IT614

Introduction to jQuery

jQuery is a very powerful JavaScript library designed to simplify writing JavaScript programs (library roughly means, it provides a set of predefined functions, which you can use in your JavaScript programs). You incorporate jQuery functions into JavaScript programs. It was released in January 2006 by John Resig. Used by over 65% of the 10,000 most visited websites, jQuery is the most popular JavaScript library in use today.

Before you can start using jQuery functions, you need to download (nothing to install), jQuery package to your computer. Go to jQuery download page: http://jquery.com/download/. This is the link you want: Download the compressed, production jQuery 2.0.3 (the latest version will have a different number). Don't double click the link, but right click on this link and select Save link as… and save it on your computer. The downloaded file is called jquery-2.0.3.min.js (you may have a different version number). This is your jQuery package. You should leave the package in a folder where you keep your HTML/JavaScript files. Once you have this package, you need to refer to this in all your HTML/JavaScript files. This is how you do this: Within <header>…</header>, include this line
<script src="jquery-2.0.3.min.js">.
There is an alternative to downloading jQuery library to your computer and making it available to your scripts (as explained above). You can directly access jQuery library from one of the several servers called CDNs (Content Delivery Networks). Here is one:
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>.
This means, instead of including <script src="jquery-2.0.3.min.js"></script> in the header, include <script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>.

Which comes first: jQuery or HTML?

jQuery selects DOM element(s) from an HTML document and then do something with them using jQuery methods. Before jQuery can work on the HTML elements, the browser must have the placed the HTML's DOM in memory. This means, as a general rule, the HTML code must appear before jQuery statements in your document.

Thus, the general format of your HTML document with jQuery looks like this:

```
<html>
<head> ...</head>
<body>
      Your HTML code
  <script>
      jQuery code
  </script>
</body>
</html>
```

There is another way of accomplishing this. You can make jQuery code wait to execute until all HTML's DOM is created completely. This is done by including your jQuery code within,

```
$(function(){
  Your jQuery code
}
```

We will not use this in the beginning.

1. The jQuery function $() or jQuery()

A very important and basic jQuery function is **jQuery().** This will be used frequently. Because it is used extensively, jQuery has provided a simpler syntax for it: **$()**.

So **$()** and **jQuery()** are equivalent.

This is what jQuery basically does: "select some DOM elements and do something with them." As you will see, **$()** does the first part of "selecting elements."

Note: If some of these things are not clear now, continue reading, everything becomes clear when you see examples.

2. The function call $('*selector*')

The argument is a selector. You can the usual CSS3 selectors, as well as some non-standard selectors (which we will see later).

Syntax:

$('*selctor*')

This returns a set of elements matching the argument *selector*. The value returned is a jQuery object, which contains all the DOM elements matching the specified selector.

Example: days.html

```
<!DOCTYPE html>
<html>
<head><script src="jquery.js"></script></head>
 <body>
      <div class="weekday" day="first"> Monday</div>
      <div class="weekday"> Tuesday</div>
      <div class="weekday"> Wednesday </div>
      <div class="weekday"> Thursday</div>
      <div class="weekday">Friday <span>TGIF</span></div>
      <div id="wkend1">Saturday</div>
      <div id="wkend2" day="last">Sunday</div>
      <hr>
     <script>
          var divs = $('div');
     </script>
</body>
</html>
```

Here the variable **divs** is a jQuery object which contains all the **div** elements. There are 7 **div**. We can access them using the array notation: divs[0] refers to <div class="weekday" day="first"> Monday</div>, divs[1] refers to <div class="weekday"> Tuesday</div>, and so on, divs[6] refers to <div id="wkend2" day="last">Sunday</div>. We will see how we can actually print the text Monday, Tuesday and so on.

Note:

For reasons explained earlier, we always include <script>…</script> before. </body>.

- **length and size()**

The call **$(**'*selector*'**)** returns a jQuery object, which can be treated like an array. It is not an array, but for all practical purposes this jQuery object can be treated like an array whose individual items are elements. These individual elements can be accessed using the usual subscript notation. Subscripts range from 0 through (length-1). To get the length of the object, you can either use the **length** property or **size()** method.

Example:

Complete program:

```
<!DOCTYPE html>
<html>
<head><script src="jquery.js"></script></head>
 <body>
 <div class="weekday" day="first"> Monday</div>
 <div class="weekday"> Tuesday</div>
 <div class="weekday"> Wednesday </div>
 <div class="weekday"> Thursday</div>
 <div class="weekday">Friday <span>TGIF</span></div>
 <div id="wkend1">Saturday</div>
 <div id="wkend2" day="last">Sunday</div>
 <hr>
<script>
   var divs = $('div');
   document.write("<p>Number of elements : ",divs.length);
   document.write("<p>Number of elements : ",divs.size());
</script>
</body>
</html>
```

Both the write statements output the same value:

```
Number of elements : 7
Number of elements : 7
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week8jQuery/days.html

- **text() and innerHTML**

As described earlier, $('*selector*') returns an array of HTML elements. The jQuery method **text()** returns the combined text contents of each element in the set of matched elements, including their descendants. If you like to see the element context text, you should use **innerHTML** (we have seen this before).

Example:

```
<script>
  var divs = $('div');
  document.write("<p>Text content of divs object: ",divs.text());
</script>
```

Output:

```
Text content of divs object: Monday Tuesday Wednesday ThursdayFriday
TGIFSaturdaySunday
```

- **Text content of an elements in jQuery object**

The items in divs object are not objects, but elements. You cannot use **text()** on individual items in divs. We use **innerHTML**.

Example:

```
<script>
  var divs = $('div');
  document.write("<p>Text content of divs[1] object: ",divs[1].innerHTML);
</script>
```

Output:

```
Text content of divs[1] object: Tuesday
```

- **Converting an element to a jQuery object**

Each item, divs[i] is an element. To convert each of this into a jQuery object, you need to apply $(divs[i]). Once you convert an element into an object, you can use the **text()** method to retrieve text content.

Example:

```
<script>
  var divs = $('div');
  document.write("<p>Text content of divs[1] object: ",$(divs[1]).text());
</script>
```

Output:

```
Text content of divs[1] object: Tuesday
```

- **Applying CSS property to a jQuery object**

We can apply any CSS property to a jQuery object, which in effect applies to the element in the jQuery object. Notice the special syntax in CSS declaration.

Syntax:

jQueryObject.**css**({"CSS property":"value"});

To include multiple CSS declarations, use the syntax:

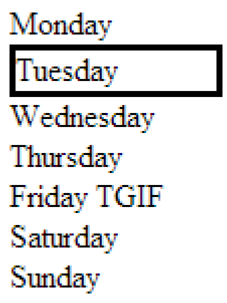
jQueryObject.**css**({"CSS property":"value", "CSS property":"value", "CSS property":"value"});

Example:

```
<script>
  var divs = $('div');
  var div1 = $(divs[1]);
  div1.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

This example uses three CSS properties.

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

3. Basic selectors

One of the fundamental operation jQuery does is to select elements. As you will see jQuery provides a large number of ways to select elements. You will be able to pick any element you want how nested or how deep they are in the DOM tree. Let us start with basics.
You can use all the CSS selectors in **$('*selector*').**

- **$('*')**

Selects all elements.

- **$('*.classValue*')**

Selects all elements with the given class.

- **$('#*idValue*')**

Selects all elements with the given id.

- **$('*element*')**

Selects all elements with the given tag name.

- **$('*selector*','*selector*',...)**

Selects the combined results of all the specified selectors.

---

4. Attribute selectors

- **$("[*attributeName*='*value*']")**

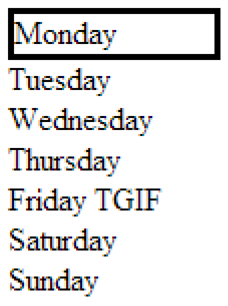
Selects elements that have the specified attribute with the specified value.

Example:

```
<script>
  var divs = $('[day="first"]');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Matches elements with attribute **day** and value **first**.

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

- **$("[*attributeName*|='*value*']")**

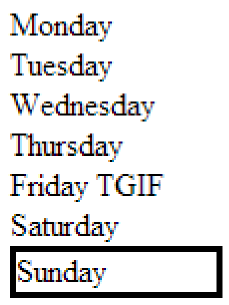
Selects elements that have the specified attribute with a value either equal to a given string or starting with that string followed by a hyphen (-).

Example:

```
<script>
  var divs = $('[day|="last"]');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Matches elements with attribute day and value last.

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

➢ **$("[*attributeName**='*value*']")**

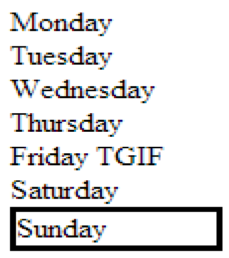
Selects elements that have the specified attribute with a value containing the specified substring.

Example:

```
<script>
  var divs = $('[day*="la"]');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Matches elements with attribute day and value containing the string la.

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

➢ **$("[*attributeName*$='*value*']")**

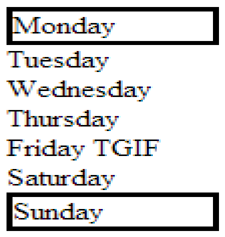
Selects elements that have the specified attribute with a value ending exactly with a given string. The comparison is case sensitive.

Example:

```
<script>
  var divs = $('[day$="st"]');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Matches elements with attribute **day** and value ending in "st".

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

➢ $("[***attributeName***^='***value***']")

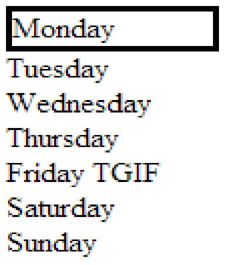
Selects elements that have the specified attribute with a value starting with the specified value. The comparison is case sensitive.

Example:

```
<script>
  var divs = $('[day^="f"]');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Selects all the elements with attribute **day** and value starting with **f**.

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

➢ $("[***attributeName***]")

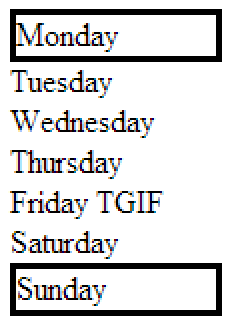
Selects elements that have the specified attribute, with any value.

Example:

```
<script>
  var divs = $('[day]');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Selects all the elements containing the attribute **day** (with any value).

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

- **$("[*attributeName*='*value*'][*attributeName*='*value*']")**

Selects all elements specified by multiple attribute selectors.

- **$("[*attributeName*!='*value*']")**

Selects elements that either don't have the specified attribute, or do have the specified attribute but not the specified value.

Example:

```
<script>
  var divs = $('[da !="first"]');
  divs.css("font-size","30px");
</script>
```

Selects elements, which do not have the attribute **da**. That is all the elements.

5. Using filters in selectors

You can narrow the result from $() by including a filter for selections.

Syntax:

$('selector:filter')

This returns the subset after applying filter.

- **:eq() selector filter**

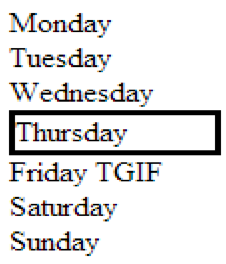
Syntax:

$('selector:eq(n)')

Selects the element at n within the matched set (counting starts from 0) If n is negative, counting starts from the end.

Example:

```
<script>
  var divs = $('div:eq(3)');
  divs.css("font-size","30px");
</script>
```



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

- **:even selector filter**

Syntax:

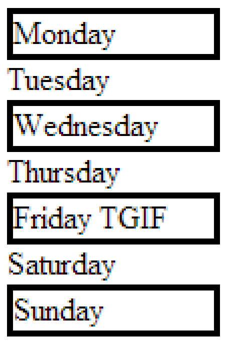
$('selector:even')

Selects even elements, zero-indexed.

Example:

```
<script>
  var divs = $('div:even');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

- **:odd selector filter**

Syntax:

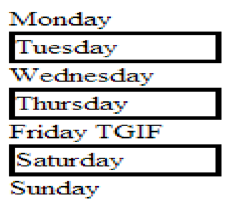
$('selector:odd')

Selects odd elements, zero-indexed.

Example:

```
<script>
  var divs = $('div:odd');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

- **:first selector filter**

Syntax:

$('selector:first')

Selects first element.

Example:

```
<script>
  var divs = $('div:first');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

- **:last selector filter**

Syntax:

$('selector:last')

Selects last element.

Example:

```
<script>
  var divs = $('div:last');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

- **:gt(n) selector filter**

Syntax:

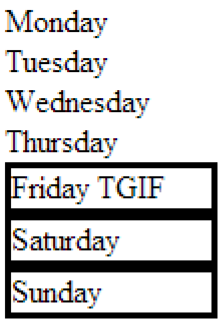
$('selector:gt(n)')

Select all elements at an index greater than n within the matched set.

Example:

```
<script>
  var divs = $('div:gt(3)');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

- **:lt(n) selector filter**

Syntax:

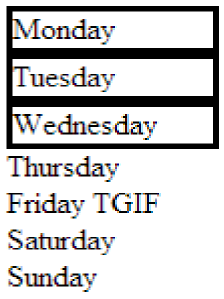
$('selector:lt(n)')

Select all elements at an index less than n within the matched set.

Example:

```
<script>
  var divs = $('div:lt(3)');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

- **:header selector filter**

Syntax:

$('selector:header')

Selects all elements that are headers, like h1, h2, h3 and so on.

➢ **:not() selector filter**

Syntax:

$('selector:not(subselector)')

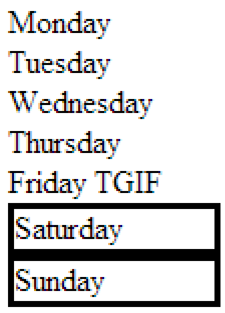
Selects all elements that do not match the subselector.

Example:

```
<script>
    var divs = $('div:not(.weekday)');
    divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Selects all div elements leaving out div tags with weekday class names.

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

6. Using Child filters in selectors

http://api.jquery.com/category/selectors/child-filter-selectors/

➢ **:first-child Selector filter**

Syntax:

$('*selector*:**first-child**')

Selects all elements that are the first child elements of their parents. This is how you understand this: Look at the parent element of the specified *selector*. Look at the first-child of the parent. If it is selector kind then it is selected. This is done for every parent of the specified selector.

➢ **:last-child Selector filter**

Syntax:

$('*selector*:**last-child**')

Selects all elements that are the last child elements of their parents. This is how you understand this: Look at the parent element of the specified *selector*. Look at the last-child of the parent. If it is selector kind then it is selected. This is done for every parent of the specified selector.

Example:

Let us modify out days.html:

```
<!DOCTYPE html>
<html>
<head><script src="jquery.js"></script></head>
 <body>
     <div class="weekday" day="first"> Monday</div>
     <div class="weekday"> Tuesday</div>
     <div class="weekday"> Wednesday <span>middle</span><h3>day</h3></div>
     <div class="weekday"> Thursday</div>
     <div class="weekday"><h3>Friday</h3><span>TGIF</span></div>
     <div id="wkend1">Saturday</div>
     <div id="wkend2" day="last">Sunday</div>
     <hr>
</body>
</html>
```

Notice I have made changes from the original days.html document. The changes are on these lines:

`<div class="weekday"> Wednesday <span>middle</span><h3>day</h3></div>`

and

`<div class="weekday"><h3>Friday</h3><span>TGIF</span></div>`

Let us take,

```
$('span:first-child');
```

First look at the parent element of **<span>.** There are two:

1. `<div class="weekday"> Wednesday <span>middle</span><h3>day</h3></div>`

and

2. `<div class="weekday"><h3>Friday</h3><span>TGIF</span></div>`

The first child of (1) is `<span>middle</span>`, which is a `<span>` tag. So `<span>middle</span>` is selected.

The first child of (2) is <h3>Friday</h3>, which is NOT a <span> tag. So `<h3>Friday</h3>` is NOT selected.

So $('span:first-child'); matches one element So <span>middle</span>.

Let us take,
$('span:last-child');

First look at the parent element of **<span>.** There are two (same as before):

1. <div class="weekday"> Wednesday <span>middle</span><h3>day</h3></div>
   and
2. <div class="weekday"><h3>Friday</h3><span>TGIF</span></div>

The last child of (1) is <h3>day</h3>, which is NOT a <span> tag. So <h3>day</h3>, is NOT selected.
The first child of (2) is <span>TGIF</span>, which is a <span> tag. So <span>TGIF</span> is selected.

So $('span:first-child'); matches one element So <span>TGIF</span>.

Note:
There are others in this category. See http://api.jquery.com/category/selectors/child-filter-selectors/ for more information on them.

7. Using content filters in selectors

http://api.jquery.com/category/selectors/content-filter-selector/

➢ **:contains("*text*") Selector filter**

Selects all elements that contain the specified text. "*text*" is case sensitive.

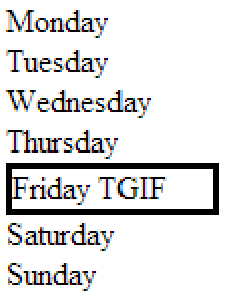
Syntax:
$('*selector*:contains("text"))

Example:
We will use the original days.html.

```
<!DOCTYPE html>
<head><script src="jquery.js"></script></head>
 <body>
```

```
    <div class="weekday" day="first"> Monday</div>
    <div class="weekday"> Tuesday</div>
    <div class="weekday"> Wednesday </div>
    <div class="weekday"> Thursday</div>
    <div class="weekday">Friday <span>TGIF</span></div>
    <div id="wkend1">Saturday</div>
    <div id="wkend2" day="last">Sunday</div>
    <hr>
   <script>
       var divs = $('div:contains("Fri")');
       divs.css({"width":"100px","height":"20px","border-style":"solid"});
   </script>
</body>
</html>
```

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

➢ **:has(*selector*) Selector filter**

Notice that the argument to **has** is a selector**. :has(selector)** selects elements which contain at least one element that matches the specified selector.

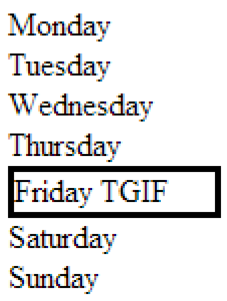
Syntax:

$('selector:**has(**selector))

Example:

```
<script>
  var divs = $('div:has(span)');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

8. Hierarchy selectors

http://api.jquery.com/category/selectors/hierarchy-selectors/

- **('*parentSelector*>*childSelector*') hierarchy selector**

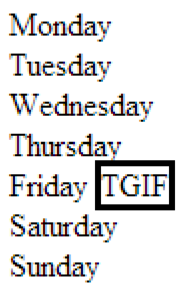
Selects the specified *childSelector*, which is a child of specified *parentSelector*.

Example:

```
<script>
  var divs = $('div>span');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

There exists a <span> within <div>. It is selected.

Output:

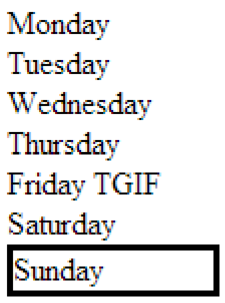


```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

Example:

```
<script>
  var divs = $('body>div#wkend2');
  divs.css({"width":"100px","height":"20px","border-style":"solid"});
</script>
```

Output:



```
Monday
Tuesday
Wednesday
Thursday
Friday TGIF
Saturday
Sunday
```

Note:

There are other hierarchy selectors. Read for more information: http://api.jquery.com/category/selectors/hierarchy-selectors/

9. Effects

We will see several interesting jQuery features, such as hiding an element, fading-in, fading-out elements and so on.

Before we discuss effects features, let us talk about jQuery "mouse button clicked event." JavaScript is an event driven programming language. An event is an action by the user. Examples of events are: clicking a button, moving the mouse pointer on a text box, entering text to a text box, changing the contents of a text box, and so on. We can make certain code in our program execute automatically when an event occurs. We will discuss all the events much later. But for now, we will discuss one event: "mouse button clicked".

10. Mouse button clicked event

We will discuss the event, "The mouse pointer is over an element, and the mouse button is pressed and released."

A jQuery method **click()** is executed each time the event is triggered. This method is called an event handler.

Syntax:

```
$('selection').click(function() {
    Body of the function
});
```

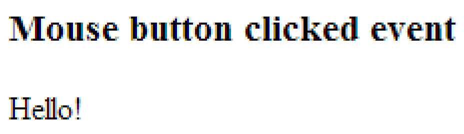
Study the syntax well. It is not like the usual function definition. Observe the (…) and {…} carefully.

Here selection is the element on which the event has occurred. In the body of the function, you include code to be executed when the event occurs.

Example:

```
<!DOCTYPE html>
<html lang="en">
<head><script src="jquery.js"></script>
<title>Effects example</title>
</head>
<body>
    <h3>Mouse button clicked event</h3>
    <p> Hello!
    <script>
    $( "p" ).click(function() {
      alert( "You clicked Hello!" );
    });
    </script>
</body>
</html>
```

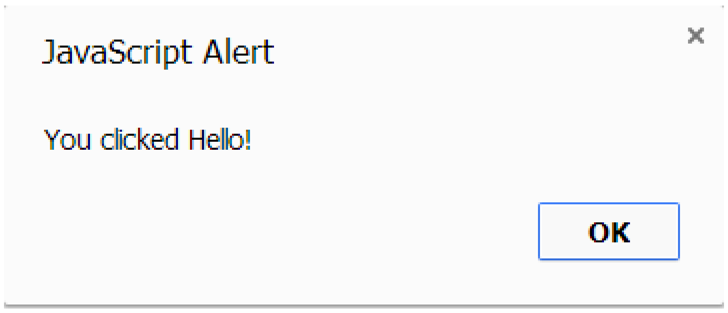
Output:



Mouse button clicked event

Hello!

When you click Hello!, you will see the alert box:



JavaScript Alert

You clicked Hello!

Here the handler method is,

```
$( "p" ).click(function() {
  alert( "You clicked Hello!" );
});
```

The <p> element is selected. **click()** says, when the element is clicked, execute the body of the function, which is just an alert box.

Before we start discussing effects, one more important concept:

- **The this keyword**

The keyword **this** is a very important concept in every object oriented programming language. Important in JavaScript/jQuery as well.

The keyword **this** refers the object on which a method is being invoked. Informally, understand the keyword **this** as the current object which is using this method. I am sure it is not clear. It can only become clear in contexts. We will see some examples soon.
We will using this event in the examples below. Let us now start discussing effects.

11. Arguments in effects methods

We will start exploring jQuery methods related effects. Most of the effects methods produce some animation and they take three arguments: *method(arg1, arg2, arg3).*
Because these three arguments appear in most of the methods we are going to see, let use discuss the three arguments in detail once here. Remember that are applicable in all the methods below.

The argument *arg1*:

The argument arg1 is called "duration." The value of this is just a number specifying how long it should take for the effect to complete. The number specified is in milliseconds. This argument is optional. If it is not included, the default value is 400 (400 milliseconds).
Instead of numbers, you can use "slow" and "fast" for duration. "slow" is equivalent to 200 and "fast" is equivalent to 600.

The argument *arg2*:

The argument arg2 is called "easing." This specifies the speed at which the animation progresses at different points within the animation. There are two possible values: swing and linear. The effect of swing easing: animation moves slower in the beginning and at the end, and slower in the middle. The effect of linear easing: animation moves at constant speed. This argument is optional. If it is not included, the default is "swing".

The argument *arg3*:

The argument arg3 is called "complete". This specifies a callback function to be fired once the animation is complete. This argument is optional. The default is "do nothing." We will not include this argument in our examples below.

Note:

All three arguments are optional. It means we can use the effects methods without any argument. In this case the default values be used.

We will now start effects methods.

12. The .hide() effect

This produces an animation of hiding an element.

Syntax:

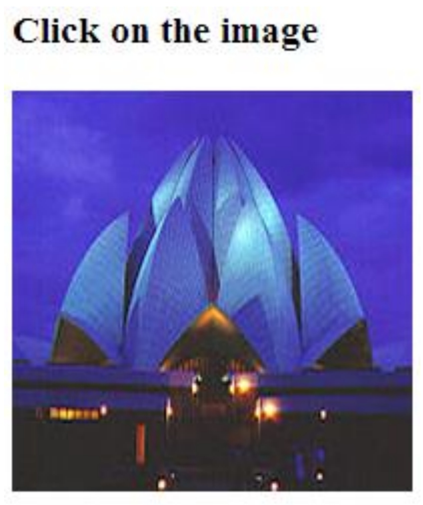
$('selector').hide(*arg1,arg2,arg3*);

Hides the selected element.

Example:

```
<!DOCTYPE html>
<head><script src="jquery.js"></script>
<title>Effects example</title>
</head>
<body>
    <h3>Mouse button clicked event</h3>
    <img src="lotus.jpg" alt="lotus" width=200 height=200></img>
    <script>
      $( "img" ).click(function() {
          $(this).hide(1000,"linear");
      });
    </script>
</body>
</html>
```

Output:



When you click on the image, the image is hidden.

13. The .show() effect

This produces an animation of showing an element.

Syntax:

$('selector').show(*arg1,arg2,arg3*);

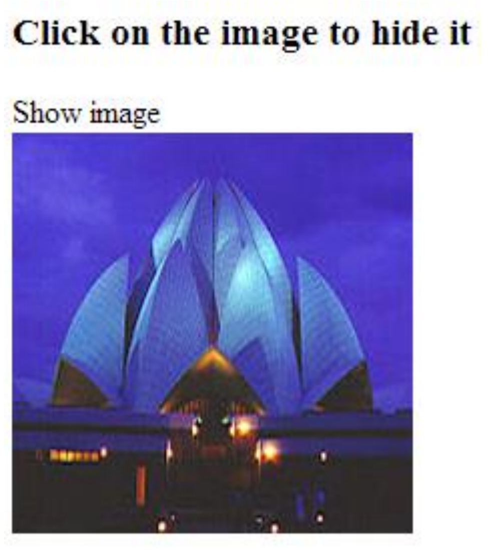
Shows the selected element.

Example:

```
<!DOCTYPE html>
<head>
    <script src="jquery.js"></script>
    <title>Effects example</title>
</head>
<body>
    <h3>Click on the image to hide it</h3>
    <div>Show image</div>
    <img src="lotus.jpg" alt="lotus" width=200 height=200></img>
    <script>
      $( "img" ).click(function() {
          $(this).hide();
      });
      $('div').click(function() {
          $("img").show(1000,"linear");
```

```
        });
    </script>
</body>
</html>
```

Output:



Click on the image to hide it

Show image

When you click on the image, it is hidden. When you click on "Show image", the image comes back.

14. The .toggle() effect

This toggles between hiding and showing an element.

Syntax:

$('selector').toggle(*arg1*,*arg2*,*arg3*);

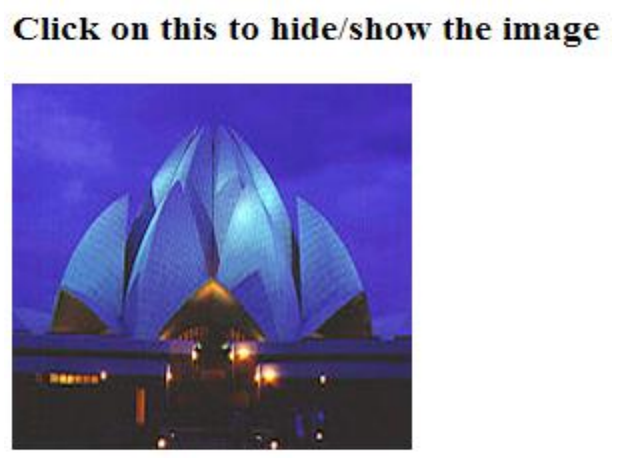
Toggles the selected element.

Example:

```
<!DOCTYPE html>
<head>
    <script src="jquery.js"></script>
    <title>Effects example</title>
</head>
<body>
    <h3>Click on this to hide/show the image</h3>
    <img src="lotus.jpg" alt="lotus" width=200 height=200></img>
    <script>
```

```
        $( "h3" ).click(function() {
          $('img').toggle();
        });
    </script>
</body>
</html>
```

Output:



Click on this to hide/show the image

15. The .fadeIn() effect

This fades in the matched element.

Syntax:

$('selector').fadeIn(*arg1,arg2,arg3*);

Fades in the selected element.

Example:

For this example, first an element must not be visible. To do this use, CSS property **display** with value **none**.

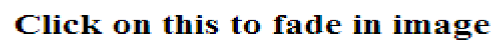
```
<!DOCTYPE html>
<head>
    <script src="jquery.js"></script>
    <title>Effects example</title>
    <style>
      img{
        display:none;
      }
    </style>
</head>
<body>
    <h3>Click on this to fade in the image</h3>
    <img src="lotus.jpg" alt="lotus" width=200 height=200></img>
    <script>
      $( "h3" ).click(function() {
        $('img').fadeIn(3000);
      });
    </script>
</body>
</html>
```

Output:

Because of this code,

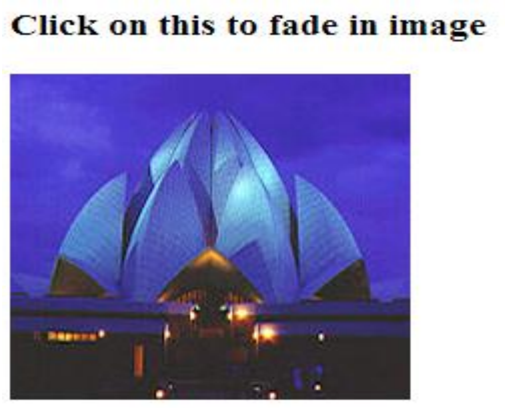
```
img{
  display:none;
}
```

The image does not show.



Click on this to fade in image

Now click the string, the image will fade in:



Click on this to fade in image

16. The .fadeOut() effect

This fades out the matched element.

Syntax:

$('selector').fadeOut(*arg1,arg2,arg3*);

Fades out the selected element.

17. The .fadeTo() effect

This fades out the matched element to a specified opacity.

Syntax:

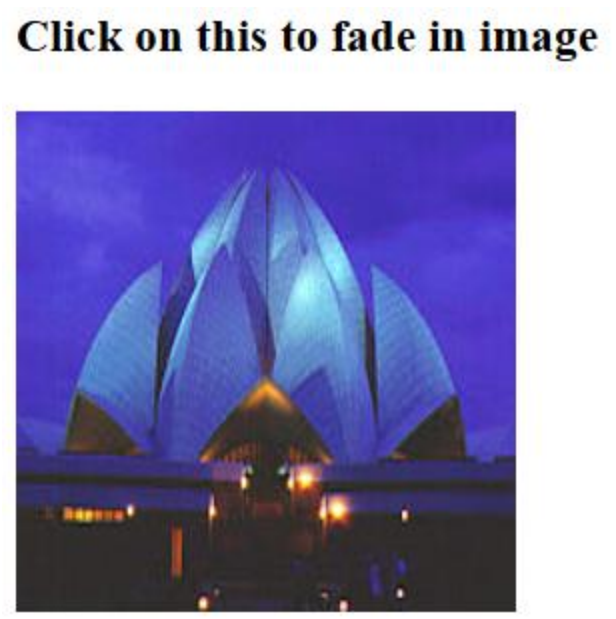
$('selector').fadeTo(*arg1,arg2,arg3*);

Fades out the selected element.

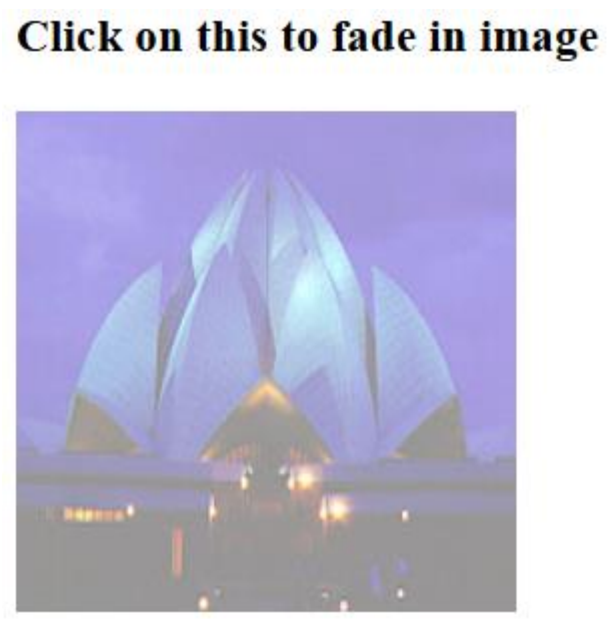
Example:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <script src="jquery.js"></script>
    <title>Effects example</title>
    <style>
    img{
      display:block;
    }
    </style>
</head>
<body>
    <h3>Click on this to fade in image</h3>
    <img src="lotus.jpg" alt="lotus" width=200 height=200></img>
    <script>
      $( "h3" ).click(function() {
        $('img').fadeTo(3000,0.5);
      });
    </script>
</body>
</html>
```

Output:


Click on this to fade in image

When you click the string, the image fades to,


Click on this to fade in image

18. The .fadeToggle() effect

This fades in/out the matched element.

Syntax:

$('selector').fadeToggle (*arg1*,*arg2*,*arg3*);

Toggles between fade out (completely) and fade in.

19. The .slideDown() effect

This slides down a hidden matched element.

Syntax:

$('selector').slideDown(*arg1,arg2,arg3*);

Slides down the selector element (if it is hidden).

20. The .slideUp() effect

This slides up a visible matched element.

Syntax:

$('selector').slideUp(*arg1,arg2,arg3*);

Slides up the selector element (if it is visible).

Example:

```
<!DOCTYPE html>
<html><head>
    <script src="jquery.js"></script>
    <title>Effects example</title>
    <style>
      img{
        display:none;
      }
    </style></head>
<body>
    <h3 id="one">Click here to slide up</h3>
    <h3 id="two">Click here to slide down</h3>
    <img src="lotus.jpg" alt="lotus" width=200 height=200></img>
    <script>
      $( "#one" ).click(function() {
        $('img').slideUp(1000);
      });
      $( "#two" ).click(function() {
        $('img').slideDown(1000);
      });
    </script>
</body>
</html>
```

Output:

The image is initially hidden.

Click here to slide up

Click here to slide down

Click slide down first,

Click here to slide up

Click here to slide down

Now click slide up.

Click here to slide up

Click here to slide down

21. The .slideToggle() effect

This toggles a matched element.

Syntax:

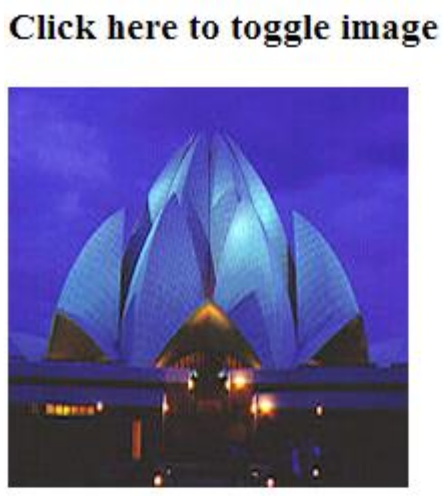
$('selector').slideToggle(arg1,arg2,arg3);

Toggles the selector element.

Example:

```
<!DOCTYPE html>
<html>
<head>
    <script src="jquery.js"></script>
    <title>Effects example</title>
</head>
<body>
    <h3 id="one">Click here to toggle image</h3>
    <img src="lotus.jpg" alt="lotus" width=200 height=200></img>
    <script>
      $( "#one" ).click(function() {
        $('img').slideToggle(1000);
      });
    </script>
</body>
</html>
```

Output:



Click here to toggle image

Try clicking a couple of times.

22. The .animation() effect

We can animate elements by using CSS properties of the element. This is the idea: when you select an element, it will have certain values for CSS properties. In the .animate() method, you specify new CSS property values. When the .animate() method is called on the element the new values take into effect producing animation.
CSS properties with only numerical values work in .animate() method. For example, width, height, fontSize can be used but not background-color.

Syntax:
$('selector').animate(arg0,arg1,arg2,arg3);

Notice that the .animate() method takes four arguments. The first argument, arg0, called *properties* argument, specifies the "target" values of CSS properties. The other three arguments, arg1, arg2, arg3 are as before, "duration", "easing" and "complete".

The word "target" is used to mean the value at the end of the animation. The element starts with whatever CSS properties it currently has and they are changed to target values. Syntax for specifying "target" CSS properties: {property:'value',property:'value',......}

You can use any property name with numeric value. Remember CSS property names look slightly different when you use them in DOM/jQuery. This is how you convert an original CSS property name into DOM/jQuery name: If the CSS property name has one word, DOM's equivalent is the same. If the CSS property name has multiple words separated by hyphen ("-"), to form the DOM equivalent, remove the hyphen and capitalize the next word.

Important note:
If your animation changes the position of the element, the position property cannot be static, which is default. So when you load your HTML page, you must always specify the position property to be relative, absolute.

Example:
```
<!DOCTYPE html>
<html>
```

```
<head>
    <script src="jquery.js"></script>
    <title>Animation example</title>
    <style>
      img{
         position: relative; /* This is important */
      }
    </style>
</head>
<body>
    <h3> Click here <h3>
    <img id="lotus" src="lotus.jpg" alt="" width="100" height="100"><p>
    <script>
       $( "h3" ).click(function() {
        $( '#lotus' ).animate({left:'450px',height:'300px',width:'300px'},1000);
      });
    </script>
</body>
</html>
```

Study the code,

```
$( "h3" ).click(function() {
  $( '#lotus' ).animate({left:'450px',height:'300px',width:'300px'},1000);
});
```

When the <h3> element is clicked, the #lotus element is animated. The #lotus element is the image. The image is loaded on the left (default) with width and height 100x100. The target values are left=450px, height=300px and width=300px. That is, when you click <h3> element, the image moves to the right and the size becomes 300x300 at the end. Duration is 1000 milliseconds.

23. Specifying more than one effect

You can specify more than one effect with a selector. The syntax is separate the method calls by a period.

Example:

```
$(selector).effectOne().effectTwo()....;
```

24. The delay(*duration*) effect

The .delay() method is used to delay the execution by so milliseconds given by the argument, *duration*. The argument *duration* is just a number specifying milliseconds.

Example:

This is the same example as the last one, with a delay:

```
<script>
 $( "h3" ).click(function() {
  $( '#lotus' ).delay(2000).animate({left:'450px',height:'300px',width:'300px'},1000);
});
```

Notice the delay method with 2000 milliseconds delay. Animation starts after 2000 milliseconds after you click the <h3> element.

25. Other effects methods

There other effects method, which I have not described in my notes:

.clearQueue()

.data()

.dequeue()

.queue()

.finish()

.stop()

See http://api.jquery.com/category/effects/ for details.