json, or script).
8	**error** A callback function that is executed if the request fails.
9	**Global** A Boolean indicating whether global AJAX event handlers will be triggered by this request. The default value is true.
10	**ifModified**

	A Boolean indicating whether the server should check if the page is modified before responding to the request.
11	**Jsonp** Override the callback function name in a jsonp request.
12	**Password** A password to be used in response to an HTTP access authentication request.
13	**processData** A Boolean indicating whether to convert the submitted data from an object form into a query-string form. The default value is true.
14	**success** A callback function that is executed if the request succeeds.
15	**Timeout** Number of milliseconds after which the request will time out in failure.
16	**timeout** Set a local timeout (in milliseconds) for the request.
17	**type** A string defining the HTTP method to use for the request (GET or POST). The default value is GET.
18	**url** A string containing the URL to which the request is sent.
19	**username** A username to be used in response to an HTTP access authentication request.
20	**xhr** Callback for creating the XMLHttpRequest object. Defaults to the ActiveXObject when available (IE), the XMLHttpRequest otherwise.

Example

Assuming we have following HTML content in /jquery/result.html file:

```
<h1>THIS IS RESULT...</h1>
```

Following is a simple example showing the usage of this method. Here we make use of *success* handler to populate returned HTML:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
   $(document).ready(function() {
      $("#driver").click(function(event){
          $.ajax( {
             url:'/jquery/result.html',
             success:function(data) {
                $('#stage').html(data);
             }
          });
      });
   });
   </script>
</head>
<body>
   <p>Click on the button to load result.html file:</p>
   <div id="stage" style="background-color:blue;">
          STAGE
   </div>
   <input type="button" id="driver" value="Load Data" />
</body>
</html>
```

This will produce the following result:

```
Click on the button to load result.html file:
```

STAGE

Load Data

ajaxSetup(options) Method

The **jQuery.ajaxSetup(options)** method sets global settings for future AJAX requests.

Syntax

Here is the simple syntax to use this method:

```
$.ajaxSetup( options )
```

Parameters

Here is the description of all the parameters used by this method:

- **options**: A set of key/value pairs that configure the Ajax request. All options are optional.

Here is the list of option which could be used as key/value pairs. Except URL, rest of the parameters are optional:

S.No.	Option & Description
1	**async** A Boolean indicating whether to perform the request asynchronously. The default value is true.
2	**beforeSend** A callback function that is executed before the request is sent.
3	**complete** A callback function that executes whenever the request finishes.
4	**contentType** A string containing a MIME content type to set for the request. The default value is application/x-www-form-urlencoded.
5	**data** A map or string that is sent to the server with the request.
6	**dataFilter** A function to be used to handle the raw responsed data of XMLHttpRequest. This is a pre-filtering function to sanitize the response.

7	**dataType** A string defining the type of data expected back from the server (xml, html, json, or script).
8	**error** A callback function that is executed if the request fails.
9	**Global** A Boolean indicating whether global AJAX event handlers will be triggered by this request. The default value is true.
10	**ifModified** A Boolean indicating whether the server should check if the page is modified before responding to the request.
11	**Jsonp** Override the callback function name in a jsonp request.
12	**password** A password to be used in response to an HTTP access authentication request.
13	**processData** A Boolean indicating whether to convert the submitted data from an object form into a query-string form. The default value is true.
14	**success** A callback function that is executed if the request succeeds.
15	**Timeout** Number of milliseconds after which the request will time out in failure.
16	**timeout** Set a local timeout (in milliseconds) for the request.
17	**Type** A string defining the HTTP method to use for the request (GET or POST). The default value is GET.

18	**url** A string containing the URL to which the request is sent.
19	**username** A username to be used in response to an HTTP access authentication request.
20	**xhr** Callback for creating the XMLHttpRequest object. Defaults to the ActiveXObject when available (IE), the XMLHttpRequest otherwise.

Example

Assuming we have following HTML content in /jquery/result.html file:

```
<h1>THIS IS RESULT...</h1>
```

Following is a simple example showing the usage of this method. Here we make use of *success* handler to populate returned HTML:

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
   $(document).ready(function() {
      $("#driver").click(function(event){
          // Do global setting.
          $.ajaxSetup({
             url: "/jquery/result.html"
          });
          $.ajax( {
             success:function(data) {
                $('#stage').html(data);
             }
          });
      });
   });
   </script>
```

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

This will produce the following result:

```
Click on the button to load result.html file:
```

STAGE

Load Data

get(url, [data], [callback], [type]) Method

The **jQuery.get(url, [data], [callback], [type])** method loads data from the server using a GET HTTP request. The method returns XMLHttpRequest object.

Syntax

Here is the simple syntax to use this method:

```
$.get( url, [data], [callback], [type] )
```

Parameters

Here is the description of all the parameters used by this method:

- **url**: A string containing the URL to which the request is sent
- **data:**: This optional parameter represents key/value pairs that will be sent to the server.
- **callback:**: This optional parameter represents a function to be executed whenever the data is loaded successfully.
- **type:**: This optional parameter represents type of data to be returned to callback function: "xml", "html", "script", "json", "jsonp", or "text".

Example

Assuming we have following PHP content in /jquery/result.php file:

```
<?php
if( $_REQUEST["name"] )
{
   $name = $_REQUEST['name'];
   echo "Welcome ". $name;
}
?>
```

Following is a simple example showing the usage of this method:

getJSON(url, [data], [callback]) Method

The **jQuery.getJSON(url, [data], [callback])** method loads JSON data from the server using a GET HTTP request. The method returns XMLHttpRequest object.

Syntax

Here is the simple syntax to use this method:

```
$.getJSON( url, [data], [callback] )
```

Parameters

Here is the description of all the parameters used by this method:

- **url**: A string containing the URL to which the request is sent
- **data:**: This optional parameter represents key/value pairs that will be sent to the server.
- **callback:**: This optional parameter represents a function to be executed whenever the data is loaded successfully.

Example

Assuming we have following JSON content in /jquery/result.json file

```
{
"name": "Zara Ali",
"age" : "67",
"sex": "female"
}
```

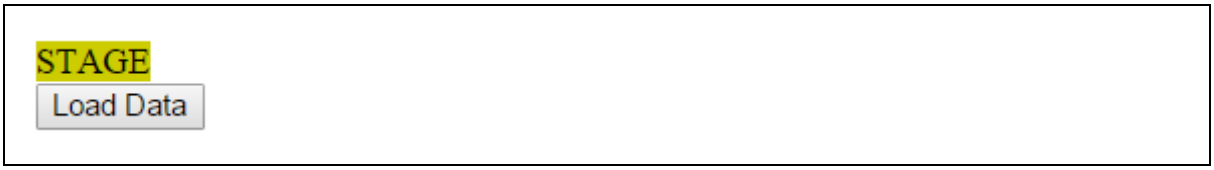
Following is a simple example showing the usage of this method:

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

```
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("#driver").click(function(event){
               $.get(
                  "result.php",
                  { name: "Zara" },
                  function(data) {
                     $('#stage').html(data);
                  }
               );
            });
         });
      </script>

   </head>

   <body>

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

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

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

   </body>

</html>
```

This should produce the following result:

Click on the button to load result.html file:



getScript(url, [callback]) Method

The **jQuery.getScript(url, [callback])** method loads and executes a JavaScript file using an HTTP GET request. The method returns XMLHttpRequest object.

Syntax

Here is the simple syntax to use this method:

```
$.getScript( url, [callback] )
```

Parameters

Here is the description of all the parameters used by this method:

- **url**: A string containing the URL to which the request is sent
- **callback:**: This optional parameter represents a function to be executed whenever the data is loaded successfully.

Example

Assuming we have following JavaScript content in /jquery/result.js file:

```
function CheckJS(){
   alert("This is JavaScript");
}
```

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("#driver").click(function(event){
               $.getScript('result.js', function(jd) {
                  // Call custom function defined in script
```

```
                CheckJS();
            });
        });
    });
</script>

</head>

<body>

    <p>Click on the button to load result.js file -</p>

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

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

</body>

</html>
```

This should produce the following result

```
Click on the button to load result.html file:
```

STAGE

Load Data

post(url, data, callback, type) Method

The **jQuery.post(url, [data], [callback], [type])** method loads a page from the server using a POST HTTP request. The method returns XMLHttpRequest object.

Syntax

Here is the simple syntax to use this method:

```
$.post( url, [data], [callback], [type] )
```

Parameters

Here is the description of all the parameters used by this method:

- **url**: A string containing the URL to which the request is sent
- **data:**: This optional parameter represents key/value pairs or the return value of the .serialize() function that will be sent to the server.
- **callback:**: This optional parameter represents a function to be executed whenever the data is loaded successfully.
- **type:**: This optional parameter represents a type of data to be returned to callback function: "xml", "html", "script", "json", "jsonp", or "text".

Example

Assuming we have the following PHP content in /jquery/result.php file:

```
<?php
if( $_REQUEST["name"] )
{
   $name = $_REQUEST['name'];
   echo "Welcome ". $name;
}
?>
```

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {

            $("#driver").click(function(event){
```

```
            $.post(
               "result.php",
               { name: "Zara" },
               function(data) {
                  $('#stage').html(data);
               }
            );

         });
      });
   </script>

</head>

<body>

   <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" />

</body>

</html>
```

This should produce the following result

Click on the button to load result.html file

STAGE

Load Data

load(url, data, callback) Method

The **load(url, data, callback)** method loads data from the server and places the returned HTML into the matched element.

Syntax

Here is the simple syntax to use this method:

```
[selector].load( url, [data], [callback] )
```

Parameters

Here is the description of all the parameters used by this method:

- **url**: A string containing the URL to which the request is sent.
- **data**: This optional parameter represents a map of data that is sent with the request.
- **callback**: This optional parameter represents a function that is executed if the request succeeds

Example

Assuming we have the following HTML content in /jquery/result.html file:

```
<h1>THIS IS RESULT...</h1>
```

Following is a simple example showing the usage of this method.

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("#driver").click(function(event){
               $('#stage').load('result.html');
            });
         });
      </script>

   </head>
```

```
   <body>

      <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" />

   </body>

</html>
```

This should produce the following result

```
Click on the button to load result.html file
```

serialize( ) Method

The **serialize( )** method serializes a set of input elements into a string of data.

Syntax

Here is the simple syntax to use this method:

```
$.serialize( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA

Example

Assuming we have the following PHP content in /jquery/serialize.php file:

```
<?php
if( $_REQUEST["name"] )
```

```
{
   $name = $_REQUEST['name'];
   echo "Welcome ". $name;
   $age = $_REQUEST['age'];
   echo "<br />Your age : ". $age;
   $sex = $_REQUEST['sex'];
   echo "<br />Your gender : ". $sex;
}
?>
```

Following is a simple example showing the usage of this method:

```
</html>
 <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {

            $("#driver").click(function(event){

               $.post(
                  "serialize.php",
                  $("#testform").serialize(),

                  function(data) {
                     $('#stage1').html(data);
                  }

               );

               var str = $("#testform").serialize();
               $("#stage2").text(str);
            });
```

```
      });
   </script>

</head>

<body>

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

   <div id="stage1" style="background-color:blue;">
      STAGE - 1
   </div>
   <br />

   <div id="stage2" style="background-color:blue;">
      STAGE - 2
   </div>

   <form id="testform">
      <table>

         <tr>
            <td><p>Name:</p></td>
            <td><input type="text" name="name" size="40" /></td>
         </tr>

         <tr>
            <td><p>Age:</p></td>
            <td><input type="text" name="age" size="40" /></td>
         </tr>

         <tr>

            <td><p>Sex:</p></td>

            <td> <select name="sex">
```

```
                  <option value="Male" selected>Male</option>
                  <option value="Female" selected>Female</option>
               </select></td>

            </tr>

            <tr>
               <td colspan="2">
                  <input type="button" id="driver" value="Load Data" />
               </td>
            </tr>

         </table>

      </form>

   </body>

</html>
```

This will produce the following result

```
Click on the button to load result.html file:
```

STAGE - 1

STAGE - 2

Name:

Age:

Sex: Female

Load Data

serializeArray( ) Method

The **serializeArray( )** method serializes all forms and form elements like the .serialize() method but returns a JSON data structure for you to work with. The JSON structure returned is not a string. You must use a plugin or third-party library to "stringify".

Syntax

Here is the simple syntax to use this method:

```
$.serializeArray( )
```

Parameters

Here is the description of all the parameters used by this method:

- NA

Example

Assuming we have the following PHP content in /jquery/serialize.php file:

```
<?php
if( $_REQUEST["name"] )
{
   $name = $_REQUEST['name'];
   echo "Welcome ". $name;
   $age = $_REQUEST['age'];
   echo "<br />Your age : ". $age;
   $sex = $_REQUEST['sex'];
   echo "<br />Your gender : ". $sex;
}
?>
```

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("#driver").click(function(event){
```

```
            $.post(
               "serialize.php",
               $("#testform").serializeArray(),
               function(data) {
                  $('#stage1').html(data);
               }
            );
            var fields = $("#testform").serializeArray();
            $("#stage2").empty();
            jQuery.each(fields, function(i, field){
               $("#stage2").append(field.value + " ");
            });
         });
      });
   </script>

</head>

<body>

   <p>Click on the button to load result.html file:</p>
   <div id="stage1" style="background-color:blue;">
      STAGE - 1
   </div>

   <br />
   <div id="stage2" style="background-color:blue;">
      STAGE - 2
   </div>

   <form id="testform">
      <table>
         <tr>
            <td><p>Name:</p></td>
            <td><input type="text" name="name" size="40" /></td>
         </tr>
```

```
            <tr>
               <td><p>Age:</p></td>
               <td><input type="text" name="age" size="40" /></td>
            </tr>

            <tr>
               <td><p>Sex:</p></td>
               <td> <select name="sex">
                  <option value="Male" selected>Male</option>
                  <option value="Female" selected>Female</option>
               </select></td>
            </tr>

            <tr>
               <td colspan="2">
                  <input type="button" id="driver" value="Load Data" />
               </td>
            </tr>
         </table>
      </form>
   </body>
</html>
```

This will produce the following result

Click on the button to load result.html file:

STAGE - 1

STAGE - 2

Name:

Age:

Sex: Female

Load Data

JQuery AJAX Events

You can call various JQuery methods during the life cycle of AJAX call progress. Based on different events/stages, the following methods are available. You can go through all the AJAX Events.

S.No.	Methods & Description
1	ajaxComplete(callback) Attach a function to be executed whenever an AJAX request completes.
2	ajaxStart(callback) Attach a function to be executed whenever an AJAX request begins and there is none already active.
3	ajaxError(callback) Attach a function to be executed whenever an AJAX request fails.
4	ajaxSend(callback) Attach a function to be executed before an AJAX request is sent.
5	ajaxStop(callback) Attach a function to be executed whenever all AJAX requests have ended.
6	ajaxSuccess(callback) Attach a function to be executed whenever an AJAX request completes successfully.

ajaxComplete(callback) Method

The **ajaxComplete(callback)** method attaches a function to be executed whenever an AJAX request completes. This is an Ajax Event.

Syntax

Here is the simple syntax to use this method:

```
$(document).ajaxComplete( )
```

Parameters

Here is the description of all the parameters used by this method:

- **callback**: The function to execute. The XMLHttpRequest and settings used for that request are passed as arguments to this function.

Example

Assuming we have the following HTML content in /jquery/result.html file

```
<h1>THIS IS RESULT...</h1>
```

Following is a simple example showing the usage of this method.

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("#driver").click(function(event){
               $('#stage1').load('result.html');
            });

            $(document).ajaxComplete(function(event, request, settings){
               $("#stage2").html("<h1>Request Complete.</h1>");
            });
         });
      </script>

   </head>

   <body>

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

      <div id="stage1" style="background-color:blue;">
         STAGE - 1
      </div>

      <div id="stage2" style="background-color:blue;">
```

```
         STAGE - 2
      </div>

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

   </body>

</html>
```

This will produce the following result

```
Click on the button to load result.html file:
```

ajaxStart(callback) Method

The **ajaxStart(callback)** method attaches a function to be executed whenever an AJAX request begins and there is none already active. This is an Ajax Event.

Syntax

Here is the simple syntax to use this method:

```
$(document).ajaxStart( callback )
```

Parameters

Here is the description of all the parameters used by this method:

- **callback**: The function to execute.

Example

Assuming we have the following HTML content in /jquery/result.html file:

```
<h1>THIS IS RESULT...</h1>
```

Following is a simple example showing the usage of this method.

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

```
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            /* Global variable */
            var count = 0;

            $("#driver").click(function(event){
               $('#stage1').load('result.html');
            });
            /* Gets called when request starts */

            $(document).ajaxStart(function(){
               count++;
               $("#stage2").html("<h1>Starts, Count :" + count + "</h1>");
            });

            /* Gets called when request complete */
            $(document).ajaxComplete(function(event,request,set){
               count++;
               $("#stage3").html("<h1>Completes,Count:" + count + "</h1>");
            });
         });
      </script>

   </head>

   <body>

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

      <div id="stage1" style="background-color:blue;">
         STAGE - 1
      </div>
```

```
      <div id="stage2" style="background-color:blue;">
         STAGE - 2
      </div>

      <div id="stage3" style="background-color:blue;">
         STAGE - 3
      </div>

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

   </body>

</html>
```

This will produce the following result

```
Click on the button to load result.html file:
```

STAGE - 1
STAGE - 2
STAGE - 3
Load Data

ajaxError(callback) Method

The **ajaxError(callback)** method attaches a function to be executed whenever an AJAX request fails. This is an Ajax Event.

Syntax

Here is a simple syntax to use this method:

```
$(document).ajaxError( callback )
```

Parameters

Here is the description of all the parameters used by this method:

- **callback**: The function to execute. The XMLHttpRequest and settings used for that request are passed as arguments to this function. A third argument, an exception object, is passed if an exception occurred while processing the request.

Example

Following is a simple example showing the usage of this method.

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $("#driver").click(function(event){
               /* Assume result.text does not exist. */
               $('#stage1').load('/jquery/result.text');
            });

            $(document).ajaxError(function(event, request, settings ){
               $("#stage2").html("<h1>Error in loading page.</h1>");
            });
         });
      </script>

   </head>

   <body>

      <p>Click on the button to load result.text file:</p>

      <div id="stage1" style="background-color:blue;">
         STAGE - 1
      </div>

      <div id="stage2" style="background-color:blue;">
         STAGE - 2
      </div>

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

```
   </body>


</html>
```

This will produce the following result

```
Click on the button to load result.html file:
```

ajaxSend(callback) Method

The **ajaxSend(callback)** method attaches a function to be executed whenever an AJAX request is sent. This is an Ajax Event.

Syntax

Here is the simple syntax to use this method:

```
$(document).ajaxSend( callback )
```

Parameters

Here is the description of all the parameters used by this method:

- **callback**: The function to execute. The XMLHttpRequest and settings used for that request are passed as arguments to the callback.

Example

Assuming we have following HTML content in /jquery/result.html file:

```
<h1>THIS IS RESULT...</h1>
```

Following is a simple example showing the usage of this method.

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
```

```
   <script type="text/javascript" language="javascript">
      $(document).ready(function() {
         /* Global variable */
         var count = 0;

         $("#driver").click(function(event){
            $('#stage0').load('result.html');
         });

         /* Gets called when request starts */
         $(document).ajaxStart(function(){
            count++;
            $("#stage1").html("<h1>Starts, Count :" + count + "</h1>");
         });

         /* Gets called when request is sent */
         $(document).ajaxSend(function(evt, req, set){
            count++;
            $("#stage2").html("<h1>Sends, Count :" + count + "</h1>");
            $("#stage2").append("<h1>URL :" + set.url  + "</h1>");
         });

         /* Gets called when request complete */
         $(document).ajaxComplete(function(event,request,settings){
            count++;
            $("#stage3").html("<h1>Completes, Count :" + count + "</h1>");
         });
      });
   </script>

</head>

<body>

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

```
      <div id="stage0" style="background-color:blue;">
         STAGE - 0
      </div>

      <div id="stage1" style="background-color:blue;">
         STAGE - 1
      </div>

      <div id="stage2" style="background-color:blue;">
         STAGE - 2
      </div>

      <div id="stage3" style="background-color:blue;">
         STAGE - 3
      </div>

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

   </body>

</html>
```

This will produce the following result:

```
Click on the button to load result.html file:
```

STAGE - 0
STAGE - 1
STAGE - 2
STAGE - 3

Load Data

ajaxStop(callback) Method

The **ajaxStop(callback)** method attaches a function to be executed whenever all AJAX requests have ended. This is an Ajax Event.

Syntax

Here is the simple syntax to use this method:

```
$[selector].ajaxStop( callback )
```

Parameters

Here is the description of all the parameters used by this method:

- **callback**: The function to execute.

Example

Assuming we have following HTML content in /jquery/result.html file:

```
<h1>THIS IS RESULT...</h1>
```

Following is a simple example showing the usage of this method.

```
<html>
<head>
<title>the title</title>
   <script type="text/javascript"
   src="/jquery/jquery-1.3.2.min.js"></script>
   <script type="text/javascript" language="javascript">
   $(document).ready(function() {
      /* Global variable */
      var count = 0;



      $("#driver").click(function(event){
          $('#stage0').load('/jquery/result.html');
      });
      /* Gets called when request starts */
      $("#stage1").ajaxStart(function(){
          count++;
          $(this).html("<h1>Starts, Count :" + count + "</h1>");
      });
      /* Gets called when request is sent */
      $("#stage2").ajaxSend(function(evt, req, set){
          count++;
          $(this).html("<h1>Sends, Count :" + count + "</h1>");
          $(this).append("<h1>URL :" + set.url  + "</h1>");
      });
```

```
      /* Gets called when request complete */
      $("#stage3").ajaxComplete(function(event,request,settings){
          count++;
          $(this).html("<h1>Completes, Count :" + count + "</h1>");
      });
      /* Gets called when all requests are ended */
      $("#stage4").ajaxStop(function(event,request,settings){
          count++;
          $(this).html("<h1>Stops, Count :" + count + "</h1>");
      });
   });
   </script>
</head>
<body>
   <p>Click on the button to load result.html file:</p>
   <div id="stage0" style="background-color:blue;">
          STAGE - 0
   </div>
   <div id="stage1" style="background-color:blue;">
          STAGE - 1
   </div>
   <div id="stage2" style="background-color:blue;">
          STAGE - 2
   </div>
   <div id="stage3" style="background-color:blue;">
          STAGE - 3
   </div>
   <div id="stage4" style="background-color:blue;">
          STAGE - 4
   </div>
   <input type="button" id="driver" value="Load Data" />
</body>
</html>
```

ajaxSuccess(callback) Method

The **ajaxSuccess(callback)** method attaches a function to be executed whenever an AJAX request completes successfully. This is an Ajax Event.

Syntax

Here is the simple syntax to use this method:

```
$(document).ajaxSuccess( callback )
```

Parameters

Here is the description of all the parameters used by this method:

- **callback**: The function to execute. The event object, XMLHttpRequest, and settings used for that request are passed as arguments to the callback.

Example

Assuming we have the following HTML content in /jquery/result.html file:

```
<h1>THIS IS RESULT...</h1>
```

Following is a simple example showing the usage of this method.

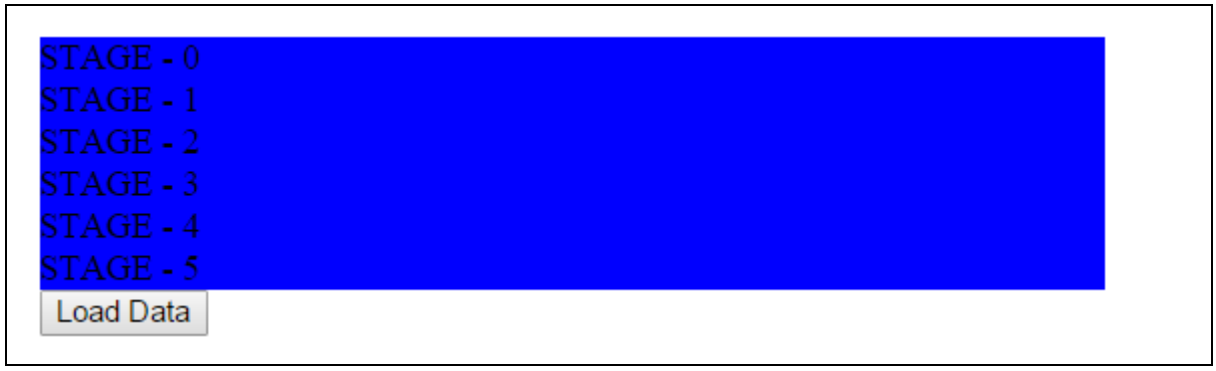
```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            /* Global variable */
            var count = 0;

            $("#driver").click(function(event){
               $('#stage0').load('result.html');
            });

            /* Gets called when request starts */
            $(document).ajaxStart(function(){
               count++;
               $("#stage1").html("<h1>Starts, Count :" + count + "</h1>");
```

```
            });

            /* Gets called when request is sent */
            $(document).ajaxSend(function(evt, req, set){
               count++;
               $("#stage2").html("<h1>Sends, Count :" + count + "</h1>");
               $("#stage2").append("<h1>URL :" + set.url  + "</h1>");
            });

            /* Gets called when request completes */
            $(document).ajaxComplete(function(event,request,settings){
               count++;
               $("#stage3").html("<h1>Completes,Count:" + count + "</h1>");
            });

            /* Gets called when request is stopped */
            $(document).ajaxStop(function(event,request,settings){
               count++;
               $("#stage4").html("<h1>Stops, Count :" + count + "</h1>");
            });

            /* Gets called when all request completes successfully */
            $(document).ajaxSuccess(function(event,request,settings){
               count++;
               $("#stage5").html("<h1>Success,Count :" + count + "</h1>");
            });
         });
      </script>

   </head>

   <body>

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

      <div id="stage0" style="background-color:blue;">
```

```
         STAGE - 0
      </div>

      <div id="stage1" style="background-color:blue;">
         STAGE - 1
      </div>

      <div id="stage2" style="background-color:blue;">
         STAGE - 2
      </div>

      <div id="stage3" style="background-color:blue;">
         STAGE - 3
      </div>

      <div id="stage4" style="background-color:blue;">
         STAGE - 4
      </div>

      <div id="stage5" style="background-color:blue;">
         STAGE - 5
      </div>

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

   </body>

</html>
```

This will produce the following result

```
Click on the button to load result.html file:
```



10. EFFECTS

jQuery provides a trivially simple interface for doing various kind of amazing effects. jQuery methods allow us to quickly apply commonly used effects with a minimum configuration. This tutorial covers all the important jQuery methods to create visual effects.

Showing and Hiding Elements

The commands for showing and hiding elements are pretty much what we would expect: **show()** to show the elements in a wrapped set and **hide()** to hide them.

Syntax

Here is the simple syntax for **show()** method:

```
[selector].show( speed, [callback] );
```

Here is the description of all the parameters:

- **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).

- **callback:** This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

Following is the simple syntax for **hide()** method:

```
[selector].hide( speed, [callback] );
```

Here is the description of all the parameters:

- **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).

- **callback:** This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

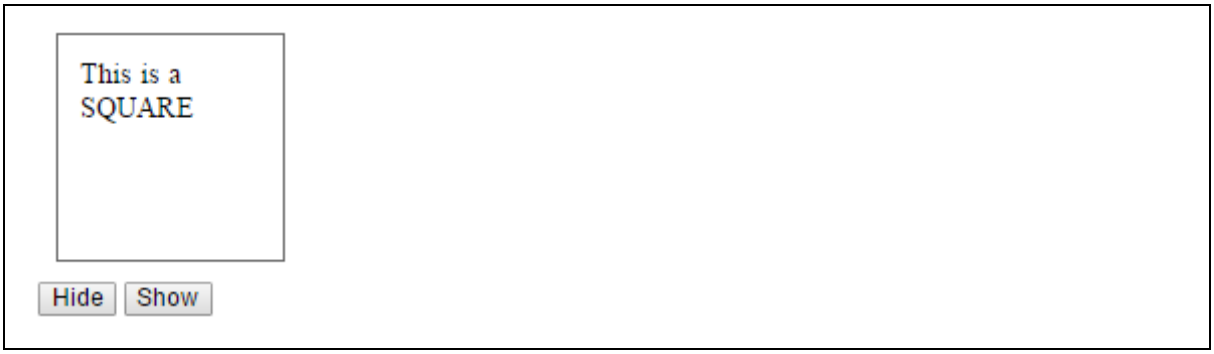
Example

Consider the following HTML file with a small JQuery coding:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
```

```
      <script type="text/javascript" language="javascript">
         $(document).ready(function() {

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

            $("#hide").click(function () {
               $(".mydiv").hide( 1000 );
            });
         });
      </script>

      <style>
         .mydiv{ margin:10px;padding:12px; border:2px solid #666; width:100px;
height:100px;}
      </style>

   </head>

   <body>

      <div class="mydiv">
         This is a SQUARE
      </div>

      <input id="hide" type="button" value="Hide" />
      <input id="show" type="button" value="Show" />

   </body>

</html>
```

This will produce the following result



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.

Syntax

Here is the simple syntax for one of the **toggle()** methods:

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

Here is the description of all the parameters:

- **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **callback:** This optional parameter represents a function to be executed whenever the animation completes; executes once for each element animated against.

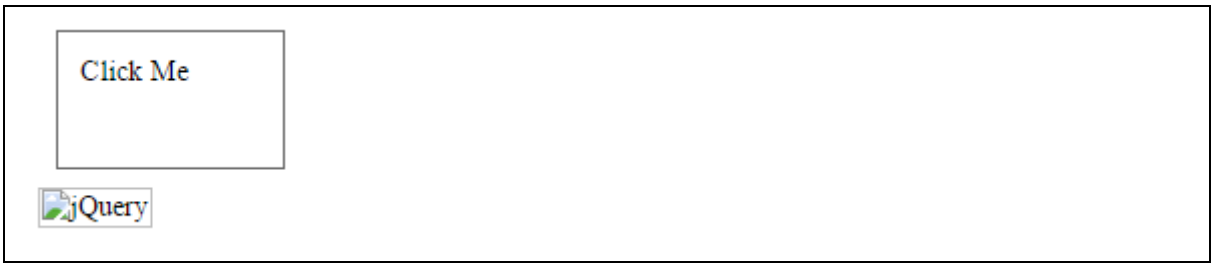
Example

We can animate any element, such as a simple containing an image:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

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

```
            });
         });
      </script>

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

   </head>

   <body>

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

</html>
```

This will produce the following result



JQuery Effect Methods

You have seen basic concept of jQuery Effects. Following table lists down all the important methods to create different kind of effects:

S.No.	Methods & Description

1	animate(params, [duration, easing, callback]) A function for making custom animations.
2	fadeIn(speed, [callback]) Fade in all matched elements by adjusting their opacity and firing an optional callback after completion.
3	fadeOut(speed, [callback]) Fade out all matched elements by adjusting their opacity to 0, then setting display to "none" and firing an optional callback after completion.
4	fadeTo(speed, opacity, callback) Fade the opacity of all matched elements to a specified opacity and firing an optional callback after completion.
5	hide( ) Hides each of the set of matched elements if they are shown.
6	hide(speed, [callback]) Hide all matched elements using a graceful animation and firing an optional callback after completion.
7	show( ) Displays each of the set of matched elements if they are hidden.
8	show(speed, [callback]) Show all matched elements using a graceful animation and firing an optional callback after completion.
9	slideDown(speed, [callback]) Reveal all matched elements by adjusting their height and firing an optional callback after completion.
10	slideToggle(speed, [callback]) Toggle the visibility of all matched elements by adjusting their height and firing an optional callback after completion.
11	slideUp(speed, [callback])

	Hide all matched elements by adjusting their height and firing an optional callback after completion.
12	stop([clearQueue, gotoEnd]) Stops all the currently running animations on all the specified elements.
13	toggle( ) Toggle displaying each of the set of matched elements.
14	toggle(speed, [callback]) Toggle displaying each of the set of matched elements using a graceful animation and firing an optional callback after completion.
15	toggle(switch) Toggle displaying each of the set of matched elements based upon the switch (true shows all elements, false hides all elements).
16	jQuery.fx.off Globally disable all animations.

animate() Method

The **animate( )** method performs a custom animation of a set of CSS properties.

Syntax

Here is the simple syntax to use this method:

```
selector.animate( params, [duration, easing, callback] );
```

Parameters

Here is the description of all the parameters used by this method:

- **params:** A map of CSS properties that the animation will move toward.
- **duration:** This is optional parameter representing how long the animation will run.
- **easing:** This is optional parameter representing which easing function to use for the transition
- **callback:** This is optional parameter representing a function to call once the animation is complete.

Example

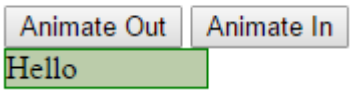
Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#out").click(function(){
               $("#block").animate({
                  width: "70%",
                  opacity: 0.4,
                  marginLeft: "0.6in",
                  fontSize: "3em",
                  borderWidth: "10px"
               }, 1500 );
            });

            $("#in").click(function(){
               $("#block").animate({
                  width: "100",
                  opacity: 1.0,
                  marginLeft: "0in",
                  fontSize: "100%",
                  borderWidth: "1px"
               }, 1500 );
            });
         });
      </script>

      <style>
         div {background-color:#bca; width:100px; border:1px solid green;}
```

```
      </style>

   </head>

   <body>

      <p>Click on any of the buttons</p>

      <button id="out"> Animate Out </button>
      <button id="in"> Animate In</button>

      <div id="block">Hello</div>

   </body>

</html>
```

This will produce the following result:

```
Click on any of the buttons
```



fadeIn() Method

The **fadeIn( )** method fades in all matched elements by adjusting their opacity and firing an optional callback after completion.

Syntax

Here is the simple syntax to use this method:

```
selector.fadeIn( speed, [callback] );
```

Parameters

Here is the description of all the parameters used by this method:

- **speed:** A string representing one of the three predefined speeds ("slow", "def", or "fast") or the number of milliseconds to run the animation (e.g. 1000).

- **callback:** This is optional parameter representing a function to call once the animation is complete.

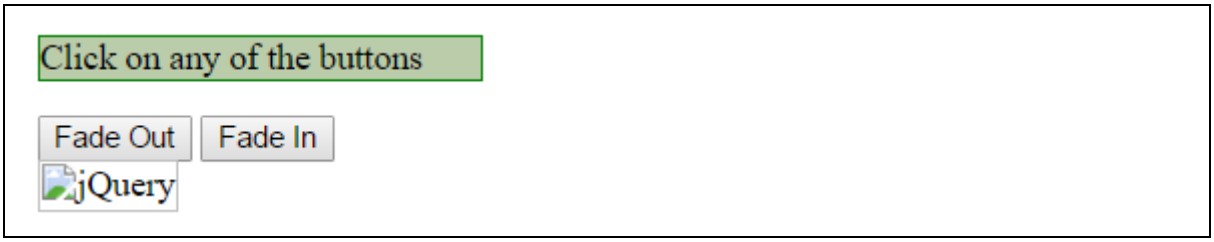
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#in").click(function(){
               $(".target").fadeIn( 'slow', function(){
                  $(".log").text('Fade In Transition Complete');
               });
            });

            $("#out").click(function(){
               $(".target").fadeOut( 'slow', function(){
                  $(".log").text('Fade Out Transition Complete');
               });
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
      </style>

   </head>

   <body>
```

```
      <p>Click on any of the buttons</p>

      <button id="out"> Fade Out </button>
      <button id="in"> Fade In</button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

      <div class="log"></div>

   </body>

</html>
```

This will produce the following result:



fadeOut() Method

The **fadeOut( )** method fades out all matched elements by adjusting their opacity to 0, then setting display to "none" and firing an optional callback after completion.

Syntax

Here is the simple syntax to use this method:

```
selector.fadeOut( speed, [callback] );
```

Parameters

Here is the description of all the parameters used by this method:

- **speed:** A string representing one of the three predefined speeds ("slow", "def", or "fast") or the number of milliseconds to run the animation (e.g. 1000).

- **callback:** This is optional parameter representing a function to call once the animation is complete.

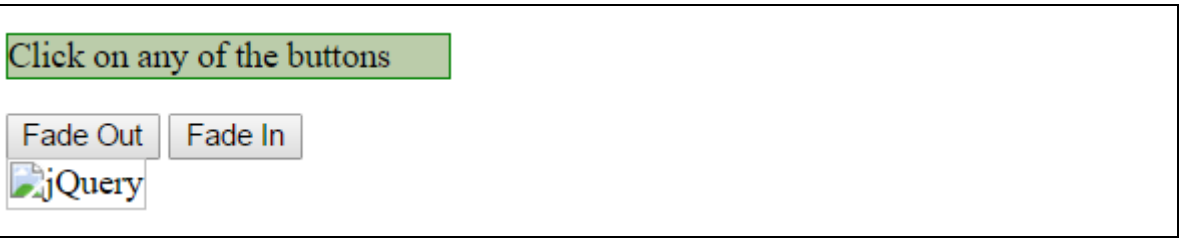
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#in").click(function(){
               $(".target").fadeIn( 'slow', function(){
                  $(".log").text('Fade In Transition Complete');
               });
            });

            $("#out").click(function(){
               $(".target").fadeOut( 'slow', function(){
                  $(".log").text('Fade Out Transition Complete');
               });
            });
         });

      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
      </style>

   </head>

   <body>
```

```
      <p>Click on any of the buttons</p>

      <button id="out"> Fade Out </button>
      <button id="in"> Fade In</button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

      <div class="log"></div>

   </body>

</html>
```

This will produce the following result:



fadeTo() Method

The **fadeTo( )** method fades the opacity of all matched elements to a specified opacity and firing an optional callback after completion.

Syntax

Here is the simple syntax to use this method:

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

Parameters

Here is the description of all the parameters used by thi method:

- **speed:** A string representing one of the three predefined speeds ("slow", "def", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **opacity:** A number between 0 and 1 denoting the target opacity.

- **callback:** This is optional parameter representing a function to call once the animation is complete.

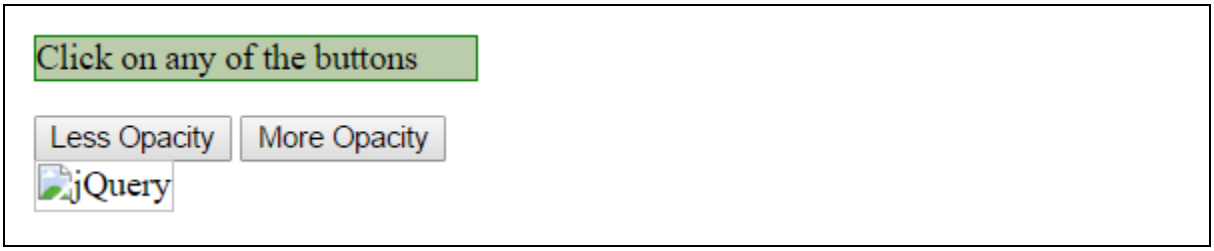
Exmple

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#more").click(function(){
               $(".target").fadeTo( 'slow', 0.7, function(){
                  $(".log").text('More Opacity Transition Complete');
               });
            });

            $("#less").click(function(){
               $(".target").fadeTo( 'slow', 0.2, function(){
                  $(".log").text('less Opacity Transition Complete');
               });
            });
         });

      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
      </style>

   </head>

   <body>
```

```
      <p>Click on any of the buttons</p>

      <button id="less"> Less Opacity </button>
      <button id="more"> More Opacity</button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

      <div class="log"></div>

   </body>

</html>
```

This will produce the following result:



hide() Method

The **hide( )** method simply hides each of the set of matched elements if they are shown. There is another form of this method which controls the speed of the animation.

Syntax

Here is the simple syntax to use this method:

```
selector.hide( );
```

Parameters

Here is the description of all the parameters used by this method:

- NA:

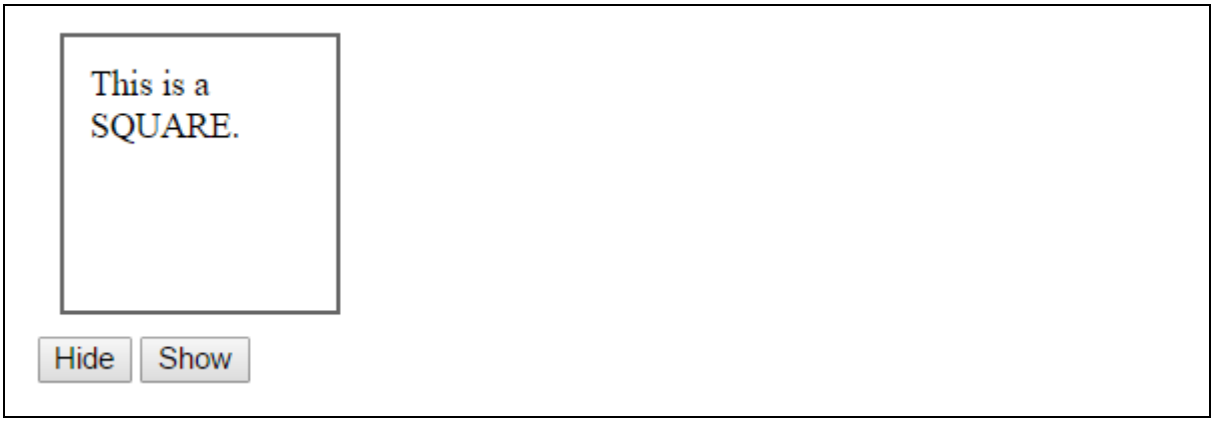
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

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

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

      </script>

      <style>
         .mydiv{ margin:10px;padding:12px; border:2px solid #666; width:100px;
height:100px;}
      </style>

   </head>

   <body>

      <div class="mydiv">
         This is a SQUARE.
      </div>

      <input id="hide" type="button" value="Hide" />
      <input id="show" type="button" value="Show" />
```

```
   </body>


</html>
```

This will produce the following result:



hide(speed, [callback]) Method

The **hide(speed, [callback])** method hides all matched elements using a graceful animation and firing an optional callback after completion.

Syntax

Here is the simple syntax to use this method:

```
selector.hide( speed, [callback] );
```

Parameters

Here is the description of all the parameters used by this method:

- **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).

- **callback:** This is optional parameter representing a function to call once the animation is complete.

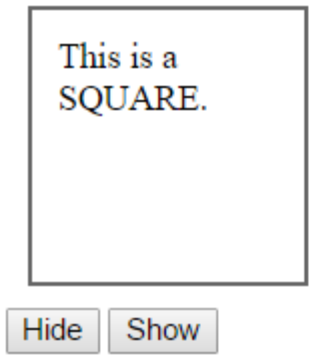
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
```

```
      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#show").click(function () {
               $(".mydiv").show( 100 );
            });

            $("#hide").click(function () {
               $(".mydiv").hide( 100 );
            });
         });

      </script>

      <style>
         .mydiv{ margin:10px;padding:12px; border:2px solid #666; width:100px;
height:100px;}
      </style>

   </head>

   <body>

      <div class="mydiv">
         This is a SQUARE.
      </div>

      <input id="hide" type="button" value="Hide" />
      <input id="show" type="button" value="Show" />

   </body>

</html>
```

This will produce the following result:



show() Method

The **show( )** method simply shows each of the set of matched elements if they are hidden. There is another form of this method which controls the speed of the animation.

Syntax

Here is the simple syntax to use this method:

```
selector.show( );
```

Parameters

Here is the description of all the parameters used by this method:

- NA:

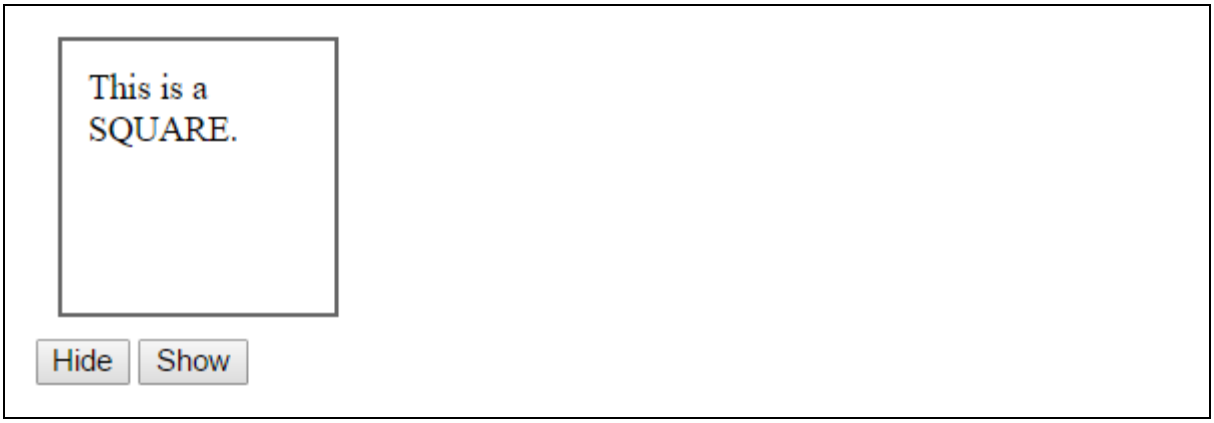
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

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

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

      </script>

      <style>
         .mydiv{ margin:10px;padding:12px; border:2px solid #666; width:100px;
height:100px;}
      </style>

   </head>

   <body>

      <div class="mydiv">
         This is a SQUARE.
      </div>

      <input id="hide" type="button" value="Hide" />
      <input id="show" type="button" value="Show" />

   </body>

</html>
```

This will produce the following result:



show(speed, [callback]) Method

The **show(speed, [callback])** method shows all matched elements using a graceful animation and firing an optional callback after completion.

Synta

Here is the simple syntax to use this method –

```
selector.show( speed, [callback] );
```

Parameter

Here is the description of all the parameters used by this method –

- **speed –** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **callback –** This is optional parameter representing a function to call once the animation is complete.

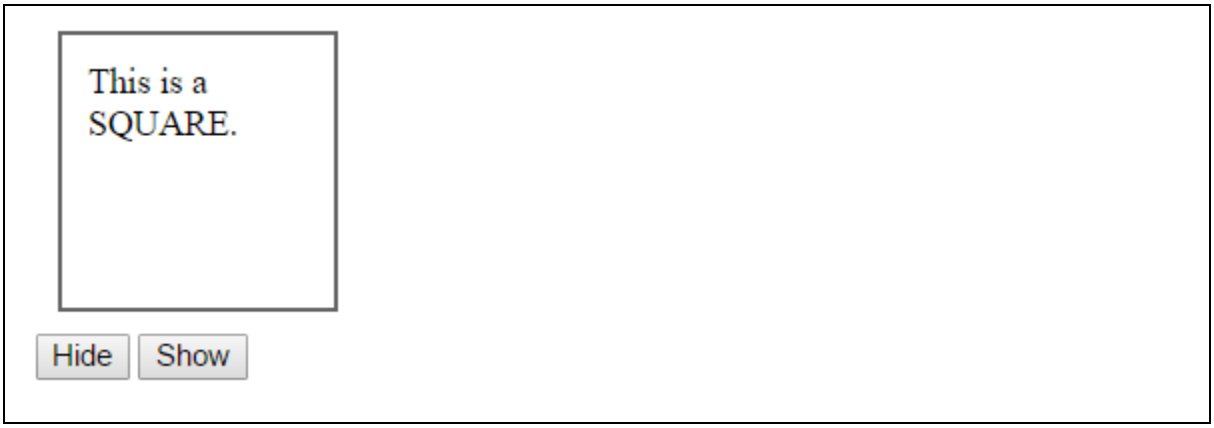
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

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

```
            $("#show").click(function () {
               $(".mydiv").show( 100 );
            });

            $("#hide").click(function () {
               $(".mydiv").hide( 100 );
            });
         });

      </script>

      <style>
         .mydiv{ margin:10px;padding:12px; border:2px solid #666; width:100px;
height:100px;}
      </style>

   </head>

   <body>

      <div class="mydiv">
         This is a SQUARE.
      </div>

      <input id="hide" type="button" value="Hide" />
      <input id="show" type="button" value="Show" />

   </body>

</html>
```

This will produce the following result:



slideDown() Method

The **slideDown()** method reveals all matched elements by adjusting their height and firing an optional callback after completion.

Syntax

Here is the simple syntax to use this method:

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

Parameters

Here is the description of all the parameters used by this method:

- **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **callback:** This is optional parameter representing a function to call once the animation is complete.

Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#show").click(function () {
```

```
            $(".mydiv").show( 100 );
         });

         $("#hide").click(function () {
            $(".mydiv").hide( 100 );
         });
      });

   </script>

   <style>
      .mydiv{ margin:10px;padding:12px; border:2px solid #666; width:100px;
height:100px;}
   </style>

</head>

<body>

   <div class="mydiv">
      This is a SQUARE.
   </div>

   <input id="hide" type="button" value="Hide" />
   <input id="show" type="button" value="Show" />

</body>

</html>
```

This will produce the following result:

slideToggle() Method

The **slideToggle()** method toggles the visibility of all matched elements by adjusting their height and firing an optional callback after completion.

Syntax

Here is the simple syntax to use this method:

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

Parameters

Here is the description of all the parameters used by this method:

- **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **callback:** This is optional parameter representing a function to call once the animation is complete.

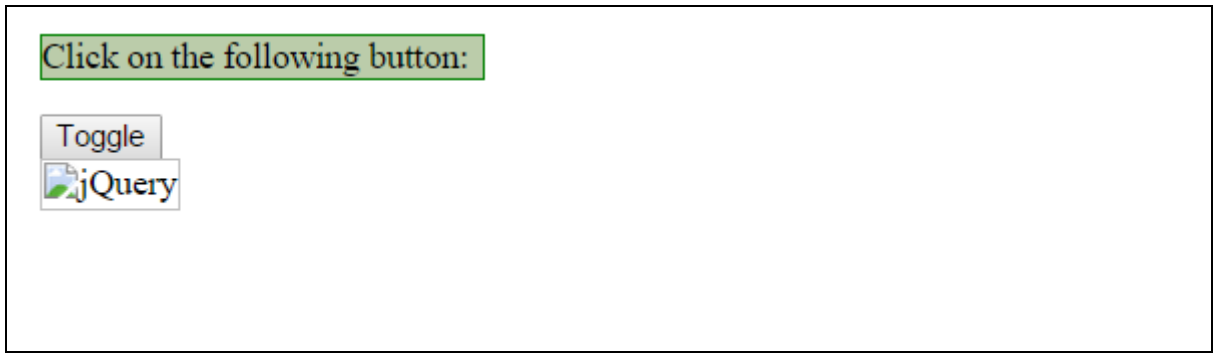
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

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

```
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
      </style>

   </head>

   <body>

      <p>Click on the following button:</p>
      <button id="toggle"> Toggle </button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

      <div class="log"></div>

   </body>

</html>
```

This will produce the following result:



slideUp() Method

The **slideUp()** method hides all matched elements by adjusting their height and firing an optional callback after completion.

Syntax

Here is the simple syntax to use this method:

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

Parameters

Here is the description of all the parameters used by this method:

- **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).
- **callback:** This is optional parameter representing a function to call once the animation is complete.

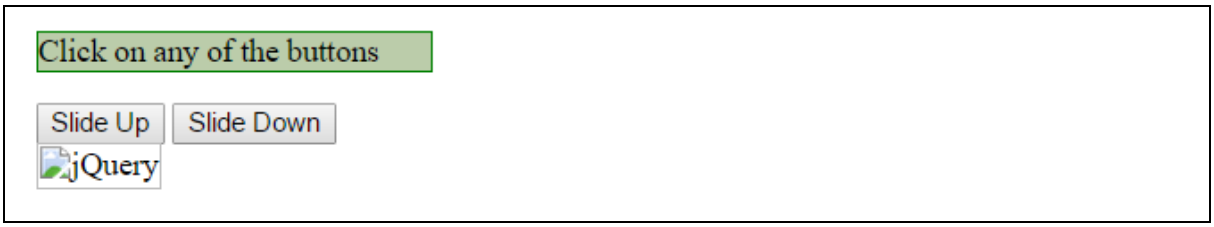
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#down").click(function(){
               $(".target").slideDown( 'slow', function(){
                  $(".log").text('Slide Down Transition Complete');
               });
            });

            $("#up").click(function(){
               $(".target").slideUp( 'slow', function(){
                  $(".log").text('Slide Up Transition Complete');
               });
            });
         });

      </script>
```

```
      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
      </style>

   </head>

   <body>

      <p>Click on any of the buttons</p>

      <button id="up"> Slide Up </button>
      <button id="down"> Slide Down</button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

      <div class="log"></div>

   </body>

</html>
```

This will produce the following result:



stop() Method

The **stop([clearQueue, gotoEnd])** method stops all the currently running animations on all the specified elements.

Syntax

Here is the simple syntax to use this method:

```
selector.stop( [clearQueue], [gotoEnd] ) ;
```

Parameters

Here is the description of all the parameters used by this method:

- **clearQueue:** This is optional boolean parameter. When set to true clears the animation queue, effectively stopping all queued animations.

- **gotoEnd :** This is optional boolean parameter. A Boolean (true/false) that when set to true causes the currently playing animation to immediately complete, including resetting original styles on show and hide and calling the callback function.

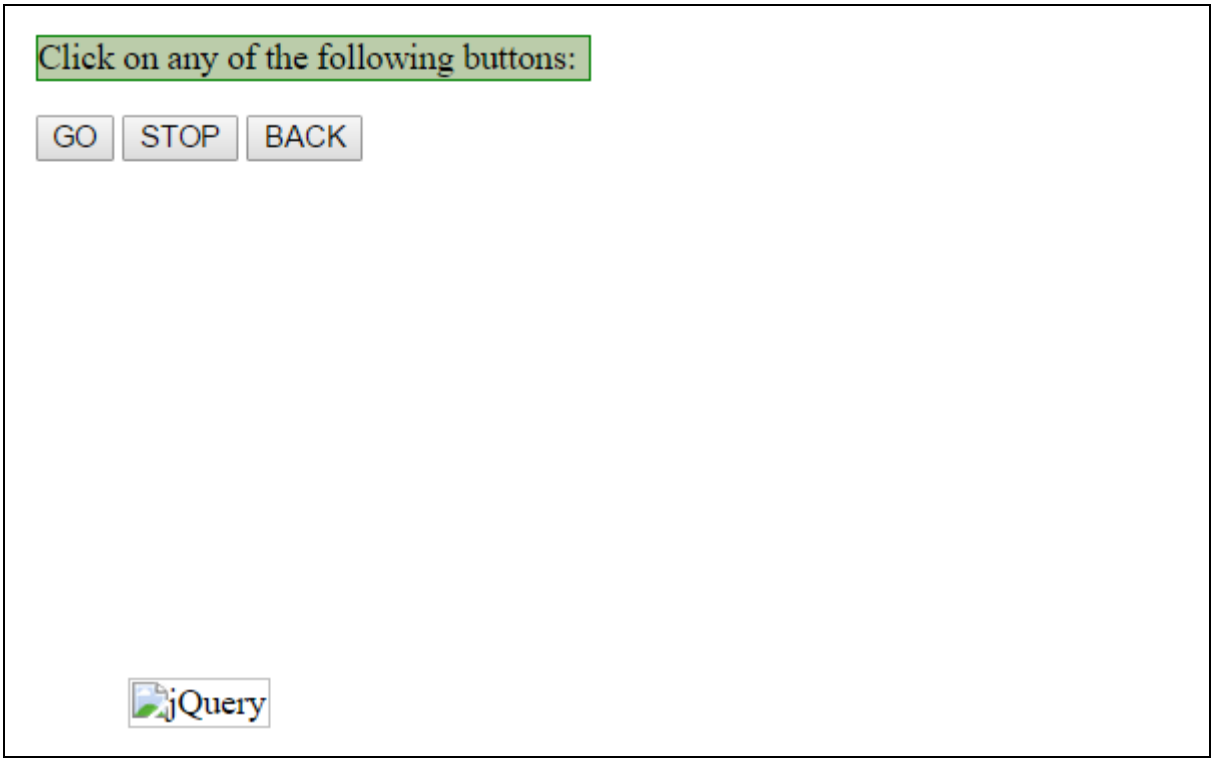
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#go").click(function(){
               $(".target").animate({left: '+=100px'}, 2000);
            });

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

            $("#back").click(function(){
               $(".target").animate({left: '-=100px'}, 2000);
            });
         });
      </script>
```

```
      <style>
         p {background-color:#bca; width:250px; border:1px solid green;}
         div{position: absolute; left: 50px; top:300px;}
      </style>

   </head>

   <body>

      <p>Click on any of the following buttons:</p>

      <button id="go"> GO</button>
      <button id="stop"> STOP </button>
      <button id="back"> BACK </button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

   </body>

</html>
```

This will produce the following result:



toggle() Method

The **toggle()** method toggles displaying each of the set of matched elements.

Syntax

Here is the simple syntax to use this method:

```
selector.toggle( );
```

Parameters

Here is the description of all the parameters used by this method:

- NA:

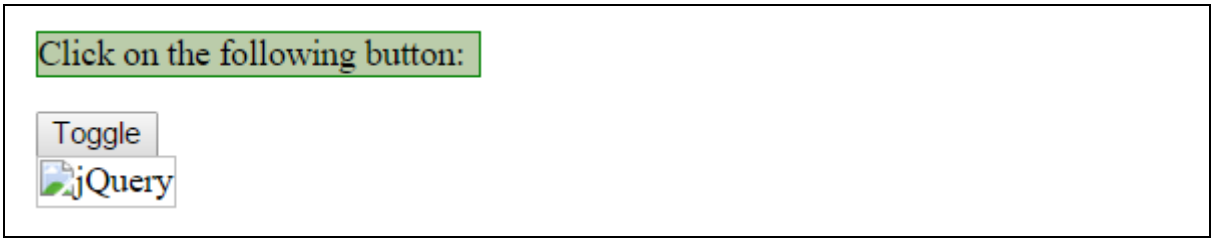
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
```

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

            $("#toggle").click(function(){
               $(".target").toggle( );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
      </style>

   </head>

   <body>

      <p>Click on the following button:</p>
      <button id="toggle"> Toggle </button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

   </body>

</html>
```

This will produce the following result:



toggle(speed, [callback]) Method

The **toggle(speed, [callback])** method toggles displaying each of the set of matched elements using a graceful animation and firing an optional callback after completion.

Syntax

Here is the simple syntax to use this method:

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

Parameters

Here is the description of all the parameters used by this method:

- **speed:** A string representing one of the three predefined speeds ("slow", "normal", or "fast") or the number of milliseconds to run the animation (e.g. 1000).

- **callback:** This is optional parameter representing a function to call once the animation is complete.

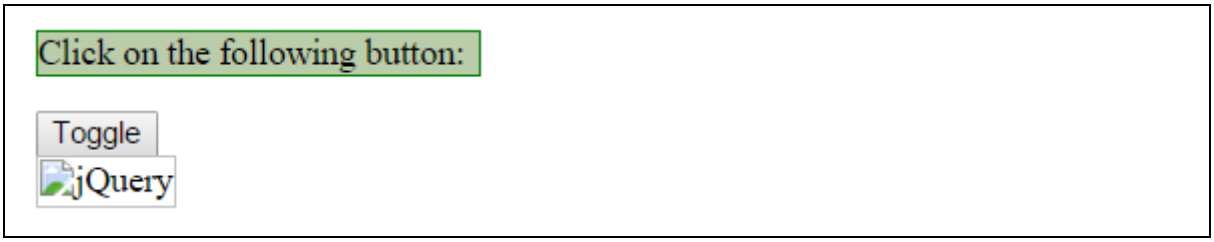
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#toggle").click(function(){
               $(".target").toggle( 'slow', function(){
                  $(".log").text('Toggle Transition Complete');
               });
            });
         });
      </script>

      <style>
```

```
         p {background-color:#bca; width:200px; border:1px solid green;}
      </style>

   </head>

   <body>

      <p>Click on the following button:</p>
      <button id="toggle"> Toggle </button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

      <div class="log"></div>

   </body>

</html>
```

This will produce the following result:



toggle(switch) Method

The **toggle(switch)** method toggle displaying each of the set of matched elements based upon the passed paratmer. If true parameter shows all elements, false hides all elements.

Syntax

Here is the simple syntax to use this method:

```
selector.toggle( switch );
```

Parameters

Here is the description of all the parameters used by this method:

- **switch:** A switch to toggle the display on.

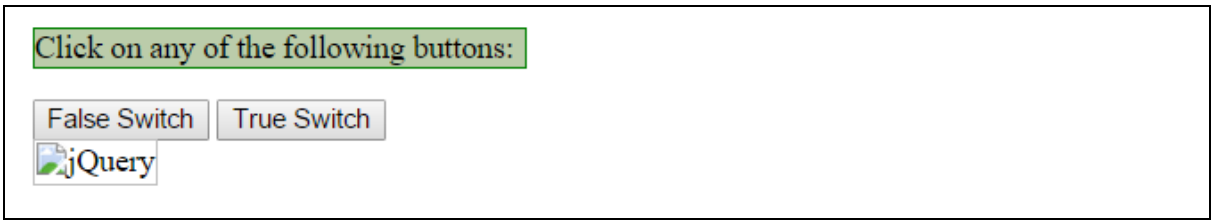
Example

Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {

            $("#false").click(function(){
               $(".target").toggle(false);
            });

            $("#true").click(function(){
               $(".target").toggle(true);
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:250px; border:1px solid green;}
      </style>

   </head>

   <body>

      <p>Click on any of the following buttons:</p>

      <button id="false"> False Switch </button>
```

```
      <button id="true"> True Switch </button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

   </body>

</html>
```

This will produce the following result:



jQuery.fx.off() Method

The **jQuery.fx.off()** method globally disables all the animations.

Syntax

Here is the simple syntax to use this method:

```
jQuery.fx.off = [true | false ] ;
```

Parameters

Here is the description of all the parameters used by this method:

- **Boolean** This should be set to either false to enable the animations or to true to disable the animations globally.

Example

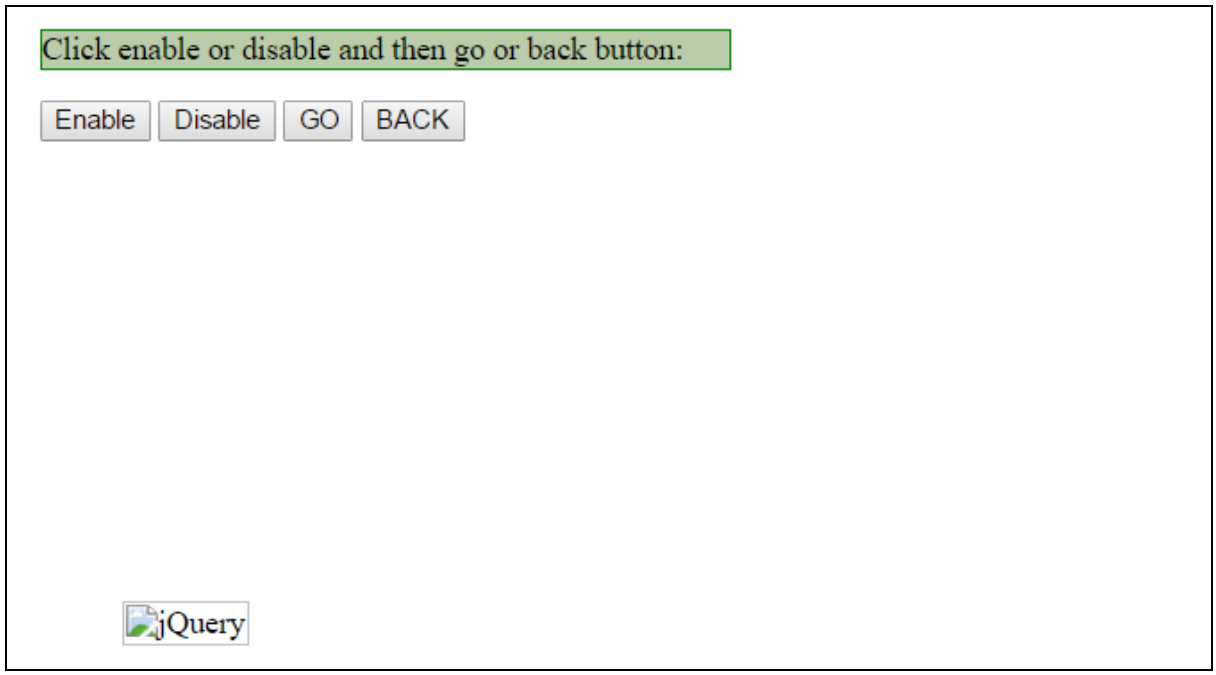
Following is a simple example showing the usage of this method:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
```

```
   <script type="text/javascript" language="javascript">
      $(document).ready(function() {

         $("#enable").click(function(){
            jQuery.fx.off = false;
         });

         $("#disable").click(function(){
            jQuery.fx.off = true;
         });

         $("#go").click(function(){
            $(".target").animate({left: '+=200px'}, 2000);
         });

         $("#back").click(function(){
            $(".target").animate({left: '-=200px'}, 200);
         });
      });
   </script>

   <style>
      p {background-color:#bca; width:350px; border:1px solid green;}
      div{position: absolute; left: 50px; top:300px;}
   </style>

</head>

<body>

   <p>Click enable or disable and then go or back button:</p>

   <button id="enable"> Enable</button>
   <button id="disable"> Disable </button>
   <button id="go"> GO</button>
   <button id="back"> BACK </button>
```

```
      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

   </body>

</html>
```

This will produce the following result:



UI Library Based Effects

To use these effects you can either download latest jQuery UI Library **jquery-ui-1.11.4.custom.zip** from [jQuery UI Library] or use Google CDN to use it in the similar way as we have done for jQuery. We have used Google CDN for jQuery UI using the following code snippet in the HTML page so we can use jQuery UI –

```
<head>
   <script src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>
</head>
```

S.No.	Methods & Description

1	**Blind** Blinds the element away or shows it by blinding it in.
2	**Bounce** Bounces the element vertically or horizontally n-times.
3	**Clip** Clips the element on or off, vertically or horizontally.
4	**Drop** Drops the element away or shows it by dropping it in.
5	**Explode** Explodes the element into multiple pieces.
6	**Fold** Folds the element like a piece of paper.
7	**Highlight** Highlights the background with a defined color.
8	**Puff** Scale and fade out animations create the puff effect.
9	**Pulsate** Pulsates the opacity of the element multiple times.
10	**Scale** Shrink or grow an element by a percentage factor.
11	**Shake** Shakes the element vertically or horizontally n-times.
12	**Size** Resize an element to a specified width and height.
13	**Slide**

	Slides the element out of the viewport.
14	**Transfer** Transfers the outline of an element to another.

Blind Effect

The **Blind** effect can be used with show/hide/toggle. This blinds the element away or shows it by blinding it in.

Syntax

Here is the simple syntax to use this effect:

```
selector.hide|show|toggle( "blind", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **direction:** The direction of the effect. Can be "vertical" or "horizontal". Default is vertical.
- **mode:** The mode of the effect. Can be "show" or "hide". Default is hide.

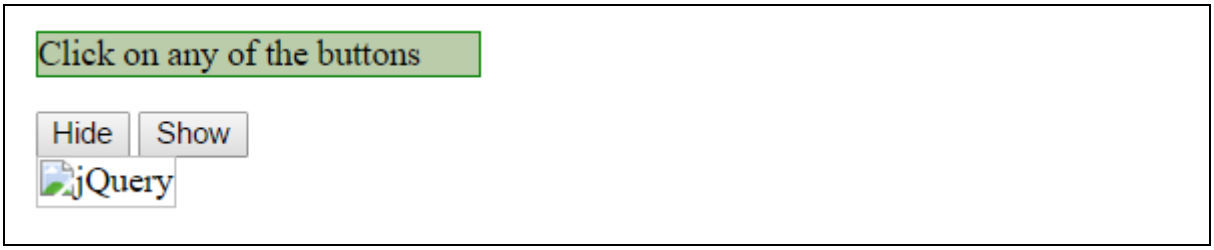
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {

            $("#hide").click(function(){
```

```
            $(".target").hide( "blind", {direction: "horizontal"}, 1000 );
         });

         $("#show").click(function(){
            $(".target").show( "blind", {direction: "horizontal"}, 1000 );
         });
      });
   </script>

   <style>
      p {background-color:#bca; width:200px; border:1px solid green;}
   </style>

</head>

<body>

   <p>Click on any of the buttons</p>

   <button id="hide"> Hide </button>
   <button id="show"> Show</button>

   <div class="target">
      <img src="./images/jquery.jpg" alt="jQuery" />
   </div>

</body>

</html>
```

This will produce the following result:



Bounce Effect

The **Bounce** effect can be used with effect() method. This bounces the element multiple times, vertically or horizontally

Syntax

Here is the simple syntax to use this effect:

```
selector.effect( "bounce", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **direction:** The direction of the effect. Can be "up", "down", "left", "right". Default is "up".
- **distance:** Distance to bounce. Default is 20
- **mode:** The mode of the effect. Can be "show", "hide" or "effect". Default is "effect".
- **times:** Times to bounce. Default is 5.

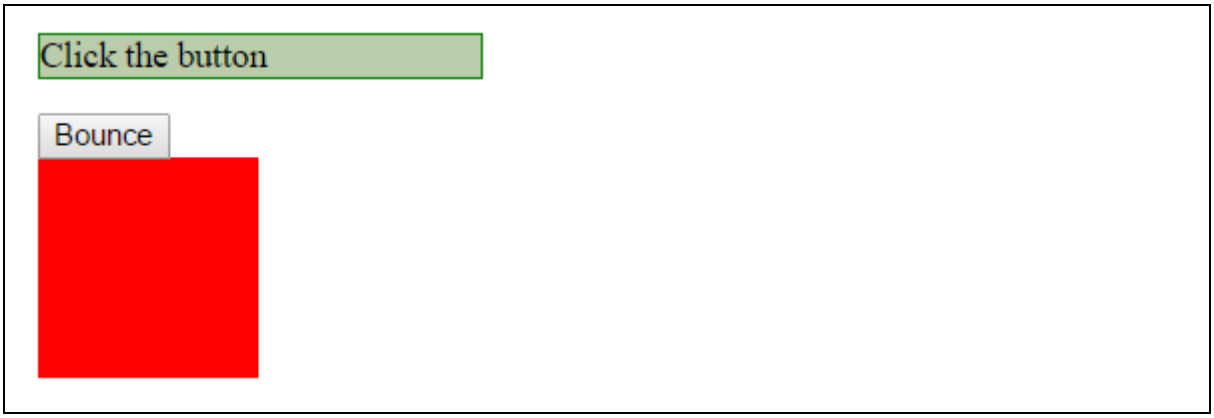
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">
```

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

            $("#button").click(function(){
               $(".target").effect( "bounce", {times:3}, 300 );
            });

         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{ width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click the button</p>
      <button id="button"> Bounce </button>

      <div class="target">
      </div>

   </body>

</html>
```

This will produce the following result:



Clip Effect

The **Clip** effect can be used with show/hide/toggle. This clips the element on or off, vertically or horizontally.

Syntax

Here is the simple syntax to use this effect:

```
selector.hide|show|toggle( "clip", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **direction:** The direction of the effect. Can be "vertical" or "horizontal". Default is vertical.
- **mode:** The mode of the effect. Can be "show" or "hide". Default is hide.

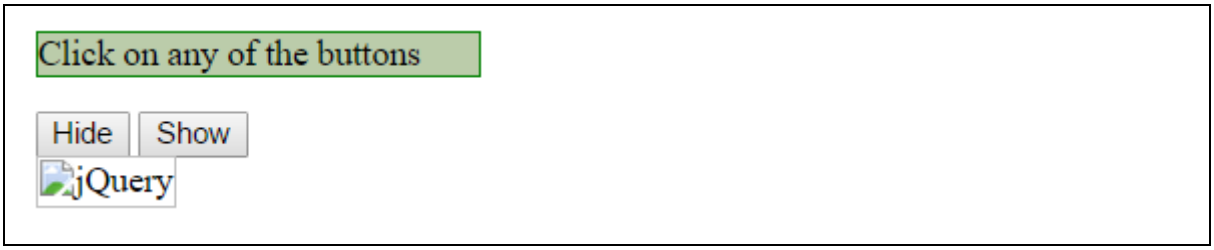
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">
```

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

            $("#hide").click(function(){
               $(".target").hide( "clip", {direction: "horizontal"}, 1000 );
            });

            $("#show").click(function(){
               $(".target").show( "clip", {direction: "vertical"}, 1000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
      </style>

   </head>

   <body>

      <p>Click on any of the buttons</p>

      <button id="hide"> Hide </button>
      <button id="show"> Show</button>

      <div class="target">
         <img src="./images/jquery.jpg" alt="jQuery" />
      </div>

   </body>

</html>
```

This will produce the following result:



Drop Effect

The **Drop** effect can be used with show/hide/toggle. This drops the element away or shows it by dropping it in.

Syntax

Here is the simple syntax to use this effect:

```
selector.hide|show|toggle( "drop", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **direction:** The direction of the effect. Can be "left", "right", "up", "down". Default is left.
- **mode:** The mode of the effect. Can be "show" or "hide". Default is hide.

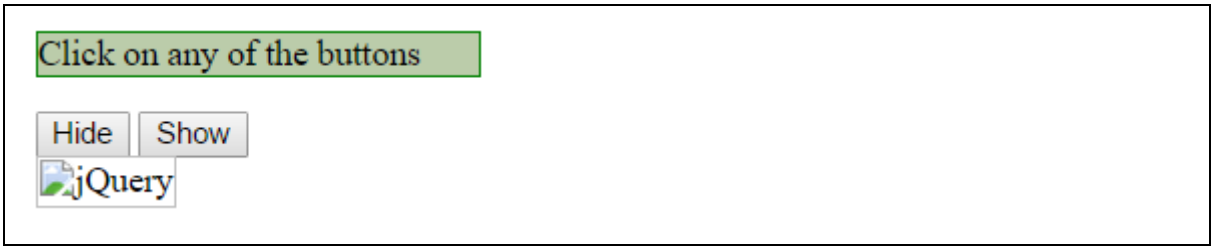
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#hide").click(function(){
```

```
            $(".target").hide( "drop", {direction: "up"}, 1000 );
         });

         $("#show").click(function(){
            $(".target").show( "drop", {direction: "down"}, 1000 );
         });
      });
   </script>

   <style>
      p {background-color:#bca; width:200px; border:1px solid green;}
   </style>

</head>

<body>

   <p>Click on any of the buttons</p>

   <button id="hide"> Hide </button>
   <button id="show"> Show</button>

   <div class="target">
      <img src="./images/jquery.jpg" alt="jQuery" />
   </div>

</body>

</html>
```

This will produce the following result:



Explode Effect

The **Explode** effect can be used with show/hide/toggle. This explodes or implodes the element into/from many pieces.

Syntax

Here is the simple syntax to use this effect:

```
selector.hide|show|toggle( "explode", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **pieces:** Number of pieces to be exploded to/imploded from.
- **mode:** The mode of the animation. Can be set to "show" or "hide".

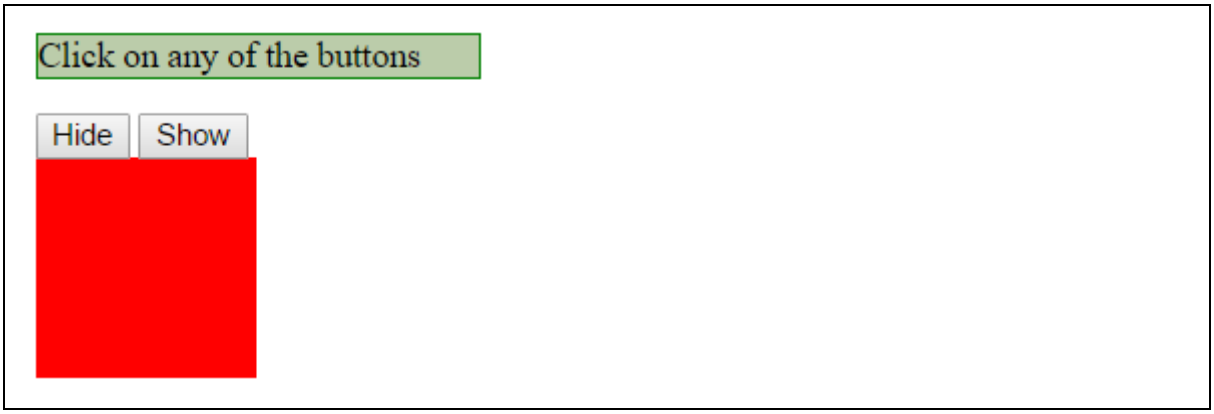
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#hide").click(function(){
               $(".target").hide( "explode", {pieces: 16 }, 2000 );
```

```
            });

            $("#show").click(function(){
               $(".target").show( "explode", {pieces: 16}, 2000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click on any of the buttons</p>

      <button id="hide"> Hide </button>
      <button id="show"> Show</button>

      <div class="target">
      </div>

   </body>

</html>
```

This will produce the following result:



Fold Effect

The **Fold** effect can be used with show/hide/toggle. This folds the element like a piece of paper.

Syntax

Here is the simple syntax to use this effect:

```
selector.hide|show|toggle( "fold", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **horizFirst:** Whether to fold horizontally first or not. Can be true or false. Defualt is false.
- **mode:** The mode of the effect. Can be "show" or "hide". Default is hide.
- **size:** Size to be folded to. Default is 15.

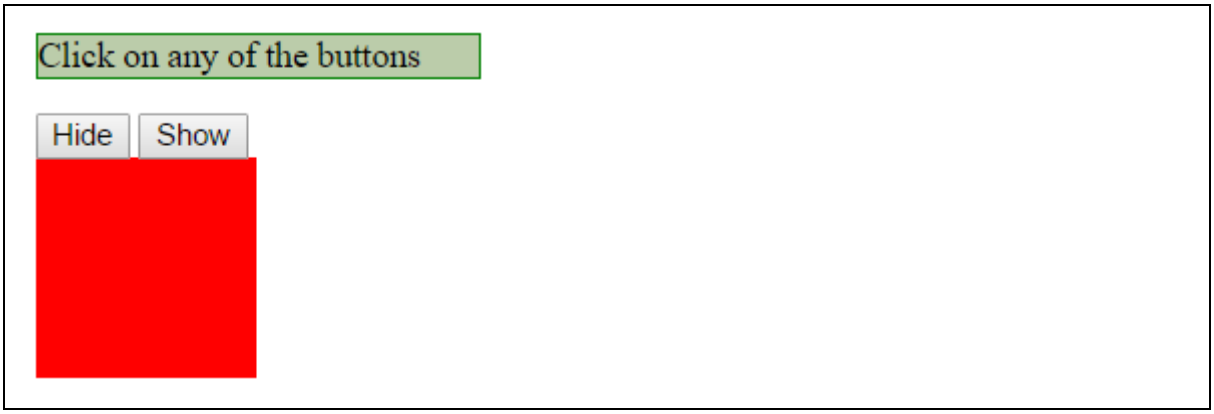
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>
```

```
      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#hide").click(function(){
               $(".target").hide( "fold", {horizFirst: true }, 2000 );
            });

            $("#show").click(function(){
               $(".target").show( "fold", {horizFirst: true}, 2000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{ width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click on any of the buttons</p>

      <button id="hide"> Hide </button>
      <button id="show"> Show</button>

      <div class="target">
      </div>

   </body>

</html>
```

This will produce the following result:



Highlight Effect

The **Highlight** effect can be used with effect() method. This highlights the element's background with a specific color, default is yellow.

Syntax

Here is the simple syntax to use this effect:

```
selector.effect( "highlight", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **color:** Highlight color. Default is "#ffff99".
- **mode:** The mode of the effect. Can be "show", "hide". Default is "show".

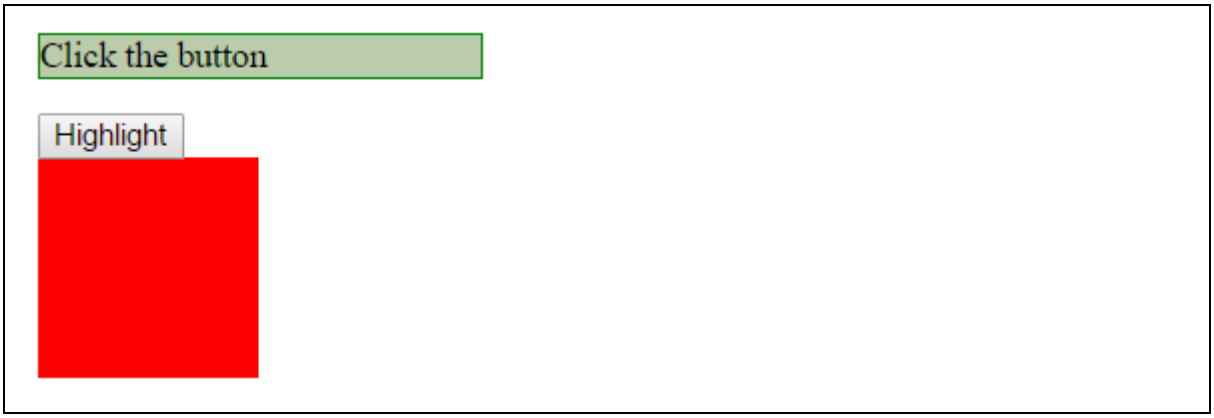
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">
```

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

            $("#button").click(function(){
               $(".target").effect( "highlight", {color:"#669966"}, 3000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{ width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click the button</p>
      <button id="button"> Highlight </button>

      <div class="target">
      </div>

   </body>

</html>
```

This will produce the following result:



Puff Effect

The **Puff** effect can be used with show/hide/toggle. This creates a puff effect by scaling the element up and hiding it at the same time.

Syntax

Here is the simple syntax to use this effect:

```
selector.hide|show|toggle( "puff", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **mode:** The mode of the effect. Can be "show" or "hide". Default is hide.
- **percent:** The percentage to scale to. Default is 150.

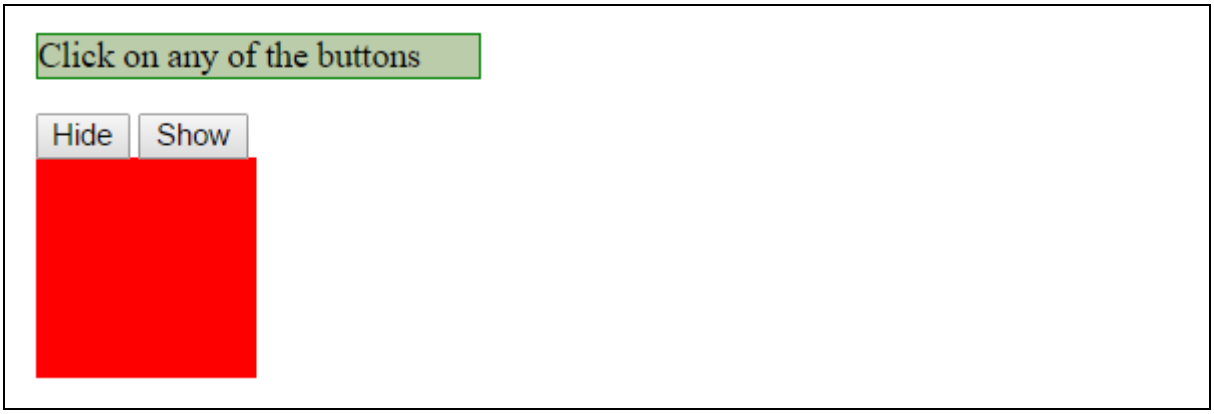
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">
```

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

            $("#hide").click(function(){
               $(".target").hide( "puff", { }, 2000 );
            });

            $("#show").click(function(){
               $(".target").show( "puff", {percent:100}, 2000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{ width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click on any of the buttons</p>

      <button id="hide"> Hide </button>
      <button id="show"> Show</button>

      <div class="target">
      </div>

   </body>

</html>
```

This will produce the following result:



Pulsate Effect

The **Pulsate** effect can be used with effect() method. This pulsates the opacity of the element multiple times.

Syntax

Here is the simple syntax to use this effect:

```
selector.effect( "pulsate", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **times:** Times to pulsate. Default is 3.
- **mode:** The mode of the effect. Can be "show", "hide". Default is "show".

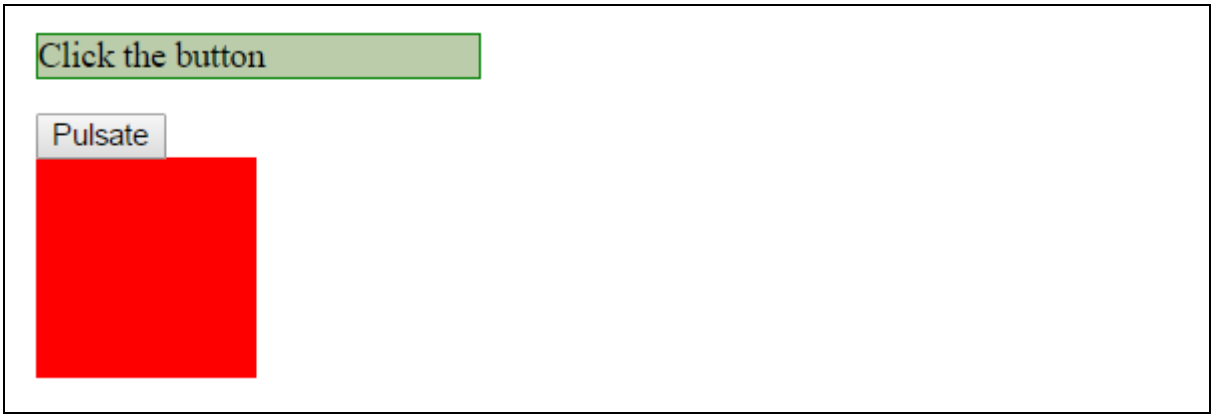
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">

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

```
            $("#button").click(function(){
               $(".target").effect( "pulsate", {times:5}, 3000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{ width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click the button</p>
      <button id="button"> Pulsate </button>

      <div class="target">
      </div>

   </body>

</html>
```

This will produce the following result:



Scale Effect

The **Scale** effect can be used with show/hide/toggle. This shrinks or grows an element by a percentage factor.

Syntax

Here is the simple syntax to use this effect:

```
selector.hide|show|toggle( "scale", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **direction:** The direction of the effect. Can be "both", "vertical" or "horizontal". Default is both.
- **from:** The state at beginning, usually not needed. This would be an object and would be given in the form of { height: .., width: .. }.
- **origin:** The vanishing point. This is an array and by default set to ['middle','center'].
- **percent:** The percentage to scale to, number. Default is 0/100.
- **scale:** Which areas of the element will be resized: 'both', 'box', 'content' Box resizes the border and padding of the element Content resizes any content inside of the element. Default is both.

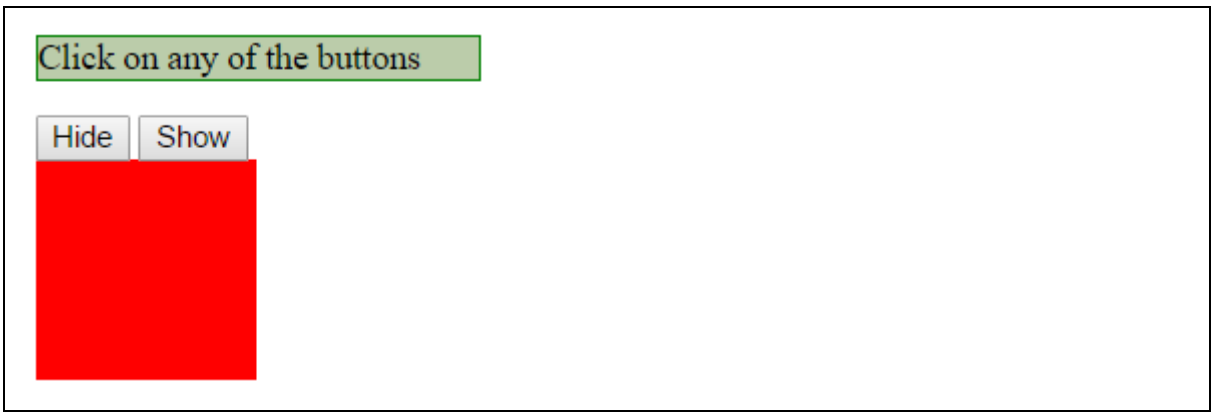
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">

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

```
            $("#hide").click(function(){
               $(".target").hide( "scale", {percent: 200, direction:
'horizontal'}, 2000 );
            });

            $("#show").click(function(){
               $(".target").show( "scale", {percent: 200, direction:
'vertical' }, 2000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{ width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click on any of the buttons</p>

      <button id="hide"> Hide </button>
      <button id="show"> Show</button>

      <div class="target">
      </div>

   </body>
</html>
```

This will produce the following result:



Shake Effect

The **Shake** effect can be used with effect() method. This shakes the element multiple times, vertically or horizontally.

Syntax

Here is the simple syntax to use this effect:

```
selector.effect( "shake", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **times:** Times to shake. Default is 3.
- **distance:** Distance to shake. Default is 20.
- **direction:** The direction of the effect. Can be "up", "down", "left", "right". Default is "left"

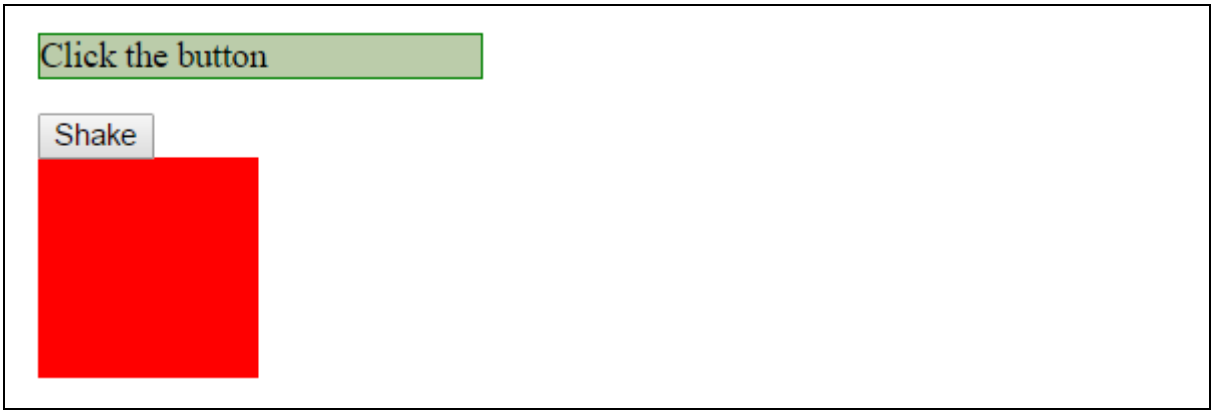
Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>
```

```
      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#button").click(function(){
               $(".target").effect( "shake", {times:4}, 1000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{ width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click the button</p>
      <button id="button"> Shake </button>

      <div class="target">
      </div>

   </body>

</html>
```

This will produce the following result:



Size Effect

The **Size** effect can be used with effect() method. This resizes an element to a specified width and height.

Syntax

Here is the simple syntax to use this effect:

```
selector.effect( "size", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **from:** State at beginning, usually not needed. This is an object in the form of { height: .., width: .. }
- **to:** Height and width to resize to. This is an object in the form of { height: .., width: .. }
- **origin:** The vanishing point, default for show/hide. This is an array and default is ['middle','center'].
- **scale:** Which areas of the element will be resized: 'both', 'box', 'content' Box resizes the border and padding of the element Content resizes any content inside of the element. Default is "both"

Example

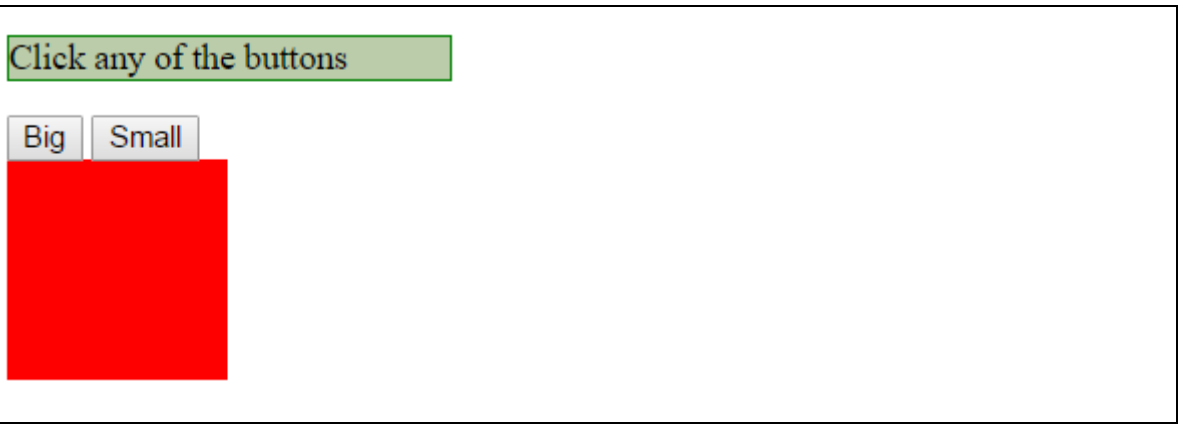
Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
```

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

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#big").click(function(){
               $(".target").effect( "size", { to: {width: 200,height: 200} }, 1000 );
            });

            $("#small").click(function(){
               $(".target").effect( "size", { to: {width: 10,height: 10} }, 1000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{ width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click any of the buttons</p>

      <button id="big"> Big </button>
      <button id="small"> Small </button>

      <div class="target">
      </div>
```

```
   </body>

</html>
```

This will produce the following result:



Slide Effect

The **Slide** effect can be used with show/hide/toggle. This slides the element out of the viewport.

Syntax

Here is the simple syntax to use this effect:

```
selector.hide|show|toggle( "slide", {arguments}, Seed );
```

Parameters

Here is the description of all the arguments:

- **direction:** The direction of the effect. Can be "left", "right", "up", "down". Default is left.
- **distance:** The distance of the effect. It is set to either the height or width of the element depending on the direction option.
- **mode:** The mode of the effect. Can be "show" or "hide". Default is show.

Example

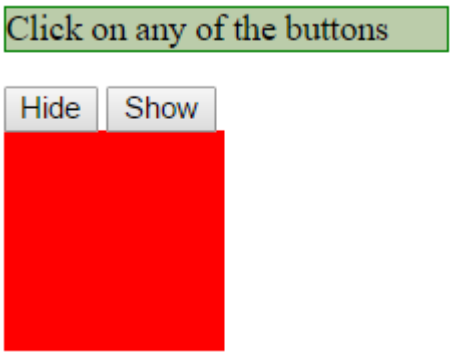
Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
```

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

      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("#hide").click(function(){
               $(".target").hide( "slide", { direction: "down"  }, 2000 );
            });

            $("#show").click(function(){
               $(".target").show( "slide", {direction: "up" }, 2000 );
            });
         });
      </script>

      <style>
         p {background-color:#bca; width:200px; border:1px solid green;}
         div{ width:100px; height:100px; background:red;}
      </style>

   </head>

   <body>

      <p>Click on any of the buttons</p>

      <button id="hide"> Hide </button>
      <button id="show"> Show</button>

      <div class="target">
      </div>
```

```
   </body>

</html>
```

This will produce the following result:



Transfer Effect

The **Transfer** effect can be used with effect() method. This *Transfers* the outline of an element to another element. It is very useful when trying to visualize interaction between two elements.

Syntax

Here is the simple syntax to use this effect:

```
selector.effect( "transfer", {arguments}, speed );
```

Parameters

Here is the description of all the arguments:

- **className:** Optional class name the transfer element will receive.
- **to:** jQuery selector, the element to transfer to.

Example

Following is a simple example showing the usage of this effect:

```
<html>
   <head>
      <title>The jQuery Example</title>
      <script type="text/javascript"
src="http://ajax.googleapis.com/ajax/libs/jquery/2.1.3/jquery.min.js"></script>
```

```
      <script type="text/javascript"
src="https://ajax.googleapis.com/ajax/libs/jqueryui/1.11.3/jquery-
ui.min.js"></script>

      <script type="text/javascript" language="javascript">

         $(document).ready(function() {

            $("div").click(function () {
               var i = 1 - $("div").index(this);
               $(this).effect("transfer",{ to: $("div").eq(i) }, 500);
            });
         });
      </script>

      <style>
         div.green { margin: 0px; width: 100px; height: 80px; background:
green; border: 1px solid black; position: relative; }
         div.red { margin-top: 10px; width: 50px; height: 30px; background:
red; border: 1px solid black; position: relative; }
         /* Following is required  to show border while transferring.*/
         .ui-effects-transfer { border: 2px solid black; }
      </style>

   </head>

   <body>

      <p>Click any of the squares:</p>

      <div class="green"></div>
      <div class="red"></div>

   </body>

</html>
```

This will produce the following result: