Contents

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, traverse 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.
- **Compatibility with languages:** The jQuery is used with all the web languages(PHP, JSP, ASP.NET, Servlet, CGI).

How to install jQuery?

It does not require much installation. You need to download jQuery library and put in web site. (http://jquery.com/)

Adding the jQuery Library to Your Pages

The jQuery library is stored as a single JavaScript file, containing all the jQuery methods.

It can be added to a web page with the following mark-up:

```
<head>
    <script type="text/javascript" src="jquery.js"></script>
</head>
```

jQuery Syntax

The jQuery syntax is tailor made for selecting HTML elements and perform some action on the element(s).

Basic syntax is: $(selector).action()

A dollar sign to define jQuery

A (selector) to "query (or find)" HTML elements

A jQuery action() to be performed on the element(s)

Examples:

$(this).hide() - hides current element.

$("p").hide() - hides all paragraphs.

$("p.test").hide() - hides all paragraphs with class="test".

$(“.test”).hide()- hides an element with class with test

$("#test").hide() - hides the element with id="test".

The Purpose of Document Ready Function

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

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

  // jQuery functions go here...

});
```

This is to prevent any jQuery code from running before the document is finished loading (is ready).

Here are some examples of actions that can fail if functions are run before the document is fully loaded:

Trying to hide an element that doesn't exist

Trying to get the size of an image that is not loaded

jQuery Selectors

jQuery selectors are one of the most important parts of the jQuery library.

jQuery selectors allow you to select and manipulate HTML elements as a group or as a single element.

jQuery selectors are required at every step while using jQuery. Selectors allow you to get the exact element/attribute you want from your HTML document.

jQuery supports the existing CSS Selectors, and in addition, it has some own custom selectors.

All type of selectors in jQuery, start with the dollar sign and parentheses: $().

The #id selector

```
<div id="divTest"></div>
<script type="text/javascript">
$(function()
{
	$("#divTest").text("Test");
});
</script>
```

The .class selector

```
<ul>
	<li class="bold">Test 1</li>
	<li>Test 2</li>
	<li class="bold">Test 3</li>
</ul>
<script type="text/javascript">
$(function()
{
	$(".bold").css("font-weight", "bold");
});
</script>
```

The element selector

$("a")

$("div")

$("span.bold").css("font-weight", "bold");

Find elements with a specific attribute

```
<span title="Title 1">Test 1</span><br />
<span>Test 2</span><br />
<span title="Title 3">Test 3</span><br />

<script type="text/javascript">
$(function()
{
	$("[title]").css("text-decoration", "underline");
});
</script>
```

Find elements with a specific value for a specific attribute

```
<a href="http://www.google.com" target="_blank">Link 1</a><br />
<a href="http://www.google.com" target="_self">Link 2</a><br />
<a href="http://www.google.com" target="_blank">Link 3</a><br />

<script type="text/javascript">
$(function()
{
	$("a[target='_blank']").append(" [new window]");
});
</script>
```

Examples of jQuery Selectors

$("*") selects all elements.

$("p") selects all <p> elements.

$("p.intro") selects all <p> elements with class="intro".

$("p#intro") selects the first <p> elements with id="intro".

$(":animated") selects all elements that are currently animated.

$(":button") selects all <button> elements and <input> elements of type="button".

$(":even") selects even elements.

$(":odd") selects odd elements.

Parent/child relation selectors

jQuery also allows you to select elements based on their parent element.

There are two variations: One which will only match elements which are a direct child to the parent element, and one which will match all the way down through the hierarchy, e.g. a child of a child of a child of a parent element.

The syntax for finding children which are direct descendants of an element looks like this:

$("div > a")

This selector will find all links which are the direct child of a div element.

Replacing the greater-than symbol with a simple space will change this to match all links within a div element, no matter if they are directly related or not:

$("div a")

Here's an example where we color bold tags blue if they are directly descending from the first test area:

```
<div id="divTestArea1">
      <b>Bold text</b>
      <i>Italic text</i>
      <div id="divTestArea2">
            <b>Bold text 2</b>
            <i>Italic text 2</i>
            <div>
                  <b>Bold text 3</b>
            </div>
      </div>
</div>

<script type="text/javascript">
$("#divTestArea1 > b").css("color", "blue");

$(“#divTestArea I”).css(“color”,”red”);
</script>
```

Some More Examples

$(this) : Selects the current HTML element

$("p#intro:first") : Selects the first <p> element with id="intro"

$(".intro") : Selects all elements with class="intro"

$("#intro") : Selects the first element with id="intro"

$("ul li:first") : Selects the first <li> element of the first

$("ul li:first-child") : Selects the first <li> element of every

$("[href]") : Selects all elements with an href attribute

$("[href$='.jpg']") : Selects all elements with an href attribute that ends with ".jpg"

$("[href='#']") **:** Selects all elements with an href value equal to "#"

$("[href!='#']") : Selects all elements with an href value NOT equal to "#"

$("div#intro .head") : Selects all elements with class="head" inside a element with id="intro"

Functions in a Separate File

If your website contains a lot of pages, and you want your jQuery functions to be easy to maintain, put your jQuery functions in a separate .js file.

When we demonstrate jQuery here, the functions are added directly into the <head> section, However, sometimes it is preferable to place them in a separate file, like this (refer to the file with the src attribute):

```
<script type="text/javascript" src="jquery.js">

</script>
<script type="text/javascript" src="my_jquery_functions.js"></script>
```

jQuery Name Conflicts

jQuery uses the $ sign as a shortcut for jQuery.

Some other JavaScript libraries also use the dollar sign for their functions.

Whenever we embed multiple javascript libraries into your HTML document,we many get name Conflict.

To overcome this name Conflict, jQuery provides an alternative to derfine some other shortcut.

The jQuery noConflict() method specifies a custom name (like jq), instead of using the dollar sign.

Examples:

Example: Maps the original object that was referenced by $ back to $.

```
jQuery.noConflict();
// Do something with jQuery
jQuery("div p").hide();

// Do something with another library's $()
$("content").style.display = 'none';
```

Example:

```
<html>
 <head>
  <script src="prototype.js"></script>
  <script src="jquery.js"></script>
  <script>
   jQuery.noConflict();

   // Use jQuery via jQuery(...)
   jQuery(document).ready(function(){
    jQuery("div").hide();
   });

   // Use Prototype with $(...), etc.
   $('someid').hide();
  </script>
 </head>
 <body>
</body>
</html>
```

```
<html>
 <head>
  <script src="prototype.js"></script>
  <script src="jquery.js"></script>
  <script>
   var jq = jQuery.noConflict();

   // Use jQuery via jq(...)
   jq(document).ready(function(){
    jq("div").hide();
   });

   // Use Prototype with $(...), etc.
   $('someid').hide();
  </script>
 </head>
 <body>
                        <div>my div</div>
</body>
 </html>
```

jQuery Effects

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

jQuery Hide and Show

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

Example

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

Both hide() and show() can take the two optional parameters: speed and callback.

Syntax:

$(selector).hide(speed,callback)

$(selector).show(speed,callback)

The speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", "normal", or milliseconds:

Example

```
$("button").click(function(){
  $("p").hide(1000);
});
```

The callback parameter is the name of a function to be executed after the hide

(or show) function completes

jQuery Toggle

The jQuery toggle() method toggles the visibility of HTML elements using the

show() or hide() methods.

Shown elements are hidden and hidden elements are shown.

Syntax:

$(selector).toggle(speed,callback)

The speed parameter can take the following values: "slow", "fast", "normal", or milliseconds.

Example

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

jQuery Slide - slideDown, slideUp, slideToggle

The jQuery slide methods gradually change the height for selected elements.

jQuery has the following slide methods:

```
$(selector).slideDown(speed,callback)

$(selector).slideUp(speed,callback)

$(selector).slideToggle(speed,callback)
```

The speed parameter can take the following values: "slow", "fast", "normal", or milliseconds.

The callback parameter is the name of a function to be executed after the function completes.

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


slideUp() Example
$(".flip").click(function(){
  $(".panel").slideUp()
})

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

jQuery Fade - fadeIn, fadeOut, fadeTo

The jQuery fade methods gradually change the opacity for selected elements.

jQuery has the following fade methods:

```
$(selector).fadeIn(speed,callback)

$(selector).fadeOut(speed,callback)

$(selector).fadeTo(speed,opacity,callback)
```

The speed parameter can take the following values: "slow", "fast", "normal", or milliseconds.

The opacity parameter in the fadeTo() method allows fading to a given opacity.

The callback parameter is the name of a function to be executed after the function completes.

```
$(document).ready(function(){
  $("div").click(function(){
    $(this).fadeOut(4000);
  });
});

$("button").click(function(){
  $("div").fadeTo("slow",0.25);
});
```

jQuery Custom Animations

The syntax of jQuery's method for making custom animations is:

$(selector).animate({params},[duration],[easing],[callback])

The key parameter is params. It defines the CSS properties that will be animated. Many properties can be animated at the same time:

```
animate({width:"70%",opacity:0.4,marginLeft:"0.6in",fontSize:"3em"});
```

Example 1

```
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({height:300},"slow");
    $("div").animate({width:300},"slow");
    $("div").animate({height:100},"slow");
    $("div").animate({width:100},"slow");
  });
});
```

Example 2

```
<script type="text/javascript">
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({left:"100px"},"slow");
    $("div").animate({fontSize:"3em"},"slow");
  });
});
</script>
```

HTML elements are positioned static by default and cannot be moved.

To make elements movable, set the CSS position property to fixed, relative or absolute.

```
<div style="height: 40px;">
    <div id="divTestBox4" style="height: 20px; width: 20px; background-color:
#89BC38; position: absolute;"></div>
</div>
<script type="text/javascript">
$(function()
{
    $("#divTestBox4").animate({ "left" : "100px" }, 1000);
    $("#divTestBox4").animate({ "left" : "20px" }, 500);
    $("#divTestBox4").animate({ "left" : "50px" }, 500);
});
</script>
```

Stopping animations with the stop() method

```
<a href="javascript:void(0);" onclick="$('#divTestArea1').slideDown(5000);">Show
box</a>
<a href="javascript:void(0);" onclick="$('#divTestArea1').stop();">Stop</a>

<div id="divTestArea1" style="padding: 100px; background-color: #89BC38; text-align:
center; display: none;">
    <b>Hello, world!</b>
</div>
```

```
<a href="javascript:void(0);"
onclick="$('#divTestArea2').slideDown(5000).slideUp(5000);">Show box</a>
<a href="javascript:void(0);" onclick="$('#divTestArea2').stop();">Stop</a>
<a href="javascript:void(0);" onclick="$('#divTestArea2').stop(true);">Stop all</a>
<a href="javascript:void(0);"
onclick="$('#divTestArea2').clearQueue().hide();">Reset</a>

<div id="divTestArea2" style="padding: 100px; background-color: #89BC38; text-align:
center; display: none;">
    <b>Hello, world!</b>
</div>
```

jQuery Effects

Here are some examples of effect functions in jQuery:

$(selector).hide() Hide selected elements

$(selector).show() Show selected elements

$(selector).toggle() Toggle (between hide and show) selected elements

$(selector).slideDown() Slide-down (show) selected elements

$(selector).slideUp() Slide-up (hide) selected elements

$(selector).slideToggle() Toggle slide-up and slide-down of selected elements

$(selector).fadeIn() Fade in selected elements

$(selector).fadeOut() Fade out selected elements

$(selector).fadeTo() Fade out selected elements to a given opacity

$(selector).animate() Run a custom animation on selected elements

A callback function is executed after the current animation is 100% finished.

jQuery Callback Functions

JavaScript statements are executed line by line. However, with animations, the next line of code can be run even though the animation is not finished. This can create errors.

To prevent this, you can create a callback function.

A callback function is executed after the current animation (effect) is finished.

jQuery Callback Example

Typical syntax: $(selector).hide(speed,callback)

The callback parameter is a function to be executed after the hide effect is completed:

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

Introduction to events

Events in JavaScript are usually something where you write a snippet of code or a name of a function within one of the event attributes on an HTML tag. For instance, you can create an event for a link by writing code like this:

```
<a href="javascript:void(0);" onclick="alert('Hello, world!');">Test</a>
```

jQuery has a different method to handle events.

The jQuery event handling methods are core functions in jQuery.

Event handlers are methods that are called when "something happens" in HTML. The term "triggered (or "fired") by an event" is often used.

It is common to put jQuery code into event handler methods in the <head> section:

$("button").click(function() {..some code... });

The method hides all <p> elements:
$("p").hide();

$(document).ready(function) :
Binds a function to the ready event of a document

$(selector).click(function) :
Triggers, or binds a function to the click event of selected elements

$(selector).dblclick(function) :
Triggers, or binds a function to the double click event of selected elements

$(selector).focus(function) :
Triggers, or binds a function to the focus event of selected elements

$(selector).mouseover(function) :
Triggers, or binds a function to the mouseover event of selected elements

And of course this is still perfectly valid when using jQuery.

However, using jQuery, you can bind code to the event of an element even easier, especially in cases where you want to attach anonymous functions or use the same code for multiple events, or even the same code for multiple events of multiple elements.

As an example, you could bind the same event to all links and span tags in your document, with only a few lines of code like this:

```
<script type="text/javascript">
$(function()
{
	$("a, span").bind("click", function() {
		alert('Hello, world!');
	});
});
</script>
```

The bind() method

One of the most important aspects of dealing with events through jQuery is the bind() and unbind() methods.

As the names imply, they will simply attach and unattach code to one or several events on a set of elements.

```
<a href="javascript:void(0);">Test 1</a>
<a href="javascript:void(0);">Test 2</a>
<script type="text/javascript">
$(function()
{
	$("a").bind("click", function() {
		alert($(this).text());
	});
});
</script>
```

```
<div id="divArea" style="background-color: silver; width: 100px; height:
100px;">
</div>
<script type="text/javascript">
$("#divArea").mousemove( function(event)
{
	$(this).text(event.pageX + "," + event.pageY);
});
</script>
```

The unbind() method

we used the bind() method to subscribe to events with jQuery.

However, you may need to remove these subscriptions again for various reasons, to prevent the event handler to be executed once the event occurs.

We do this with the unbind() method, which in its simplest form simply looks like this:

```
$("a").unbind();
```

This will remove any event handlers that you have attached with the bind() function.

However, you may want to only remove event subscriptions of a specific type, for instance clicks and doubleclicks:

```
$("a").unbind("click doubleclick");
```

```
<a href="javascript:void(0);">Test 1</a>
<a href="javascript:void(0);">Test 2</a>
<script type="text/javascript">
var msg = "Hello, world!";
$(function()
{
	$("a").bind("click", function() {
		alert("First event handler!");
	});

	$("a").bind("click", function() {
		alert("Second event handler!");
		$("a").unbind("click");
	});
});
</script>
```

jQuery HTML Manipulation

jQuery contains powerful methods (functions) for changing and manipulating HTML elements and attributes.

Changing HTML Content

$(selector).html(content)

The html() method changes the contents (innerHTML) of matching HTML elements.

$("p").html("Hello..jQuery");

jQuery contains powerful methods (functions) for changing and manipulating HTML elements and attributes.

Adding HTML content

$(selector).append(content)

The append() method appends content to the inside of matching HTML elements.

$(selector).prepend(content)

The prepend() method "prepends" content to the inside of matching HTML elements.

Example

$("p").append(" JQuery");

$(selector).after(content)

The after() method inserts HTML content after all matching elements.

$(selector).before(content)

The before() method inserts HTML content before all matching elements.

Example

$("p").after(" JQuery.");

Getting and setting attributes [attr()]

The name of the attribute we wish to get:

```
<a href="http://www.google.com" id="aGoogle1">Google Link</a>
<script type="text/javascript">
$(function()
{
      alert($("#aGoogle1").attr("href"));
});
</script>
```

To change an attribute, we simply specify an extra parameter:

```
a href="http://www.google.com" id="aGoogle2">Google Link</a>
<script type="text/javascript">
$(function()
{
      $("#aGoogle2").attr("href", "http://www.google.co.uk");
});
</script>
```

Here we set both the href and the title attributes simultaneously:

```
<a href="http://www.google.com" id="aGoogle3">Google Link</a>
<script type="text/javascript">
$(function()
{
      $("#aGoogle3").attr(
      {
            "href" : "http://www.google.co.uk",
            "title" : "Google.co.uk"
      });
});
</script>
```

jQuery CSS Manipulation

jQuery css() Method

jQuery has one important method for CSS manipulation: css()

The css() method has three different syntaxes, to perform different tasks.

css(name) - Return CSS property value

css(name,value) - Set CSS property and value

css({properties}) - Set multiple CSS properties and values

Return CSS Property

Use css(name) to return the specified CSS property value of the FIRST matched element:

Example

```
$(this).css("background-color");
```

```
$(document).ready(function(){
  $("div").click(function(){
    $("p").html($(this).css("background-color"));
  });
});
```

Set CSS Property and Value

Use css(name,value) to set the specified CSS property for ALL matched elements:

Example

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

Set Multiple CSS Property/Value Pairs

Use css({properties}) to set one or more CSS property/value pairs for the selected elements:

Example

```
$("p").css({"background-color":"yellow","font-size":"200%"});
```

```
$(document).ready(function(){
  $("button").click(function(){
    $("p").css("background-color","yellow");
  });
});
```

Getting and setting CSS classes

```
<style>

.bold {
	font-weight: bold;
}

.blue {
	color: blue;
}

</style>

<a href="javascript:void(0);" onclick="ToggleClass(this);">Toggle class</a>

<script type="text/javascript">
function ToggleClass(sender)
{

	if($(sender).hasClass("bold"))
		$(sender).removeClass("bold");
	else
		$(sender).addClass("bold");
}
</script>
```

We can also select multiple elements, where we can add or remove multiple classes, as well.

Here's an example of just that:

```
<div id="divTestArea1">
      <span>Test 1</span><br />
      <div>Test 2</div>
      <b>Test 3</b><br />
</div>
<script type="text/javascript">
$(function()
{
      $("#divTestArea1 span, #divTestArea1 b").addClass("blue");
      $("#divTestArea1 div").addClass("bold blue");
});
</script>
```

Method chaining

```
<div id="divTest1"></div>
<script type="text/javascript">
      $("#divTest1").text("Hello, world!").css("color", "blue");
</script>


<div id="divTest2"></div>
<script type="text/javascript">
      $("#divTest2").text("Hello, world!")
                              .removeClass("blue")
                              .addClass("bold")
                              .css("color", "blue");
</script>
```

More about jQuery:
For click and most other events, you can prevent the default behavior by calling event.preventDefault() in the event handler:

```
<!doctype html>
<html>
<head>
  <meta charset="utf-8">
  <title>Demo</title>
</head>
<body>
  <a href="http://jquery.com/">jQuery</a>
  <script src="jquery.js"></script>
  <script>

  $( document ).ready(function() {
    $( "a" ).click(function( event ) {
      alert( "The link will no longer take you to jquery.com" );
      event.preventDefault();
    });
  });

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

Utility Methods

jQuery offers several utility methods in the $ namespace.

These methods are helpful for accomplishing routine programming tasks

Below are examples of a few of the utility methods

$.trim()

Removes leading and trailing whitespace:

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

$.each()

Iterates over arrays and objects:

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

$.each({ foo: "bar", baz: "bim" }, function( k, v ) {
   console.log( k + " : " + v );
});
```

$.inArray()

Returns a value's index in an array, or -1 if the value is not in the array:

```
var myArray = [ 1, 2, 3, 5 ];

if ( $.inArray( 4, myArray ) !== -1 ) {
   console.log( "found it!" );
}
```

$.extend()

Changes the properties of the first object using the properties of subsequent objects:

```
var firstObject = { foo: "bar", a: "b" };
var secondObject = { foo: "baz" };

var newObject = $.extend( firstObject, secondObject );

console.log( firstObject.foo ); // "baz"
console.log( newObject.foo ); // "baz"
```

For jQuery collections, use .each():

.each() is used directly on a jQuery collection.

It iterates over each matched element in the collection and performs a callback on that object.

The index of the current element within the collection is passed as an argument to the callback.

The value (the DOM element in this case) is also passed, but the callback is fired within the context of the current matched element so the this keyword points to the current element as expected in other jQuery callbacks.

For example, given the following markup:

```
<ul>
   <li><a href="#">Link 1</a></li>
   <li><a href="#">Link 2</a></li>
   <li><a href="#">Link 3</a></li>
</ul>
```

each() may be used like so:

```
$( "li" ).each( function( index, element ){
   console.log( $( this ).text() );
});
```

```
$( "input" ).each( function( i, el ) {
   var elem = $( el );
   elem.val( elem.val() + "%" );
});

$( "input" ).val(function( index, value ) {
   return value + "%";
});
```

Selecting Form Elements

jQuery offers several pseudo-selectors that help find elements in forms.

These are especially helpful because it can be difficult to distinguish between form elements based on their state or type using standard CSS selectors.

:checked

Not to be confused with *:checkbox*, :checked targets *checked* checkboxes, but keep in mind that this selector works also for *checked* radio buttons, and <select> elements (for <select> elements only, use the :selected selector):

```
 $( "form :checked" );
```

The :checked pseudo-selector works when used with **checkboxes**, **radio buttons** and **selects**.

I :disabled

Using the :disabled pseudo-selector targets any <input> elements with the disabled attribute:

```
 $( "form :disabled" );
```

In order to get the best performance using :disabled, first select elements with a standard jQuery selector, then use .filter(":disabled"), or precede the pseudo-selector with a tag name or some other selector.

:enabled

Basically the inverse of the *:disabled* pseudo-selector, the :enabled pseudo-selector targets any elements that *do not* have a disabled attribute:

```
$( "form :enabled" );
```

In order to get the best performance using :enabled, first select elements with a standard jQuery selector, then use .filter(":enabled"), or precede the pseudo-selector with a tag name or some other selector.

:input

Using the :input selector selects all <input>, <textarea>, <select>, and <button> elements:

```
$( "form :input" );
```

:selected

Using the :selected pseudo-selector targets any selected items in <option> elements:

```
$( "form :selected" );
```

In order to get the best performance using :selected, first select elements with a standard jQuery selector, then use .filter(":selected"), or precede the pseudo-selector with a tag name or some other selector.

Selecting by type

jQuery provides pseudo selectors to select form-specific elements according to their type:

- :password
- :reset
- :radio
- :text
- :submit
- :checkbox
- :button
- :image
- :file

Cloning Elements

Methods such as .appendTo() move the element, but sometimes a copy of the element is needed instead. In this case, use .clone() first:

```
// Making a copy of an element.

// Copy the first list item to the end of the list:
$( "#myList li:first" ).clone().appendTo( "#myList" );
```

If you need to copy related data and events, be sure to pass true as an argument to .clone()

Creating New Elements

jQuery offers a trivial and elegant way to create new elements using the same $() method used to make selections:

```
// Creating new elements from an HTML string.
$( "<p>This is a new paragraph</p>" );
$( "<li class=\"new\">new list item</li>" );

// Creating a new element with an attribute object.
$( "<a/>", {
   html: "This is a <strong>new</strong> link",
   "class": "new",
   href: "foo.html"
});


// Getting a new element on to the page.

var myNewElement = $( "<p>New element</p>" );

myNewElement.appendTo( "#content" );

myNewElement.insertAfter( "ul:last" ); // This will remove the p from #content!

$( "ul" ).last().after( myNewElement.clone() ); //
```

```
/ Creating and adding an element to the page at the same time.
$( "ul" ).append( "<li>list item</li>" );

ar myItems = [];
var myList = $( "#myList" );

for ( var i = 0; i < 100; i++ ) {
   myItems.push( "<li>item " + i + "</li>" );
}

myList.append( myItems.join( "" ) );
```

jQuery.grep()

jQuery.grep(array, function [, invert])

Description: Finds the elements of an array which satisfy a filter function.
The original array is not affected.

```
<!doctype html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <title>jQuery.grep demo</title>
 <style>
 div {
  color: blue;
 }
 p {
  color: green;
  margin: 0;
 }
 span {
  color: red;
 }
 </style>
 <script src="https://code.jquery.com/jquery-1.10.2.js"></script>
</head>
<body>

<div></div>
```

```
<p></p>
<span></span>

<script>
var arr = [ 1, 9, 3, 8, 6, 1, 5, 9, 4, 7, 3, 8, 6, 9, 1 ];
$( "div" ).text( arr.join( ", " ) );

arr = jQuery.grep(arr, function( n, i ) {
  return ( n !== 5 && i > 4 );
});
$( "p" ).text( arr.join( ", " ) );

arr = jQuery.grep(arr, function( a ) {
  return a !== 9;
});

$( "span" ).text( arr.join( ", " ) );
</script>

</body>
</html>
```

jQuery.parseJSON()

jQuery.parseJSON(json)

Description: Takes a well-formed JSON string and returns the resulting JavaScript value.

Parse a JSON string.

```
var obj = jQuery.parseJSON( '{ "name": "John" }' );
alert( obj.name === "John" );
```

How do I determine the state of a toggled element?

You can determine whether an element is collapsed or not by using the :visible and :hidden selectors.

```
var isVisible = $( "#myDiv" ).is( ":visible" );

var isHidden = $( "#myDiv" ).is( ":hidden" );
```

jQuery Event Handling

jQuery offers convenience methods for most native browser events.

These methods — including .click(), .focus(), .blur(), .change(), etc. — are shorthand for jQuery's .on() method.

The **on** method is useful for binding the same handler function to multiple events, when you want to provide data to the event handler, when you are working with custom events, or when you want to pass an object of multiple events and handlers.

```
// Event setup using a convenience method
$( "p" ).click(function() {
   console.log( "You clicked a paragraph!" );
});

// Equivalent event setup using the `.on()` method
$( "p" ).on( "click", function() {
   console.log( "click" );
});
```

Inside the Event Handler Function

Every event handling function receives an event object, which contains many properties and methods.

The event object is most commonly used to prevent the default action of the event via the .preventDefault() method.

However, the event object contains a number of other useful properties and methods, including:

pageX, pageY

The mouse position at the time the event occurred, relative to the top left corner of the page display area (not the entire browser window).

type

The type of the event (e.g., "click").

which

The button or key that was pressed.

data

Any data that was passed in when the event was bound. For example:

```
/ Event setup using the `.on()` method with data
$( "input" ).on(
   "change",
   { foo: "bar" }, // Associate data with event binding
   function( eventObject ) {
      console.log("An input value has changed! ", eventObject.data.foo);
   }
);
```

preventDefault()

Prevent the default action of the event (e.g. following a link).

stopPropagation()

Stop the event from bubbling up to other elements.

Setting Up Multiple Event Responses

```
// Multiple events, same handler
$( "input" ).on(
   "click change", // Bind handlers for multiple events
   function() {
      console.log( "An input was clicked or changed!" );
   }
);

// Binding multiple events with different handlers
$( "p" ).on({
   "click": function() { console.log( "clicked!" ); },
   "mouseover": function() { console.log( "hovered!" ); }
});
```

Handling Events

Simple event binding

```
// When any <p> tag is clicked, we expect to see '<p> was clicked' in the console.
```

```
$( "p" ).on( "click", function() {
  console.log( "<p> was clicked" );
});
```

Many events, but only one event handler

```
/ When a user focuses on or changes any input element,
// we expect a console message bind to multiple events
$( "div" ).on( "mouseenter mouseleave", function() {
  console.log( "mouse hovered over or left a div" );
});
```

Many events and handlers

```
$( "div" ).on({
  mouseenter: function() {
    console.log( "hovered over a div" );
  },
  mouseleave: function() {
    console.log( "mouse left a div" );
  },
  click: function() {
    console.log( "clicked on a div" );
  }
});
```

The event object

```
$( "div" ).on( "click", function( event ) {
  console.log( "event object:" );
  console.dir( event );
});
```

Passing data to the event handler

You can pass your own data to the event object.

```
$( "p" ).on( "click", {
  foo: "bar"
}, function( event ) {
  console.log( "event data: " + event.data.foo + " (should be 'bar')" );
```

```
});
```

Binding events to elements that don't exist yet

This is called *event delegation*

```
$( "ul" ).on( "click", "li", function() {
  console.log( "Something in a <ul> was clicked, and we detected that it was an <li>
element." );
});
```

Connecting Events to Run Only Once

```
// Switching handlers using the `.one()` method
$( "p" ).one( "click", function() {
  console.log( "You just clicked this for the first time!" );
  $( this ).click(function() {
    console.log( "You have clicked this before!" );
  });
});
```

Disconnecting Events

```
// Unbinding all click handlers on a selection
$( "p" ).off( "click" );

// Unbinding a particular click handler, using a reference to the function
var foo = function() {
  console.log( "foo" );
};

var bar = function() {
  console.log( "bar" );
};

$( "p" ).on( "click", foo ).on( "click", bar );

// foo will stay bound to the click event
$( "p" ).off( "click", bar );
```

Stop the event from bubbling up to other elements.

/ Preventing a link from being followed

```
$( "a" ).click(function( event ) {
   var elem = $( this );
   if ( elem.attr( "href" ).match( "evil" ) ) {
      event.preventDefault();
      elem.addClass( "evil" );
   }
});
```

Understanding Event Delegation

Event delegation allows us to attach a single event listener, to a parent element, that will fire for all descendants matching a selector, whether those descendants exist now or are added in the future.

```
<html>
<body>
<div id="container">
   <ul id="list">
      <li><a href="http://domain1.com">Item #1</a></li>
      <li><a href="/local/path/1">Item #2</a></li>
      <li><a href="/local/path/2">Item #3</a></li>
      <li><a href="http://domain4.com">Item #4</a></li>
   </ul>
</div>
</body>
</html>
```

When an anchor in our #list group is clicked, we want to log its text to the console. Normally we could directly bind to the click event of each anchor using the .on() method:

```
// Attach a directly bound event handler
$( "#list a" ).on( "click", function( event ) {
   event.preventDefault();
   console.log( $( this ).text() );
});
```

While this works perfectly fine, there are drawbacks. Consider what happens when we add a new anchor after having already bound the above listener:

```
// Add a new element on to our existing list
$( "#list" ).append( "<li><a href='http://newdomain.com'>Item #5</a></li>" );
```

If we were to click our newly added item, nothing would happen. This is because of the directly bound event handler that we attached previously. Direct events are only attached to elements at the time the .on() method is called. In this case, since our new anchor did not exist when .on() was called, it does not get the event handler.

Event Propagation

Any time one of our anchor tags is clicked, a *click* event is fired for that anchor, and then bubbles up the DOM tree, triggering each of its parent click event handlers

- <a>
-
- <ul #list>
- <div #container>
- <body>
- <html>
- *document* root

This means that anytime you click one of our bound anchor tags, you are effectively clicking the entire document body! This is called *event bubbling* or *event propagation*.

```
// Attach a delegated event handler
$( "#list" ).on( "click", "a", function( event ) {
  event.preventDefault();
  console.log( $( this ).text() );
});
```

Triggering Event Handlers

jQuery provides a way to trigger the event handlers bound to an element without any user interaction via the .trigger() method.

```
<a href="http://learn.jquery.com">Learn jQuery</a>
// This will not change the current page
$( "a" ).trigger( "click" );
```

Creating Custom Plugin's in jQuery:

Sometimes you want to make a piece of functionality available throughout your code
$("a").css("color", "red");

Whenever you use the $ function to select elements, it returns a jQuery object.

This object contains all of the methods you've been using (.css(), .click(), etc.) and all of the elements that fit your selector.

The jQuery object gets these methods from the $.fn object.

This object contains all of the jQuery object methods, and if we want to write our own methods, it will need to contain those as well.

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

$( "a" ).greenify(); // Makes all the links green.
```

Chaining

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

$( "a" ).greenify().addClass( "greenified" );
```

Working with jQueryUI

It is also an API for javascript.

It includes predefined functions for UI related Widgets.

jQuery UI functions:

datepicker()
tabs()
tooltip()
autocomplete()
dialog()
draggagle()
drappable()

To use all these functions, we just need to download jqueryui api and embed that .js files and .css files into your document.

Then you can use these functions.

Samples: for datepicker

```
<!doctype html>
<html lang="en">
<head>
 <meta charset="utf-8">
 <title>jQuery UI Datepicker - Default functionality</title>
 <link rel="stylesheet" href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
 <script src="//code.jquery.com/jquery-1.10.2.js"></script>
 <script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
 <link rel="stylesheet" href="/resources/demos/style.css">
 <script>
 $(function() {
   $( "#datepicker" ).datepicker();
 });
 </script>
</head>
<body>

<p>Date: <input type="text" id="datepicker"></p>


</body>
</html>
```

Samples: for autocomplete

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>jQuery UI Autocomplete - Default functionality</title>
  <link rel="stylesheet" href="//code.jquery.com/ui/1.11.4/themes/smoothness/jquery-ui.css">
  <script src="//code.jquery.com/jquery-1.10.2.js"></script>
  <script src="//code.jquery.com/ui/1.11.4/jquery-ui.js"></script>
  <link rel="stylesheet" href="/resources/demos/style.css">
  <script>
  $(function() {
    var availableTags = [
      "ActionScript",
      "AppleScript",
      "Asp",
      "BASIC",
      "C",
      "C++",
      "Clojure",
      "COBOL",
      "ColdFusion",
      "Erlang",
      "Fortran",
      "Groovy",
      "Haskell",
      "Java",
      "JavaScript",
      "Lisp",
      "Perl",
      "PHP",
      "Python",
      "Ruby",
      "Scala",
      "Scheme"
    ];
    $( "#tags" ).autocomplete({
      source: availableTags
    });
  });
  </script>
</head>
<body>

<div class="ui-widget">
  <label for="tags">Tags: </label>
  <input id="tags">
</div>


</body>
</html>
```

For Slider:

```
 <script>
 $(function() {
   $( "#slider" ).slider();
 });
 </script>
</head>
<body>

<div id="slider"></div>


</body>
```

For tooltip:

```
 <script>
 $(function() {
   $( document ).tooltip();
 });
 </script>
 <style>
 label {
   display: inline-block;
   width: 5em;
 }
 </style>
</head>
<body>

<p><a href="#" title="That&apos;s what this widget is">Tooltips</a> can be attached to any element.
When you hover
the element with your mouse, the title attribute is displayed in a little box next to the element, just like a
native tooltip.</p>
<p>But as it's not a native tooltip, it can be styled. Any themes built with
<a href="http://jqueryui.com/themeroller/" title="ThemeRoller: jQuery UI&apos;s theme builder
application">ThemeRoller</a>
will also style tooltips accordingly.</p>
<p>Tooltips are also useful for form elements, to show some additional information in the context of each
field.</p>
<p><label for="age">Your age:</label><input id="age" title="We ask for your age only for statistical
purposes."></p>
<p>Hover the field to see the tooltip.</p>


</body>
```

For tabs:

```
 <script>
 $(function() {
   $( "#tabs" ).tabs();
 });
 </script>
</head>
<body>

<div id="tabs">
 <ul>
   <li><a href="#tabs-1">Nunc tincidunt</a></li>
   <li><a href="#tabs-2">Proin dolor</a></li>
   <li><a href="#tabs-3">Aenean lacinia</a></li>
 </ul>
 <div id="tabs-1">
   <p>Proin elit arcu, rutrum commodo, vehicula tempus, commodo a, risus. Curabitur nec arcu. Donec sollicitudin mi sit amet mauris. Nam elementum quam ullamcorper ante. Etiam aliquet massa et lorem. Mauris dapibus lacus auctor risus. Aenean tempor ullamcorper leo. Vivamus sed magna quis ligula eleifend adipiscing. Duis orci. Aliquam sodales tortor vitae ipsum. Aliquam nulla. Duis aliquam molestie erat. Ut et mauris vel pede varius sollicitudin. Sed ut dolor nec orci tincidunt interdum. Phasellus ipsum. Nunc tristique tempus lectus.</p>
 </div>
 <div id="tabs-2">
   <p>Morbi tincidunt, dui sit amet facilisis feugiat, odio metus gravida ante, ut pharetra massa metus id nunc. Duis scelerisque molestie turpis. Sed fringilla, massa eget luctus malesuada, metus eros molestie lectus, ut tempus eros massa ut dolor. Aenean aliquet fringilla sem. Suspendisse sed ligula in ligula suscipit aliquam. Praesent in eros vestibulum mi adipiscing adipiscing. Morbi facilisis. Curabitur ornare consequat nunc. Aenean vel metus. Ut posuere viverra nulla. Aliquam erat volutpat. Pellentesque convallis. Maecenas feugiat, tellus pellentesque pretium posuere, felis lorem euismod felis, eu ornare leo nisi vel felis. Mauris consectetur tortor et purus.</p>
 </div>
 <div id="tabs-3">
   <p>Mauris eleifend est et turpis. Duis id erat. Suspendisse potenti. Aliquam vulputate, pede vel vehicula accumsan, mi neque rutrum erat, eu congue orci lorem eget lorem. Vestibulum non ante. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos himenaeos. Fusce sodales. Quisque eu urna vel enim commodo pellentesque. Praesent eu risus hendrerit ligula tempus pretium. Curabitur lorem enim, pretium nec, feugiat nec, luctus a, lacus.</p>
   <p>Duis cursus. Maecenas ligula eros, blandit nec, pharetra at, semper at, magna. Nullam ac lacus. Nulla facilisi. Praesent viverra justo vitae neque. Praesent blandit adipiscing velit. Suspendisse potenti. Donec mattis, pede vel pharetra blandit, magna ligula faucibus eros, id euismod lacus dolor eget odio. Nam scelerisque. Donec non libero sed nulla mattis commodo. Ut sagittis. Donec nisi lectus, feugiat porttitor, tempor ac, tempor vitae, pede. Aenean vehicula velit eu tellus interdum rutrum. Maecenas commodo. Pellentesque nec elit. Fusce in lacus. Vivamus a libero vitae lectus hendrerit hendrerit.</p>
 </div>
</div>
```

For dialog:

```
 <script>
 $(function() {
   $( "#dialog" ).dialog();
 });
 </script>
</head>
<body>

<div id="dialog" title="Basic dialog">
 <p>This is the default dialog which is useful for displaying information. The dialog window can be moved, resized and closed with the 'x' icon.</p>
</div>


</body>
```

jQuery Mobile

jQuery Mobile is the easiest way to build sites and apps that are accessible on all popular smartphone, tablet, and desktop devices.

This framework provides a set of touch-friendly UI widgets and an AJAX-powered navigation system to support animated page transitions.

```
<!doctype html>
<html>
<head>
  <title>My Page</title>
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="stylesheet" href="https://code.jquery.com/mobile/1.2.0/jquery.mobile-
1.2.0.min.css">
  <script src="https://code.jquery.com/jquery-1.8.2.min.js"></script>
  <script src="https://code.jquery.com/mobile/1.2.0/jquery.mobile-
1.2.0.min.js"></script>
</head>
<body>
  <div data-role="page">

    <div data-role="header">
      <h1>My Title</h1>
    </div><!-- /header -->

    <div data-role="content">
      <p>Hello world</p>
    </div><!-- /content -->

    <div data-role="footer">
      <h4>My Footer</h4>
    </div><!-- /footer -->

  </div><!-- /page -->
</body>
</html>
```

Make a Listview

jQuery Mobile includes a diverse set of common listviews that are coded as lists with a data-role="listview" added.

```
<ul data-role="listview" data-inset="true" data-filter="true">
   <li><a href="#">Acura</a></li>
   <li><a href="#">Audi</a></li>
   <li><a href="#">BMW</a></li>
   <li><a href="#">Cadillac</a></li>
   <li><a href="#">Ferrari</a></li>
</ul>
```

Add a Slider

```
<form>
   <label for="slider-0">Input slider:</label>
   <input type="range" name="slider" id="slider-0" value="25" min="0" max="100" />
</form>
```

Make a Button

```
<a href="#" data-role="button" data-icon="star">Star button</a>
```