JQUERY - EVENTS HANDLING

We have the ability to create dynamic web pages by using events. Events are actions that can be detected by your Web Application.

Following are the examples events −

- A mouse click
- A web page loading
- Taking mouse over an element
- Submitting an HTML form
- A keystroke on your keyboard
- etc.

When these events are triggered you can then use a custom function to do pretty much whatever you want with the event. These custom functions call Event Handlers.

Binding event handlers

Using the jQuery Event Model, we can establish event handlers on DOM elements with the **bind** method as follows −

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

      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $('div').bind('click', function( event ){
               alert('Hi there!');
            });
         });
      </script>

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

   </head>

   <body>

      <p>Click on any square below to see the result:</p>

      <div >ONE</div>
      <div >TWO</div>
      <div >THREE</div>

   </body>

</html>
```

This code will cause the division element to respond to the click event; when a user clicks inside this division thereafter, the alert will be shown.

This will produce following result:

The full syntax of the bind command is as follows −

```
selector.bind( eventType[, eventData], handler)
```

Following is the description of the parameters −

- **eventType −** A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.
- **eventData −** This is optional parameter is a map of data that will be passed to the event handler.
- **handler −** A function to execute each time the event is triggered.

Removing event handlers

Typically, once an event handler is established, it remains in effect for the remainder of the life of the page. There may be a need when you would like to remove event handler.

jQuery provides the **unbind** command to remove an exiting event handler. The syntax of unbind is as follows :

```
selector.unbind(eventType, handler)

or

selector.unbind(eventType)
```

Following is the description of the parameters −

- **eventType −** A string containing a JavaScript event type, such as click or submit. Refer to the next section for a complete list of event types.
- **handler −** If provided, identifies the specific listener that's to be removed.

Event Types

The following are cross platform and recommended event types which you can bind using JQuery −

S.NO.	Event Type & Description
1	**blur** Occurs when the element loses focus.
2	**change** Occurs when the element changes.
3	**click**

Occurs when a mouse click.

4 **dblclick**

Occurs when a mouse double-click.

5 **error**

Occurs when there is an error in loading or unloading etc.

6 **focus**

Occurs when the element gets focus.

7 **keydown**

Occurs when key is pressed.

8 **keypress**

Occurs when key is pressed and released.

9 **keyup**

Occurs when key is released.

10 **load**

Occurs when document is loaded.

11 **mousedown**

Occurs when mouse button is pressed.

12 **mouseenter**

Occurs when mouse enters in an element region.

13 **mouseleave**

Occurs when mouse leaves an element region.

14 **mousemove**

Occurs when mouse pointer moves.

15 **mouseout**

Occurs when mouse pointer moves out of an element.

16 **mouseover**

Occurs when mouse pointer moves over an element.

17 **mouseup**

	Occurs when mouse button is released.
18	**resize** Occurs when window is resized.
19	**scroll** Occurs when window is scrolled.
20	**select** Occurs when a text is selected.
21	**submit** Occurs when form is submitted.
22	**unload** Occurs when documents is unloaded.

The Event Object

The callback function takes a single parameter; when the handler is called the JavaScript event object will be passed through it.

The event object is often unnecessary and the parameter is omitted, as sufficient context is usually available when the handler is bound to know exactly what needs to be done when the handler is triggered, however there are certain attributes which you would need to be accessed.

The Event Attributes

The following event properties/attributes are available and safe to access in a platform independent manner −

S.NO.	Property & Description
1	**altKey** Set to true if the Alt key was pressed when the event was triggered, false if not. The Alt key is labeled Option on most Mac keyboards.
2	**ctrlKey** Set to true if the Ctrl key was pressed when the event was triggered, false if not.
3	**data** The value, if any, passed as the second parameter to the bind command when the handler was established.
4	**keyCode** For keyup and keydown events, this returns the key that was pressed.

5 **metaKey**

Set to true if the Meta key was pressed when the event was triggered, false if not. The Meta key is the Ctrl key on PCs and the Command key on Macs.

6 **pageX**

For mouse events, specifies the horizontal coordinate of the event relative from the page origin.

7 **pageY**

For mouse events, specifies the vertical coordinate of the event relative from the page origin.

8 **relatedTarget**

For some mouse events, identifies the element that the cursor left or entered when the event was triggered.

9 **screenX**

For mouse events, specifies the horizontal coordinate of the event relative from the screen origin.

10 **screenY**

For mouse events, specifies the vertical coordinate of the event relative from the screen origin.

11 **shiftKey**

Set to true if the Shift key was pressed when the event was triggered, false if not.

12 **target**

Identifies the element for which the event was triggered.

13 **timeStamp**

The timestamp *inmilliseconds* when the event was created.

14 **type**

For all events, specifies the type of event that was triggered *forexample*, *click*.

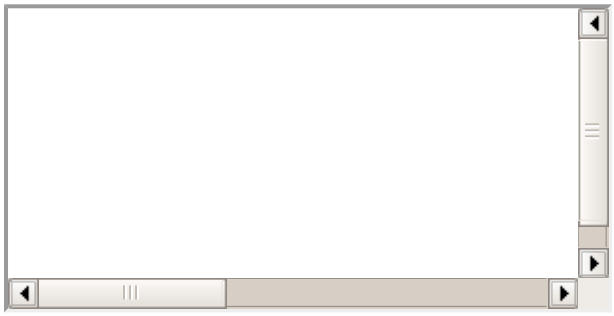
15 **which**

For keyboard events, specifies the numeric code for the key that caused the event, and for mouse events, specifies which button was pressed 1*forleft*, 2*formiddle*, 3*forright*.

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

```
      <script type="text/javascript" language="javascript">
         $(document).ready(function() {
            $('div').bind('click', function( event ){
               alert('Event type is ' + event.type);
               alert('pageX : ' + event.pageX);
               alert('pageY : ' + event.pageY);
               alert('Target : ' + event.target.innerHTML);
            });
         });
      </script>

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

   </head>

   <body>

      <p>Click on any square below to see the result:</p>

      <div >ONE</div>
      <div >TWO</div>
      <div >THREE</div>

   </body>

</html>
```

This will produce following result:



The Event Methods

There is a list of methods which can be called on an Event Object −

S.No.	Method & Description
1	preventDefault Prevents the browser from executing the default action.
2	isDefaultPrevented Returns whether event.preventDefault was ever called on this event object.
3	stopPropagation Stops the bubbling of an event to parent elements, preventing any parent handlers from being notified of the event.
4	isPropagationStopped Returns whether event.stopPropagation was ever called on this event object.

5	stopImmediatePropagation Stops the rest of the handlers from being executed.
6	isImmediatePropagationStopped Returns whether event.stopImmediatePropagation was ever called on this event object.

Event Manipulation Methods

Following table lists down important event-related methods −

S.No.	Method & Description
1	bind*type*, [*data*], *fn* Binds a handler to one or more events *likeclick* for each matched element. Can also bind custom events.
2	off*events*[, *selector*][, *handler(eventObject*]) This does the opposite of live, it removes a bound live event.
3	hover*over*, *out* Simulates hovering for example moving the mouse on, and off, an object.
4	on*events*[, *selector*][, *data*], *handler* Binds a handler to an event *likeclick* for all current − and future − matched element. Can also bind custom events.
5	one*type*, [*data*], *fn* Binds a handler to one or more events to be executed once for each matched element.
6	ready*fn* Binds a function to be executed whenever the DOM is ready to be traversed and manipulated.
7	trigger*event*, [*data*] Trigger an event on every matched element.
8	triggerHandler*event*, [*data*] Triggers all bound event handlers on an element.
9	unbind[*type*], [*fn*] This does the opposite of bind, it removes bound events from each of the matched elements.

Event Helper Methods

jQuery also provides a set of event helper functions which can be used either to trigger an event to bind any event types mentioned above.

Trigger Methods

Following is an example which would triggers the blur event on all paragraphs −

```
$("p").blur();
```

Binding Methods

Following is an example which would bind a **click** event on all the −

```
$("div").click( function () {
   // do something here
});
```

Here is a complete list of all the support methods provided by jQuery −

S.No.	Method & Description
1	**blur** Triggers the blur event of each matched element.
2	**blur*fn*** Bind a function to the blur event of each matched element.
3	**change** Triggers the change event of each matched element.
4	**change*fn*** Binds a function to the change event of each matched element.
5	**click** Triggers the click event of each matched element.
6	**click*fn*** Binds a function to the click event of each matched element.
7	**dblclick** Triggers the dblclick event of each matched element.
8	**dblclick*fn*** Binds a function to the dblclick event of each matched element.
9	**error** Triggers the error event of each matched element.

10 **error*fn***

Binds a function to the error event of each matched element.

11 **focus**

Triggers the focus event of each matched element.

12 **focus*fn***

Binds a function to the focus event of each matched element.

13 **keydown**

Triggers the keydown event of each matched element.

14 **keydown*fn***

Bind a function to the keydown event of each matched element.

15 **keypress**

Triggers the keypress event of each matched element.

16 **keypress*fn***

Binds a function to the keypress event of each matched element.

17 **keyup**

Triggers the keyup event of each matched element.

18 **keyup*fn***

Bind a function to the keyup event of each matched element.

20 **load*fn***

Binds a function to the load event of each matched element.

21 **mousedown*fn***

Binds a function to the mousedown event of each matched element.

22 **mouseenter*fn***

Bind a function to the mouseenter event of each matched element.

23 **mouseleave*fn***

Bind a function to the mouseleave event of each matched element.

24 **mousemove*fn***

Bind a function to the mousemove event of each matched element.

25 **mouseout***fn*

Bind a function to the mouseout event of each matched element.

26 **mouseover***fn*

Bind a function to the mouseover event of each matched element.

27 **mouseup***fn*

Bind a function to the mouseup event of each matched element.

28 **resize***fn*

Bind a function to the resize event of each matched element.

29 **scroll***fn*

Bind a function to the scroll event of each matched element.

30 **select**

Trigger the select event of each matched element.

31 **select***fn*

Bind a function to the select event of each matched element.

32 **submit**

Trigger the submit event of each matched element.

33 **submit***fn*

Bind a function to the submit event of each matched element.

34 **unload***fn*

Binds a function to the unload event of each matched element.