IT614

JavaScript Events

1. What is event-driven programming?

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. In addition to user generated events, the browser itself can trigger events such as the page is loaded, etc.

JavaScripts are event-driven programs. JavaScript/jQuery provides special methods, which bind event handlers to events. Event binding refers to telling the browser that a particular function should be called whenever some 'event' occurs. A JavaScript programmer completes the code in event handlers telling the browser what do to when the event happens. In this chapter, we will discuss in detail the event handlers associated with several standard events.

2. A basic jQuery method text()

Before we get into events, let us see a basic jQuery method, **text().** This method is useful in retrieving the contents of an HTML element (using selector). It can also be used to change the contents.

Basic syntax:

$(*selector*).text()

This retrieves the text in the *selector*.

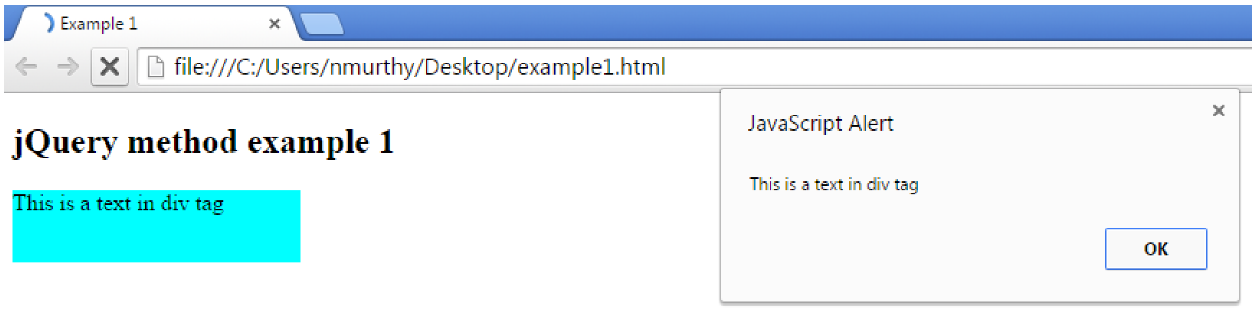
$(*selector*).text("*string*");

This places the argument text within the selector.

Example:

```
<!DOCTYPE html>
<html>
 <head> <script src="jquery.js"></script>
 <title>Example 1</title>
 <style>
 div{
   width: 100px;
   height: 100px;
   background-color:cyan;
}
</style>
</head>
<body>
<h2>jQuery method example 1</h2>
 <div>This is a text in div tag</div>
<script>
alert($('div').text());
$('h2').text("This is replaced text");
</script>
</body>
</html>
```

This HTML page first displays,



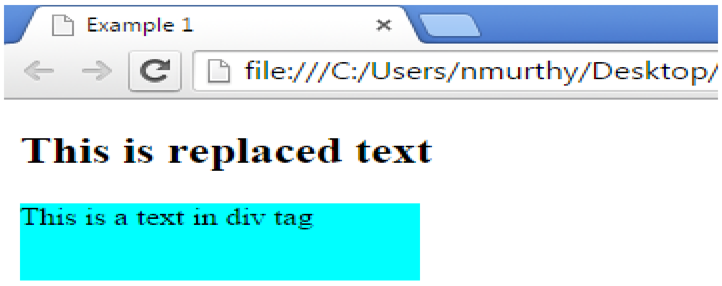
jQuery method example 1

This is a text in div tag

This is a text in div tag

The alert box is the result of **alert($('div').text());** This code retrieves text from the <div> tag and display it in the alert box.

When you click OK in the alert() window, the code **$('h2').text("This is replaced text");** is executed. This statement replaces the text in <h2></h2> by **This is replaced text**.



This is replaced text

This is a text in div tag

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example1.html

3. Objects, properties and methods

We will be using the term objects in this chapter. It is important to understand the basic concept of objects. Objects are used in several languages (Java, C++, JavaScript, JQuery and so on).

Think an object as an entity which contains two kind of items: properties and methods. A property of an object holds a value (could be a number, a string and so on), and a method is a function, which returns a value when called.

Let me give you a very simple example to make these ideas clearer.

Assume we have an object called **car1**. The **car1** object contains properties called **make**, **model**, **color**, **and price.** These properties **make, model, color and price** hold certain values. Assume these values are: make is Toyota, model is Camry, color is white, and price is 20,000. Let us also assume that the object **car1** contains a method (a method is a function) called **finalPrice**, which takes one argument for *tax*: **finalPrice(*tax*).** This methods returns the final price after adding price of the car + tax.

Accessing the values of properties of an object:
The syntax for accessing the value of a property is
objectName.propertyName.

Example:
To access the value of the property **make** of the object **car1** is **car1.make**, which gives Toyota.
Similarly, car1.model gives Camry, car1.color gives white and car1.price gives 20,000.

Accessing a method of an object:
The syntax for accessing a method is
objectName*.*methodName(argument)

When you call a method, you may have to provide a value for the argument. Some methods may not require arguments.

Example:
Let us call the method **finalPrice(*tax*)** of **car1** with ***tax*** value 6 (that is, 6%):
car1.finalPrice(6)
This will return 21200. (after adding 20000 + tax of 1200).

Note: You may have another object **car2** of the same kind as **car1**. This means **car2** will have the same properties and methods as in **car1**, but the actual values of properties may be different.

jQuery comes with several built-in objects. Whenever you like to use a jQuery object, you need to know (from references) the properties and methods of the object. It is impossible to learn all the properties and methods of all the jQuery objects. In the following, I introduce more useful properties and methods of jQuery objects we come across.

4. jQuery methods binding event handlers to events

The typical syntax for binding an event handler to an event is,

```
$(selector).event (function(eventObject){
  Body of the function
});
```

Where,

$(*selector*)	The jQuery object which generates the event.
event	Binds an event handler to the event
function	Callback function. Executed each time the event is triggered
eventObject	The event automatically provides an *eventObject* to the function as an argument. This object contains many properties and methods related to the event.

In the *body of the function* you write statement you want executed when the event happens. A function such as this is called a callback function. The argument to such functions is automatically provided.

Note: It is very important to follow the syntax for a callback function. Notice the '(', …'{',..'}', and ');'.

Example 2:

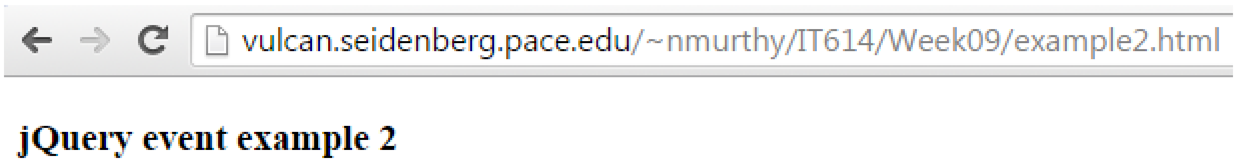
```
<!DOCTYPE html>
<html>
<body>
<h3>jQuery event example 2</h3>
<script>
  $('h3').click(function(eventObject){
    alert("Hello");
  });
</script>
</body>
</html>
```

Focus on the code,

```
  $('h3').click(function(eventObject){
    alert("Hello");
  });
```

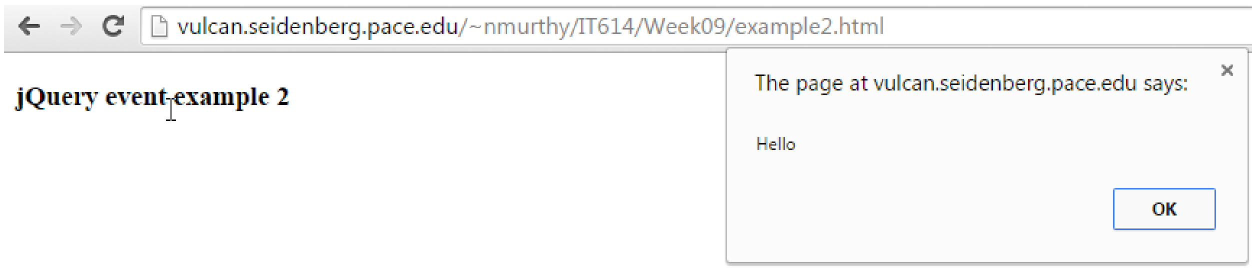
This code says, when the user clicks (this is the event) on the <h3> element, execute the function. The body of the function has only one line: **alert ("Hello");**

Here is output in a browser:


vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example2.html

jQuery event example 2

The string, jQuery event example 2is the <h3> element. When you click on this string the event is triggered and the body of the function. Click on the string to see:


vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example2.html

jQuery event example 2

The page at vulcan.seidenberg.pace.edu says:

Hello

5. More on event object

When an event occurs, the corresponding event handler is executed. An event object is created on the fly and it is made available to the handler function as an argument:

```
$(selector).event(function(eventObject){
  Body of the function
});
```

The argument ***eventObject*** is an object. This object is automatically provided by JavaScript when the event happens. We use this ***eventObjcet*** in the body of the method to access the properties and methods of this object.   What are the properties and methods available in the ***eventObject***? It depends on the event. For details, we need to see a manual. But, I will list important/useful properties and methods for the objects we come across in this course.

Note: In some examples we may not need this argument in the body of the function. In such a case we will omit it:

```
$(selector).event(function(){
  Body of the function
});
```

6. The click event

We will now explore the event of clicking mouse pointer on an element (selector). When this event happens, an ***eventObject*** is created by JavaScript. This ***eventObject*** has several properties and methods. We will consider only selected properties here:

➢ The pageX property

The **pageX** property holds the mouse position (at the position of clicking) relative to the left edge of the document.

➢ The pageY property

The **pageY** property holds the mouse position (at the position of clicking) relative to the top edge of the document.

Remember **pageX** and **pageY** are properties of the click event object, which is **eventObject** in the code. That means, to access the values of these properties, we use the syntax: **eventObject.pageX** and **eventObject.pageY.**

Example 3:

```
<!DOCTYPE html>
<html> <head> <script src="jquery.js"></script>
 <style>
 div{ width: 250px; height: 100px; background-color:cyan;}
 span{ width: 200px; height: 50px; background-color:yellow;}
</style>
</head>
<body >
 <div></div>
 <span></span>
<script>
$( 'div' ).click(function(eventObject) {
  $('span').text("X coordinate: "+eventObject.pageX+"  Y coordinate: "+eventObject.pageY);
});
</script></body></html>
```

Let me explain the code:

- We are using jQuery methods, so we need include **jquery.js**:
  <script src="jquery.js"></script>
  You must make jqury.js file available in the same directory as the HTML file.
- <div></div> is a 200px x100px rectangle in cyan color (specified in CSS)

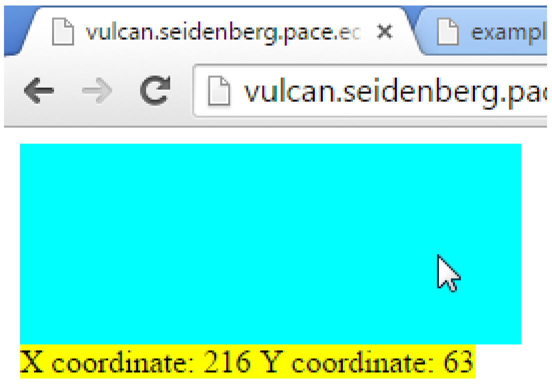
- <span></span> is a place holder for output.
- The crucial part is this:

```
$( 'div' ).click(function(eventObject) {
  $('span').text("X coordinate: "+eventObject.pageX+" Y coordinate:
"+eventObject.pageY);
});
```

You have to carefully follow the syntax (notice all the $ signs, (and { carefully).

This code says: "when the user **clicks** anywhere on '**div'** execute the body of the function. A profound concept here is, when this event happens, JavaScript automatically creates an event object and passes it the function (in the argument ***eventObject***). And we use ***eventObject*.pageX** and ***eventObject*.pageY** to obtain the position of the mouse pointer on the **<div>** element. These values are displayed in the <span> element using **text()** method discussed earlier.

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example3.html



X coordinate: 216 Y coordinate: 63

The rectangle is the <div> element. The value 216 is the distance of the mouse pointer from the left edge of the rectangle and 63 is the distance from the top edge.

When you try it, move the mouse pointer on the rectangle and click, and see how these values change.

- **The target, and currentTarget properties**

The **target** property refers to the DOM element that started the event. The **currentTarget** property refers to the element which is attached to the event.

Example:

Consider the HTML code:

```
<h3>jQuery event example</h3>
<div>Hello
<h3> This is h3</h3>
</div>
```

Here the <h3> element is contained within the <div> element. In other words, <div> is the parent of <h3>.

Now consider the event binding code:

```
$('div').click(function (eventObject) {
        ….
        ….
        });
```

Here the click event is attached to <div> element. The <h3> element is not attached to any event. But, because <div> is a parent of <h3>, when you click <h3>, it fires the parent element (this is called event bubbling). That is, when you click <h3>, because it is not attached to any event, its parent event will be fired.

Because click event is attached to <div> element, the **currentTarget** property is <div>. If you click the div element (not <H3>), then both **target** and **currentTarget** are <div>, whereas if you click <h3> element, target is <h3> and currentTarget is still <div>.

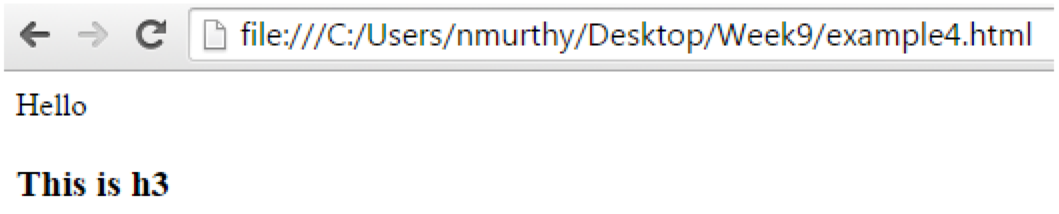
Example 4

```
<!DOCTYPE html>
<html>
 <head> <script src="jquery.js"></script>
 <title>Example 4</title>
</head>
<body>
<div>Hello
<h3> This is h3</h3>
</div>
```

```
<script>
$('div').click(function (eventObject) {
     $('span').text("target: "+eventObject.target.nodeName+" current target:
"+eventObject.currentTarget.nodeName);
        });
</script>
<span></span>
</body>
</html>
```

We will use <span></span> to place the output.

Browser output:



file:///C:/Users/nmurthy/Desktop/Week9/example4.html

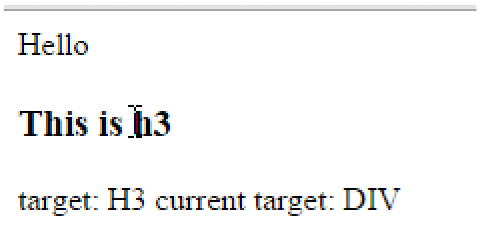
Hello

This is h3

3

Here **Hello** is <div> element and **This is h3** is <h3> element (within <div>).

When you click <h3> element,

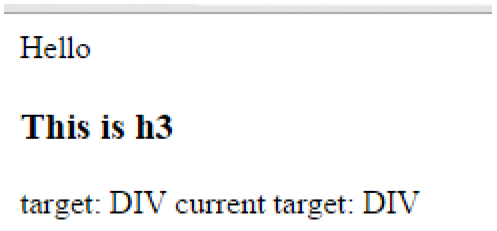


Hello

This is h3

target: H3 current target: DIV

When you click <div> element,



Hello

This is h3

target: DIV current target: DIV

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example4.html

➢ **The keyword this**

The word **this** is a keyword. It always represents currentTarget.

In the previous example, you can replace **eventObject.currentTarget** by **this**.

- **The data property**

If you like to pass additional values to the function, you can do so by using the following syntax. Values are passed using name and value pair. The values passed can be retrieved in the function using the **data** property.

$(*selector*).*event***({***name1:"value1", name2:"value2"},***function(***eventObject***) {**

Body of the function

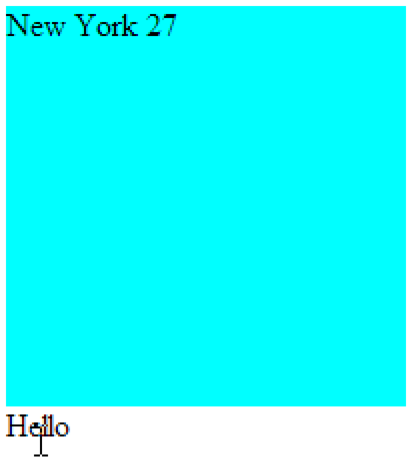
});

The values can be retrieved in the function using the syntax **data**.*name*.

Example 5:

```
<!DOCTYPE html>
<html> <head> <script src="jquery.js"></script>
 <style>
 div{
    width: 200px;
    height: 200px;
    background-color:cyan;
}
</style>
 </head>
<body >
 <div></div>
 <span>Hello</span>
<script>
$( 'span' ).click({name:"New York", age:"27"},function(event) {
  $('div').text(event.data.name+" "+event.data.age);
});
</script></body></html>
```

Output:



When you click Hello (which is <span> element), you will see New York 27 in the <div> element.

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example5.html

➢ **The preventDefault() method**

If **preventDefalult()** method is called, the default action of the event will not be triggered. We will an example of use of it later.

7. Browser and document events

➢ The **resize()** method

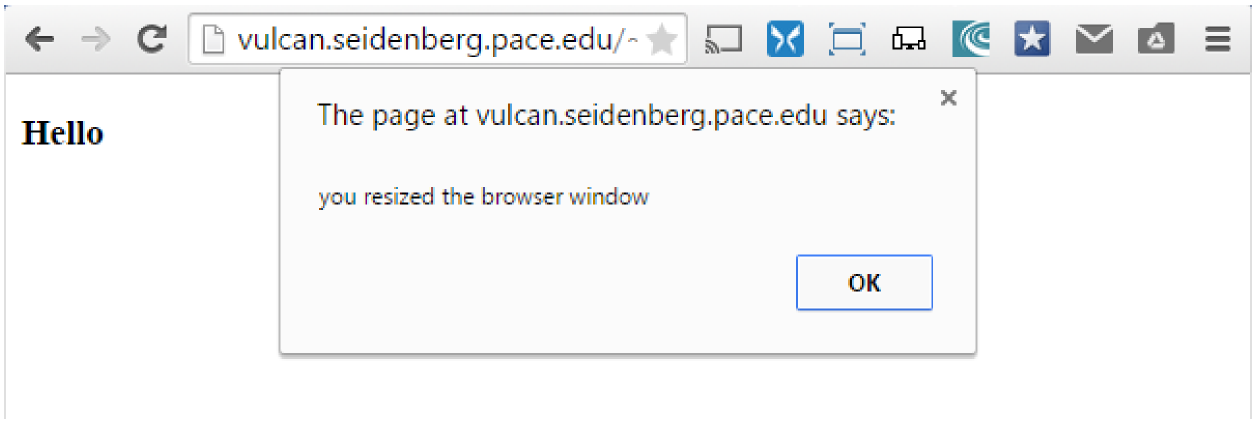
Event: Size of the browser window (or an element $(selector)) is changed.

The **resize()** method binds an event handler to this event.

Example 6:

```
<!DOCTYPE html>
<html> <head> <script src="jquery.js"></script></head>
<body >
 <h3>Hello</h3>
<script>
$( window ).resize(function() {
  alert("you resized the browser window");
});
</script></body></html>
```

When this page is being displayed on a browser, if you change the size of the browser window, you will an alert dialog box:


vulcan.seidenberg.pace.edu/~

Hello

The page at vulcan.seidenberg.pace.edu says:

you resized the browser window

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example6.html

---

- **The onload event**

Event: The HTML code is completely loaded by the browser.

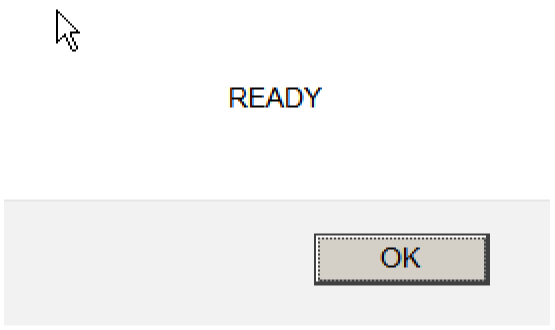
Normally you write a function you want executed when this event happens. onload() is generally used in the <body> tag.

Example:

Here is a simple example:

```
<body onload=alert("READY")>
….
….
….
</body>
```

When the HTML page is completely loaded, you will see:


READY

Note:

You can also use **onload** with some other tags (such as <img>).

Note:

This example shows another way of using **onload**:

```
<script>
window.onload = function() {
   alert( "READY" );
}
</script>
```

➢ **The .ready() method**

This is a very important method.

Event: Browser finished loading HTML.

The **ready()** method binds a handler function to this event.

When we include jQuery in our documents, it is important to remember that jQuery has to wait for all the HTML code to be loaded by the browser before it starts working on elements. That is why we have always included jQuery statements after all the HTML code is loaded. That is, we have included all our jQuery statements just before </body>.

The use of **ready()** method is another way of making sure that HTML code is loaded by the browser before jQuery starts working.

Syntax for using **ready()** method:

```
$(document).ready(function() {
The entire jQuery code goes here.
});
```

Note:

The **onload** waits for everything loaded, including all images and even banner ads. But, **ready()** is better in the sense that, it runs the code in the function as soon as the document is ready to be manipulated. So you always use **ready()** to start jQuery.

Alternative to using alert() dialog boxes

We regularly use **alert()** dialog boxes to output some results. There is another of displaying results without using alert boxes (using alert boxes is fine. This is just a different way of doing it).

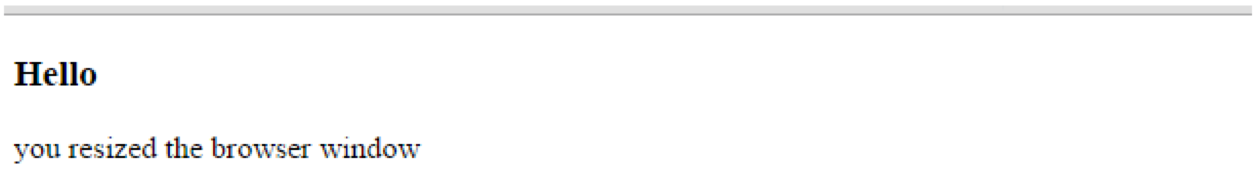
First create an empty <div></div>.
Within the event handler function, include the following statement,
$('div').text("string to be displayed");
When this statement is executed, the string to be displayed is shown between the <div></div> tags.

Example:

We will do the same example again, but use idea instead of an alert box:

```
<!DOCTYPE html>
<html>
 <head> <script src="jquery.js"></script></head>
<body >
 <h3>Hello</h3>
 <div></div>
<script>
$( window ).resize(function() {
  $('div').text("you resized the browser window");
});
</script></body></html>
```

When you resize the window, you will see:



Hello

you resized the browser window

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example7.html

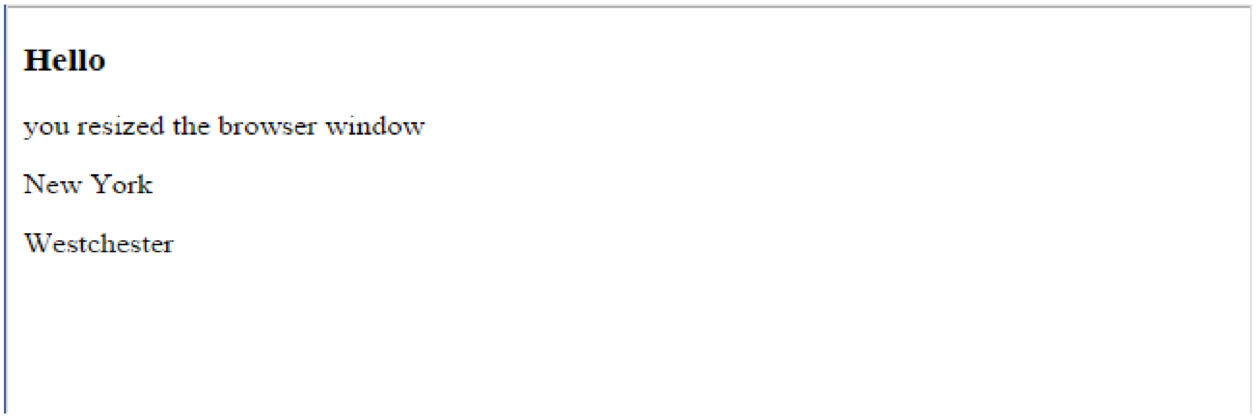
➢ Adding multiple lines of output

When you add a second line to the <div> area, the first line is replaced by the second line. If you want to keep the first line and add another line, use **append**() instead of **text**(). Also, you can append HTML tags (such as <P>).

Example 8:

```
<!DOCTYPE html>
<html> <head> <script src="jquery.js"></script></head>
<body >
 <h3>Hello</h3>
 <div></div>
<script>
$( window ).resize(function() {
  $('div').text("you resized the browser window");
  $('div').append("<P>New York");
  $('div').append("<p>");
  $('div').append("Westchester");
});
</script></body></html>
```

When you minimize the window, you will see,



```
Hello

you resized the browser window

New York

Westchester
```

Try It:

http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example8.html

➢ Making the output fadeout

Here is a mechanism to make the output fancy. Here we make the text fade out (by using the **fadeOut()** method)

When you use the JavaScript **fadeout()** method, you can provide argument (just a number), which specifies after specified milliseconds fade out occurs. If you don't include any argument, default is 400 milliseconds.

Example 9:

```
<!DOCTYPE html>
<html> <head> <script src="jquery.js"></script></head>
<body >
 <h3>Hello</h3>
 <div></div>
<script>
$( window ).resize(function() {
  $('div').text("you resized the browser window");
  $('div').append("<P>New York");
  $('div').append("<p>");
  $('div').append("Westchester");
  $('div').fadeOut(1000);
});
</script></body></html>
```

In this example, the three lines of text is first visible when you resized the browser window. After 1000 milliseconds, the text vanishes.

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example9.html

Other arguments you can use with fadeOut() method:

See other arguments (interesting): http://api.jquery.com/fadeout/

8. Mouse pointer events

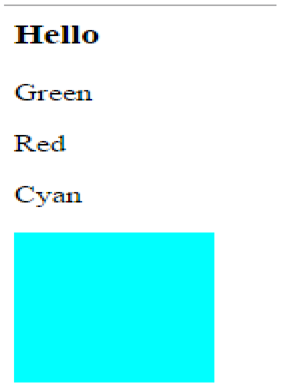
There are several events related to mouse pointer activities. We have seen one important mouse event: **click**. Let us do one more example with click event. In this example, I have used '**this'** keyword and see how CSS is also used.

Example 10

In this example there are three <P> elements. The keyword **this** refers to the <P> element which was clicked. (read this again. That is the power of **this** keyword).

```
<html> <head> <script src="jquery.js"></script>
 <style>
 div{
 background-color:gray;
 width:100px;
 height:100px;
 }
 </style> </head>
<body >
 <h3>Hello</h3>
 <p>Green
 <p>Red
 <p>Cyan
 <div> </div>
<script>
$( 'p' ).click(function() {
  $('div').css("background-color",(this).innerHTML);
});
</script></body></html>
```

Browser output:



```
Hello

Green

Red

Cyan
```

Click different <P> elements and see the output.

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example10.html

9. Other jQuery methods binding mouse activities events:

The **.dblclick()** method

The **.hover()** method

The **.mousedown()** method

The **.mouseup()** method

The **.mouseenter()** method

The **.mouseleave()** method

The **.mousemove()** method

The **.mouseout()** method

The **.mouseover()** method

The name of the methods indicate the event. Experiment with these by replacing **click** in the previous example by these names.

As you can imagine, the differences between some of these is subtle. Read the manual for details: http://api.jquery.com/category/events/mouse-events/

10. Form events

We have discussed form elements in an earlier chapter. We will now see events related to form elements and jQuery methods binding event handlers to these events.

Before we start form event methods, let us look at one method to retrieve values from form elements, **val().**

➢ **The val() method**

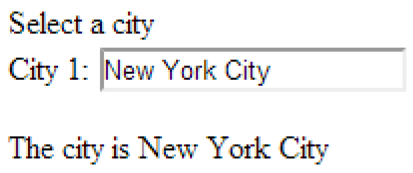
The **val()** method is primarily used to get the values of form elements such as <input>, <select> and <textarea>. In the case of <select multiple="multiple"> elements, the **val()** method returns an array containing each selected option; if no option is selected, it returns **null**.

Example 11:

```
<!doctype html>
<html>
<head> <script src="jquery.js"></script><title>form val() example </title></head>
<body>
```

```
<form>
City 1: <input type=text value="New York City" size=20><p>
</form>
<div></div>
<script>
var inp = $('input');
document.write("The city is ",inp.val());
</script>
</body>
</html>
```

Output:



Select a city

City 1: New York City

The city is New York City

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example11.html

---

➢ **The .focus() method**

Event: The mouse pointer is taken on an element and mouse button is clicked. That is, the element gains focus.

The **focus()** method binds an event handler to this event.

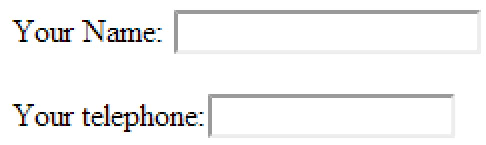
Example 12:

To keep the example simple, I have not included any attributes in <form> and I have not included a submit button.

```
<!doctype html>
<html>
 <head>
 <script src="jquery.js"></script>
 <title>form event example </title>
 </head>
<body >
<form>
Your Name: <input id="nam" type=text name="yourName" size=20><p>
Your telephone:<input id="tel" type=text name="telephone" size=15>
</form>
```

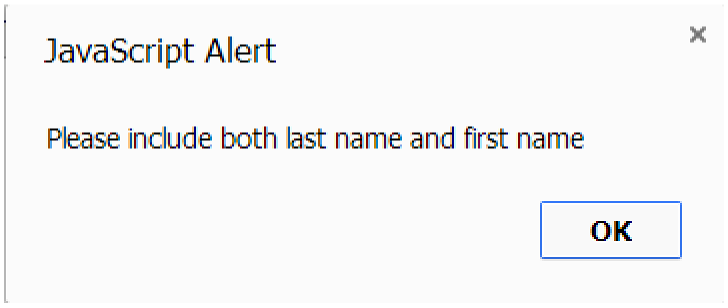
```
<script>
$( "#nam" ).focus(function() {
 alert("Please include both last name and first name");
});
</script>
</body>
</html>
```

Output:


Your Name:

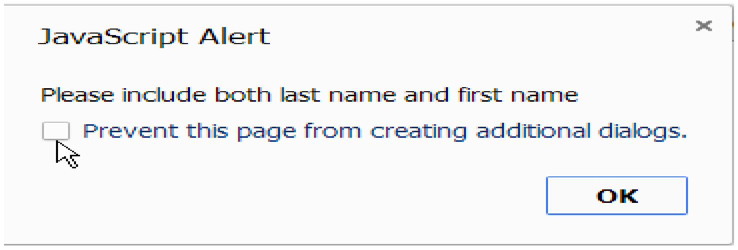
Your telephone:

When you click the name box to enter your name, you will see the alert dialog box:


JavaScript Alert

Please include both last name and first name

Unfortunately, every time you try to type your name, the dialog box appears (this time with "prevent this page…" check box):


JavaScript Alert

Please include both last name and first name

Prevent this page from creating additional dialogs.

Check the small check box to prevent the browser displaying the alert dialog box again.

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example12.html

---

➢ **The blur() method**

Event: The mouse pointer is taken away from a form element.

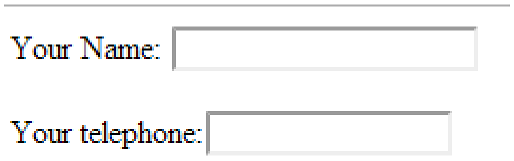
The **.blur()** method binds an event handler to this event.

Example:

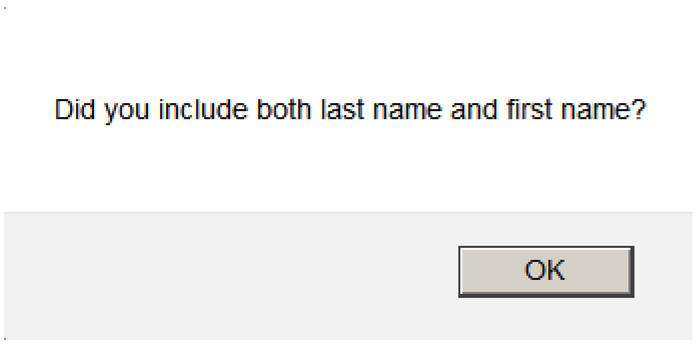
HTML document is as before.

```
<script>
$( "#nam" ).blur(function() {
  alert("Did you include both last name and first name?");
});
</script>
```

Output:



When you type the name in the first box, and take the mouse pointer out and click somewhere else, this box loses focus – triggers the blur event handler. The event handler function displays an alert dialog box:


Did you include both last name and first name?

➢ **The .change() method**

Event: The value in an element is changed. Applicable to <input>, <textarea> and <select> elements only.

The **.change()** method binds an event handler to this event.

When the input box is empty and you enter a value, it is considered change.

Example:

HTML document is as before.

```
<script>
$( "#nam" ).change(function() {
  alert("Did you include both last name and first name?");
});
</script>
```

➢ The select() method

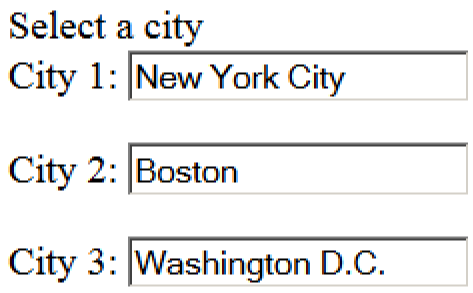
Event: The value in an element is changed. Applicable to <input type="text">, and <textarea> elements only.

The **.select()** method binds an event handler to this event.

Example 13:

```
<!doctype html>
<html>
 <head>
 <script src="jquery.js"></script>
 <title>form event example </title>
 </head>
<body>
Select a city
<form>
City 1: <input type=text value="New York City" size=20><p>
City 2: <input type=text value="Boston" size=20><p>
City 3: <input type=text value="Washington D.C." size=20><p>
</form>
<script>
$( "input" ).select(function() {
  alert("You selected "+$(this).val());
});
</script>
</body>
</html>
```

Output:

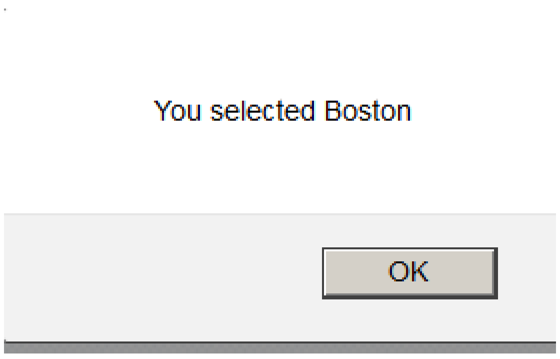

Select a city

City 1: New York City

City 2: Boston

City 3: Washington D.C.

When you select (highlight) a city, you will see,


You selected Boston

Note: Study the code in **alert()** dialog box.

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example13.html

---

- **The submit() method**

Event: The submit button in a form is clicked.

The **submit()** method binds an event handler to this event.

This is a very important step in processing a form. We will spend more time discussing this.

In practice the event handler is bound to the form.

In the event handler function use JavaScript/jQuery to verify all the items. If you find something wrong in the data, you can alert the user and prevent the data from going to the server program.

This is how you cancel the submission:

In the event handler callback function, include a parameter, *eventObject*. The event automatically assigns this parameter an event object, which contains many properties and methods. We will not use them much. But one method we now use is preventDefault(). If we decide to cancel form data submission, we call this function within the callback function, which prevents from form data being submitted to the server.

Example:

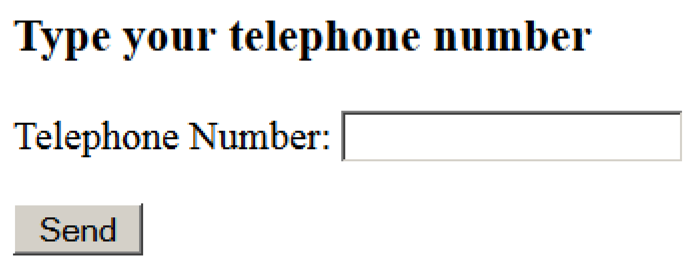
In this example, we will have a form and assign this URL to **action** attribute: http://vulcan.seidenberg.pace.edu/~nmurthy/cgi-bin/IT614/Example6.pl. The purpose of this is just to experiment with submission and cancellation. The program Example6.pl is a small program on the server which just acknowledges the data sent.

In the callback function we use a confirm dialog box. As you know a confirm dialog box displays a dialog box with two option buttons: OK and Cancel. If you click OK, the dialog box returns true and if you click cancel, it returns false. If you click Cancel, the **preventDefault()** method is called. Study the code well.

Example 14.html

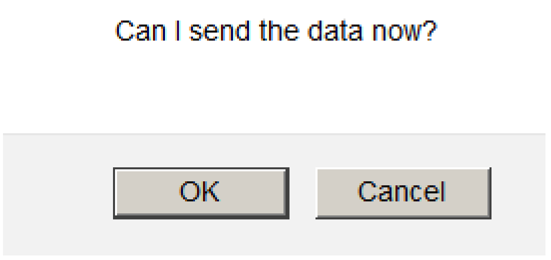
```
<!doctype html>
<html>
 <head>
 <script src="jquery.js"></script>
 <title>submit example </title>
 </head>
<body>
<h3>Type your telephone number</h3>
<form action="http://vulcan.seidenberg.pace.edu/~nmurthy/cgi-bin/IT614/Example6.pl">
Telephone Number: <input type="text name="tel"><p>
<input type="submit" Value="Send">
</form>
<script>
$("form").submit(function(event) {
  var ans = confirm("Can I send the data now");
  if( !ans){
      alert("Data not sent");
      event.preventDefault();
  }
});
</script></body>
</html>
```

Output:


Type your telephone number

Telephone Number:

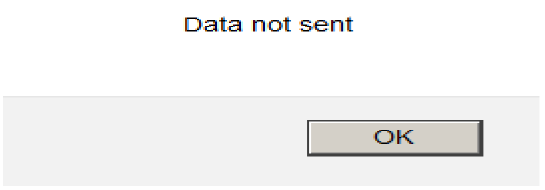
Type something and click Send button. You will see,


Can I send the data now?

If you click OK, the data will be sent to the server. The server side program sends a reply:

Thank you . I have noted your telephone number.

If you click Cancel, the data will not be sent to the server. You will see,


Data not sent

You will see the original screen.

Try It:

http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example14.html

---

➢ Processing form data

As you know, HTML forms are one of the standard ways to obtain input from the user. We have seen forms with text boxes, text area, radio buttons, check boxes, and so on.

The usual HTML tag <form> is used to create forms. A form is a JavaScript object.

The following are some of the form attributes: name, id, method and action.

The value of the attribute **name** is mainly used by the program on a server, which processes the data. On the client side (that is, on the browser side), it is not much useful. The value of the attribute **method** can be "get" or "post". We have already discussed the difference between these two.

The value of the attribute **action** is the URL to a program on a server. This program receives the data from the form and processes it. You will learn server side programming in a different course.

In our examples below, we will not use the attributes **name**, **id** and **method**. For our examples to work, we need to use the action attribute with some relevant value. So for action value in our examples we will just use a JavaScript alert box:
action="javascript:alert('Done!')".

As you know, the form data will be sent only when the submit button in the form is pressed.
As you know, the following are some of the elements you can have within a <form> tag:

```
 <input type="checkbox">
 <input type="file">
 <input type="hidden">
 <input type="image">
 <input type="password">
 <input type="radio">
 <input type="reset">
 <input type="submit">
 <input type="text">
 <select> <option>…</option> </select>
 <textarea></textarea>
```

We will discuss some of these below.

- **Using text(), val() .html() methods and innerHTML property**

Let me explain the differences and how/when to use these. All the four values are used to retrieve and placing values in elements. Standard terminology used for retrieving and placing values is "getting" and "setting".

All the four are used to both get and set values.

.text() gets the combined text contents of each element in the set of matched elements, including their descendants. You cannot use it to get the text value of **input** or **textarea** elements.

.val() is used to get the value of form elements **input**, **select** and **textarea**. It gets the value from the first element in the set of matched elements.

.html() gets the HTML contents of the first element in the set of matched elements. This method uses the browser's **innerHTML** property.

innerHTML gets the text of all the matching elements. **innerHTML** works on elements NOT on jQuery objects.

Example:
Consider the HTML code,
<span>Hello<b> New York </b></span>

Statement	**Value**
$('span').text()	Hello New York
$('span').val()	val() cannot be used with $('span'). Use val() with form elements.
$('span').html()	Hello<b> New York </b>
$('span')[0].innerHTML	Hello **New York**. You cannot use innerHTML on $('span'). Must be an element, not a jQuery object.

.text(), .html(), .val() and **innerHTML** can be used to set values as shown by the following examples:
Consider,
<span>Hello<b> New York </b></span>

Example:
$('span').text("<h6><i>Westchester</i></h6>");
Replaces (on the browser screen), Hello **New York** by <h6><i>Westchester</i></h6>
Example:
$('span').val("<h6><i>Westchester</i></h6>");
Does not work. .val() can only be used with form elements.

Example:

$('span').html("<h6><i>Westchester</i></h6>");

Replaces (on the browser screen), Hello **New York** by *Westchester*

Example:

$('span')[0].innerHTML="<h6><i>Westchester</i></h6>";

Replaces (on the browser screen), Hello **New York** by *Westchester*

➢ **Processing text box data**

Text boxes are created using **input** tag with **type**="**text**".

Syntax:

<**input type**="**text**" **name**="*textboxName*" **size**="*width*" **value**="*some string*">

attribute	value	comments
type	**text**	create a text box
name	a user created name	associate the name with the box
size	a user specified number	width of the text box - in number of characters
value	a user created string	the string will appear in the box

Note:

The attribute size is optional. The default size is 20 characters long.

The attribute value is optional. The default is that the box will be empty.

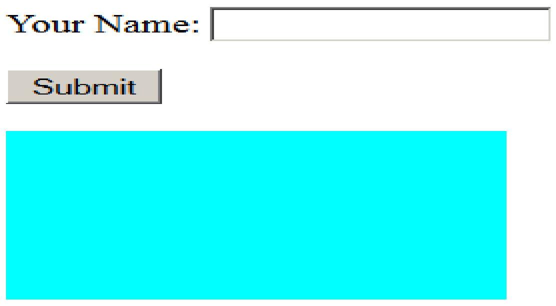
Example:

First we will see a template for button pressed callback function. In our examples, we will use a blank <div></div> and place output in this area.

```
<!doctype html>
<html> <head> <script src="jquery.js"></script> <title>Processing text boxes </title>
 <style>
 div{
   width:200px;
   height:100px;
   background-color:cyan;
   }
  </style>
 </head>
<body>
<form action="javascript:alert( 'Done!')">
Your Name: <input type="text" name="yourName" size=20><p>
<input type="submit" value="Submit">
</form>
<div></div>
<script>
$( "form" ).submit(function() {
….
….
….
});
</script>
</body>
</html>
```

Notice, the <div></div> is styled to be rectangle in color. We will place output here. We will see what we can do within the callback function body. This function will be executed when the user presses the button.

Here is a browser output:



➢ **Various ideas related to processing input data within the callback function:**

var inputs = $('input');

The variable inputs holds all the <input> elements. The variable inputs is used to access different <input> elements using array notation. inputs[0] refers to the first <input> element, inputs[1] refers to the second <input> element, and so on.

var inputs = $('input : text');

The variable inputs holds all the <input type="text"> elements. The variable inputs is used to access different <input type="text"> elements using array notation. inputs[0] refers to the first <input> element, inputs[1] refers to the second <input> element, and so on.

Note:

$('input') picks up all <input> elements; $('input : text') picks up only <input> elements with type="text".

length and size()

You can either use length property or size() method to retrieve the number of elements in an object.

Example:

var inputs = $('input');

inputs.length and inputs.size() both return the number of <input> elements.

The val() method

As seen before, the **val()** method is used to retrieve the value in a text box.

Example:

var inputs = $('input');

$(inputs[0]).val() returns the text in the first input box.

The val() method can also be used to place a value in a form element.

Example:

$(inputs[0]).val("Thank you");

The string "Thank you" will be placed in the textbox.

- **Checking for a textbox for empty**

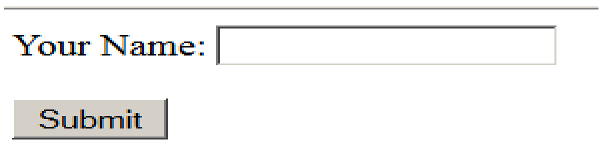
The user might not have input data in a textbox. We need to flag it. We have done it using proper attributes in HTML5 input tag. We will see how we can do it using JavaScript/jQuery. In fact, by doing this way, we can customize the error message to the user.

Checking for empty text boxes is easy, just compare the value retrieved from the box to an empty string, "".

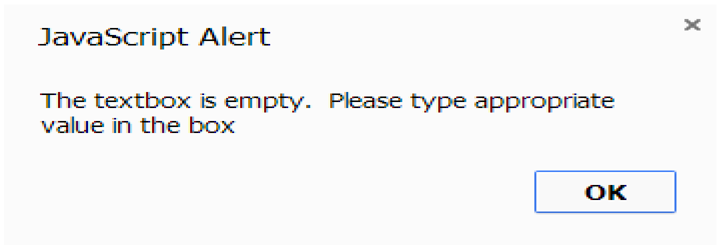
Example 15:

```
<!doctype html>
<html>
 <head>
<script src="jquery.js"></script>
 <title>Processing text boxes </title>
</head>
<body>
<form action="javascript:alert( 'Done!')">
Your Name: <input type="text" name="yourName"  size=20><p>
<input type="submit" value="Submit">
</form>
<script>
$( "form" ).submit(function() {
var inputs=$('input');
if($(inputs[0]).val() == ""){
  alert("The textbox is empty.  Please type appropriate value in the box");
}
});
</script>
</body>
</html>
```

Output:



If you click the Submit button without typing any value in the box, you will see


JavaScript Alert

The textbox is empty. Please type appropriate value in the box

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example15.html

➢ **Using regular expressions to validate data in the input textbox**

We have done it using pattern attribute in HTML5 input tag. We will see how we can do it using JavaScript/jQuery. In fact, by doing this way, we can customize the error message to the user.

First define the regular expression you like to use and assign it to a variable using the syntax,
var regex = /*regularExpression*/;

Next compare the value from the input textbox to the regular expression, typically in an **if** statement using the following syntax:

if(!*regex*.**test**($(*selector*).**val**())){
    alert("*Input invalid*");
}

Example 16:

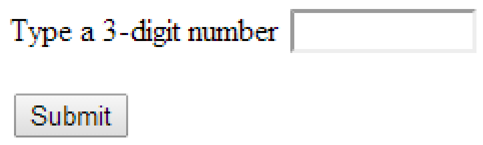
Let us use the following HTML code,

```
<form action="javascript:alert( 'Done!')">
Type a 3-digit number <input type="text" name="number"  size=10><p>
<input type="submit" value="Submit">
</form>
```

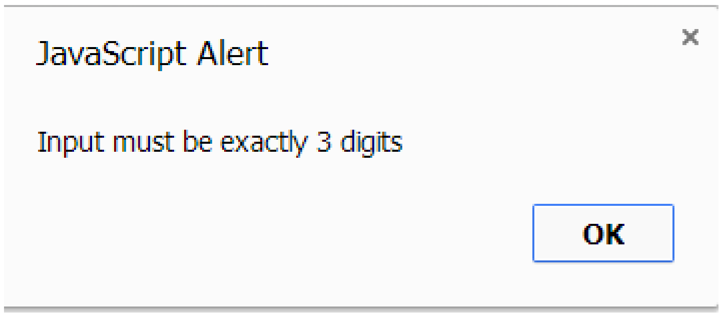
Let us check for the input to be exactly 3 digits.
A regular expression for a 3-digit number: ^[0-9][0-9][0-9]$
Now the JavaScript/jQuery code:

```
<script>
$( "form" ).submit(function() {
    var inputs=$('input');
    var regex = /^[0-9][0-9][0-9]$/;
    if(!regex.test($(inputs[0]).val())){
      alert("Input must be exactly 3 digits");
}
});
</script>
```

Output:



If you input anything other than a 3-digit number, you will see


JavaScript Alert

Input must be exactly 3 digits

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example16.html

---

➢ **Mouse movement events with input text boxes**

In addition to submit button event, we can use mouse movement events with input text boxes.

As we have seen before, here are the jQuery methods related to mouse movement events.

The click() method

The .dblclick() method

The .hover() method

The .mousedown() method

The .mouseup() method

The .mouseenter() method

The .mouseleave() method

The .mousemove() method

The .mouseout() method

The .mouseover() method

Let us take another example for the use of **click()** method.

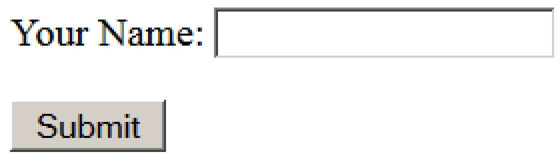
Event: The mouse pointer is over the textbox, and the mouse button is pressed and released.

The **click()**method binds an event handler to this event.

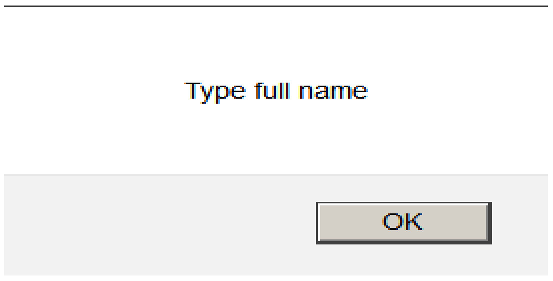
Example 17:

```
<!doctype html>
<html>
 <head>
<script src="jquery.js"></script> <title>Processing text boxes </title>
</head>
<body>
<form action="javascript:alert( 'Done!')">
Your Name: <input type="text" id="one" name="yourName"  size=20><p>
<input type="submit" value="Submit">
</form>
<script>
$('#one').click(function() {
  alert("Type full name");
});
</script>
</body></html>
```

Output:



When click the textbox with mouse pointer, you will see an alert box:


Type full name

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example17.html

Similarly we can use other mouse movement methods. Experiment with them.

11. Processing checkbox data

We will first review checkbox HTML. Check boxes enable users to check their choices among several check bozes. A check box is created using the <input> tag within a <form> tag.

syntax:

```
<form>
<input type="checkbox" name="cboxname1" value="value1" checked> text1 <p>
<input type="checkbox" name="cboxname2" value="value2" > text2 <p>
...
...
...
</form>
```

Each input will display a small checkbox along with the specified text. Users can click their choices.

The attribute **checked** is optional. If it is included, the checkbox is displayed checked.

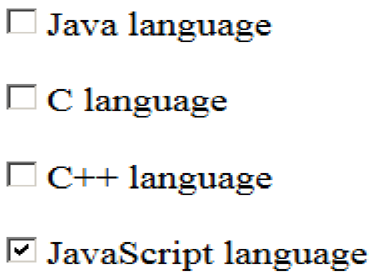
attribute	**value**	**description**
type	checkbox	creates a checkbox
name	a programmer specified name	Associates the name with the checkbox. This value will be used by the server side program. You can skip this.
value	a programmer specified string	Associates the string value with the checkbox.
checked	none	This is optional. if included the corresponding checkbox will be displayed checked

Example:

```
<html><head><title>checkbox example</title></head><body>
<form>
   <input type="checkbox" value="Java">Java language<p>
   <input type="checkbox" value="C">C language<p>
   <input type="checkbox" value="C++">C++ language<p>
   <input type="checkbox" value="JavaScript" checked>Javascript language<p>
```

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

Display on browser:



Processing a checkbox

Each checkbox is a JavaScript object. The set of checkboxes forms an array and the array is obtained using,
var checkedInputs = $('input');

Once we have the array in checkedInputs, we can access each checkbox by the usual subscript notation checkedInputs[0], checkedInputs [1], checkedInputs[2], and so on.

The jQuery **prop('checked')** method is used find out a particular checkbox was checked or not. We use the prop('checked') method in the following way:

checkedInputs[0].**prop('checked'),** checkedInputs[0].**prop('checked'),** checkedInputs[0].**prop('checked'),** and so on.

The prop('checked') method returns a Boolean – true or false. If returns true if the checkbox is checked, otherwise it returns false.

We use **val()** method to get the value of the checkbox. This is the value of the attribute **value** in the checkbox HTML code. For example, checkedInputs[0].val() would return Java.

We use the usual **submit()** method and the callback function event handler to process checkboxes.

Example 18:

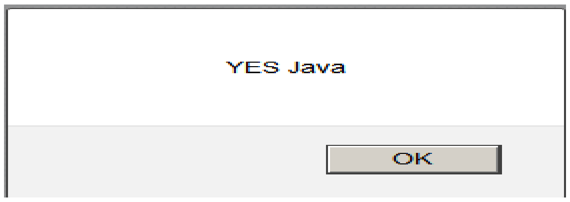
Consider the same HTML form, with a submit button added:

```
<body>
<form>
<input type="checkbox" value="Java">Java language<p>
<input type="checkbox" value="C">C language<p>
<input type="checkbox" value="C++">C++ language<p>
<input type="checkbox" value="JavaScript" checked>Javascript language<p>
<input type="submit">
</form>
</body>
```

Now the callback function:

```
<script>
$("form").submit(function(){
var checkedInputs = $('input');
    if($(checkedInputs[0]).prop('checked')){
      alert("YES "+$(checkedInputs[0]).val());
    }
});
</script>
```

If you check Java checkbox and click submit button, you will see the alert dialog box,


YES Java

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example18.html

---

Processing multiple checked boxes

As you know, when you use checkbox HTML code, you can check multiple checkboxes. Let us see how to collect all the items checked by the user.

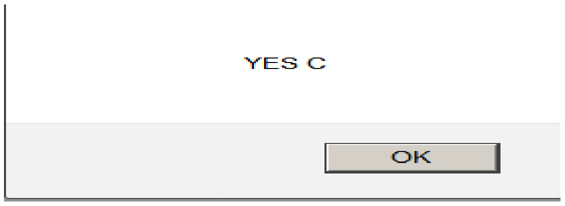
As mentioned, the checkedInputs variable in the function works like an array of <input> elements. We need to check each of these for checked condition. Obviously do it using a loop.

Let us add a loop in the function and do other necessary modifications:

Example 19:

```
<script>
$("form").submit(function(){
var checkedInputs = $('input');
for(i=0;i<checkedInputs.length;i++){
    if($(checkedInputs[i]).prop('checked')){
      alert("YES "+$(checkedInputs[i]).val());
    }
}
});
</script>
```

If you check several checkboxes and click submit button, you will see an alert dialog box like this for each checkbox checked:


YES C

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example19.html

Check several checkboxes and try.

➢ **jQuery $.each() loop**

In addition to the usual JavaScript loops, jQuery provides another loop: **$.each().** The first argument to **$.each()** is an array and the second argument is a callback function.

Syntax for using $.each()

$.**each**(*arrayElements*, **function**(*index,value*){

…….

});

In the body of the function the values of index and value range from 0 through all the elements in the array. That is, in the body of the loop, **index** is the subscript and value is **val.** But you don't use the notation arrayElements[index].

Example:

```
<script>
    myArray = [20, 10, 30,80,50];
    $.each(myArray,function(index,val){
    alert("index = "+index+" value = "+val);
    });
</script>
```

Output:

```
index = 0 value = 20
index = 1 value = 10
index = 2 value = 30
index = 3 value = 80
index = 4 value = 50
```

Note:
The array need not be an array of numbers. It can be an array of elements. In this case the second argument **val** is an element.

Example 20:
Let us process multiple selections in checkboxes, using $.each() loop. Here is a complete example.

```
<!doctype html>
<html>
 <head>
 <meta charset="utf-8">
<script src="jquery.js"></script>
 <title>Processing checkboxes</title>
 <style>
 div{
    width:200px;
    height:150px;
    background-color:cyan;
    }
    </style>
</head>
<body>
<form action="javascript:()">
<input type="checkbox" value="Java" >java language<p>
<input type="checkbox" value="C" checked>c language<p>
<input type="checkbox" value="C++">c++ language<p>
<input type="checkbox" value="Javascript">javascript language<p>
<input type="submit" value="Send">
</form>
<div></div>
<script>
$("form").submit(function(){
var checkedInputs = $('input');
$('div').append("You have chosen...");
$('div').append("<p>");
$.each( checkedInputs, function(i,item) {
    if($(checkedInputs[i]).prop('checked')){
      $('div').append($(checkedInputs[i]).val());
      $('div').append("<br>");
     }
  });
});
</script>
</body>
</html>
```

Try It:
http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example20.html

Important note:
jQuery provides a much better selector: 'input:checked'. This automatically picks up all the elements which has been checked. This means you don't even need to check if the button is checked. Just display values of the elements selected. Here is a modified code:

Example 21:

```
<!doctype html>
<html>
 <head>
 <meta charset="utf-8">
<script src="jquery.js"></script>
 <title>Processing checkboxes</title>
 <style>
 div{
    width:200px;
    height:150px;
    background-color:cyan;
    }
    </style>
</head>
<body>
<form action="javascript:()">
<input type="checkbox" value="Java" >java language<p>
<input type="checkbox" value="C" checked>c language<p>
<input type="checkbox" value="C++">c++ language<p>
<input type="checkbox" value="Javascript">javascript language<p>
<input type="submit" value="Send">
</form>
<div></div>
<script>
$("form").submit(function(){
    var checkedInputs = $('input:checked');
    $('div').append("You have chosen...");
    $('div').append("<p>");
```

```
  for(i=0;i<checkedInputs.length;i++){
    $('div').append($(checkedInputs[i]).val());
    $('div').append("<p>");
   }
});
</script>
</body>
</html>
```

Try It: http://vulcan.seidenberg.pace.edu/~nmurthy/IT614/Week09/example21.html

Note:
In addition to submit button event, we can use mouse movement events with input checkboxes.

12. Processing radio buttons

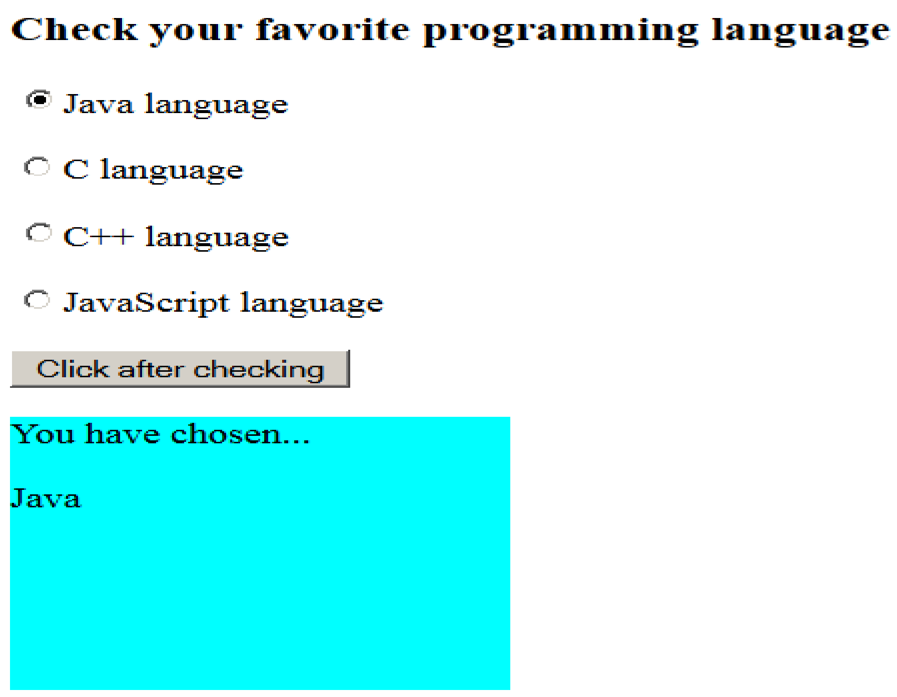
We will first review radio button HTML. Radio buttons are similar to check boxes, but the choices made here are mutually exclusive, within a group. The reader can make only one of many given choices. Unlike a group of check boxes, a group of radio buttons will have the same name.
Here is an example of radio buttons HTML:

```
<form action="javascript:()">
<input type="radio" name="languages" value="java">java language<p>
<input type="radio" name="languages" value="c" checked>c language<p>
<input type="radio" name="languages" value="c++">c++ language<p>
<input type="radio" name="languages" value="javascript">javascript language<p>
<input type="submit" value="click after checking">
</form>
```

As in the case of checkboxes, we use the jQuery **prop('checked')** method is used find out the radio button the user checked.
Example:
This example is similar to the previous example, except that in the loop we get only one answer – not multiple.

```
<body> <h3>check your favorite programming language</h3>
<form action="javascript:()">
<input type="radio" name="languages" value="java">java language<p>
<input type="radio" name="languages" value="c">c language<p>
<input type="radio" name="languages" value="c++">c++ language<p>
<input type="radio" name="languages" value="javascript">javascript language<p>
<input type="submit" value="click after checking">
</form>
<div></div>
<script>
$("form").submit(function(){
var checkedinputs = $('input:radio');
$('div').append("you have chosen...");
$('div').append("<p>");
$.each( checkedinputs, function(i,item) {
    if($(checkedinputs[i]).prop('checked')){
       $('div').append($(checkedinputs[i]).val());
       $('div').append("<br>");
     }
  });
});
</script>
</body></html>
```

Output:



You have chosen...

Java

Note the selector in the code: var checkedinputs = $('input:radio');

Important note:

jQuery provides a much better selector: 'input:checked'. This picks up the element (in radio buttons it is only one element) which has been checked. This means you don't even need to check if the button is checked, and you don't need a loop to process the data. Here is a modified code:

```
$("form").submit(function(){
   var checkedInputs = $('input:checked');
   $('div').append("You have chosen...");
   $('div').append("<p>");
   $('div').append(checkedInputs.val());
   $('div').append("<br>");
});
```

Note:

In addition to submit button event, we can use mouse movement events with radio buttons.

13. Processing selection lists

A quick review of selection lists HTML:

A selection list is a dropdown list of items from which the user can choose one item or several items.

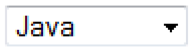
A selection list is created using the SELECT tag within a FORM tag.

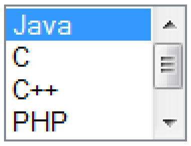
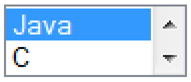
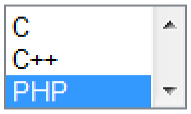
Syntax:

```
<form>
<select name="aname" size=number multiple >
<option value=value1> item1 </option>
<option value=value2> item2 </option>
<option value=value3> item3 </option>
...
...
...
</select>
</form>
```

Example:

HTML code	Browser display

`<form>`
`<select name="language">`
`<option value="Java">Java language</option>`
`<option value="C">C language </option>`
`<option value="C++">C++ language </option>`
`<option value="PHP">PHP language </option>`
`<option value="JavaScript">JavaScript language </option>`
`</select>`
`</form>`
	 Java

<form> <select name="language" **multiple**> <option value="Java">Java language </option> <option value="C">C language </option> <option value="C++">C++ language </option> <option value="PHP">PHP language </option> <option value="JavaScript">JavaScript language </option> </select> </form>	 Java C C++ PHP
<form> <select name="language" **size=2** multiple> <option value="Java">Java language </option> <option value="C">C language </option> <option value="C++">C++ language </option> <option value="PHP">PHP language </option> <option value="JavaScript">JavaScript language </option> </select> </form>	 Java C
<form> <select name="language" size=3> <option value="Java">Java language </option> <option value="C">C language </option> <option value="C++">C++ language </option> <option value="PHP" **selected**>PHP language </option> <option value="JavaScript">JavaScript language </option> </select> </form>	 C C++ PHP

Note:

When you want to select multiple items, keep the **ctrl** key pressed.

Processing selected items

As usual, we will use callback function on submit event:

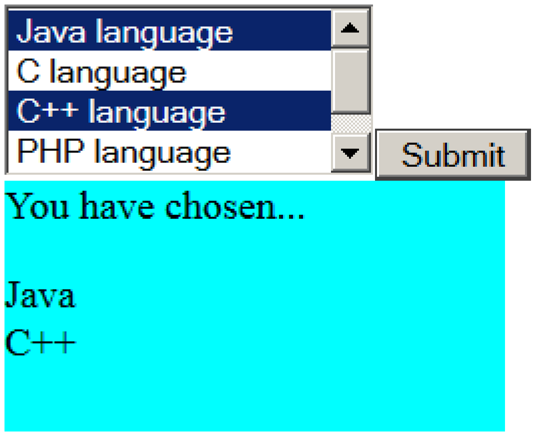
```
$("form").submit(function(){
});
```

The logic in the body of the function is as before, except the selector now is $(**'option:selected'**), which returns an "array" of values selected by the user.

Example:

```
<!doctype html>
<html>
 <head>
 <meta charset="utf-8">
<script src="http://code.jquery.com/jquery-1.10.1.min.js"></script>
 <title>Processing selections</title>
 <style>
 div{
    width:200px;
    height:200px;
    background-color:cyan;
    }
</style>
</head>
<body>
<form action="javascript:()">
<select name="language" multiple>
    <option value="Java">Java language</option>
    <option value="C">C language</option>
    <option value="C++">C++ language</option>
    <option value="PHP">PHP language</option>
    <option value="JavaScript">JavaScript language</option>
    <input type="submit" value="Submit">
</select>
</form>
<div></div>
<script>
$("form").submit(function(){
    var selectedItems = $('option:selected');
    $('div').append("You have chosen...");
    $('div').append("<p>");
    for(i=0;i<selectedItems.length;i++){
      $('div').append($(selectedItems[i]).val());
      $('div').append("<br>");
     }
});
</script>
</body>
</html>
```

Output:



```
You have chosen...

Java
C++
```

Note:

The output is the values of the attribute value in <option>.

Instead of,

$('div').append($(selectedItems[i]).val());

You can use,

$('div').append($(selectedItems[i]).text());

In this case, the output will be string between <option>…</option>