

<script>

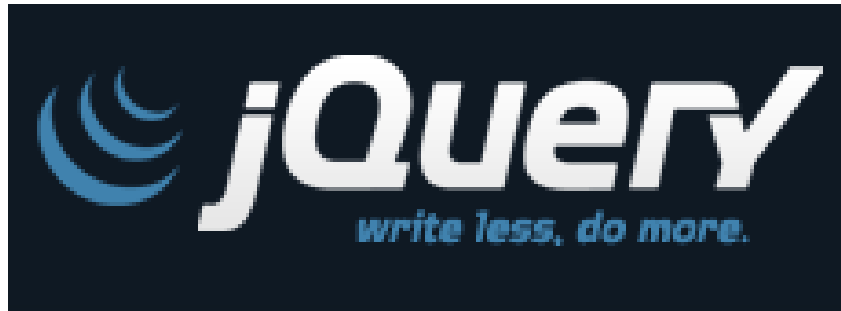


JavaScript

</script>

JQuery

Jquery Part 2



jQuery simplifies HTML document traversing, event handling, animating, and Ajax interactions for Rapid Web Development

JQuery Dom Manipulation



- ☐ Create Elements
- ☐ Insert Elements
- ☐ Remove Elements

JQuery Dom Manipulation (Cont.)

- Clone Elements



❑ Clone Elements:

- .clone() method.
- Used to create a copy of elements
- .clone(true), to copy element with its event handlers.
- Example:

```
//Clone all <p> elements and insert them at the end of the <body> element
$("button").click(function(){
    $("p").clone().appendTo("body");
});
```

JQuery Dom Manipulation (Cont.)

- Add & Insert Elements



❑ Add & Insert Elements(Cont.)

Method	Example
.append()	Inserts content at the end of the selected elements (inside).
	<pre>\$('#TestList').append('Appended item'); \$("p").append("Some appended text.");</pre>
.prepend()	Inserts content at the beginning of the selected elements(inside).
	<pre>\$('#TestList').prepend('Appended item')</pre>
.after()	Inserts content after the selected elements.
	<pre>\$("img").after("<p>test</p>");</pre>
.before()	Inserts content before the selected elements.
	<pre>\$("img").before("<p>test</p>");</pre>

JQuery Dom Manipulation - Add & Insert Elements(Cont.)



- ❑ Add Several New Elements With `append()` and `prepend()`

```
var txt1 = "<p>Text.</p>";           // Create element with HTML
var txt2 = $("<p></p>").text("Text."); // Create with jQuery
var txt3 = document.createElement("p"); // Create with DOM
txt3.innerHTML = "Text.";
$("#div1").append(txt1, txt2, txt3); // Append the new elements
```

- ❑ Add Several New Elements With `after()` and `before()`

```
var txt1 = "<b>I </b>";           // Create element with HTML
var txt2 = $("<i></i>").text("jQuery"); // Create with jQuery
var txt3 = document.createElement("b"); // Create with DOM
txt3.innerHTML = "jQuery!";
$("img").after(txt1, txt2, txt3); // Insert new elements after img
```

JQuery Dom Manipulation (Cont.)

- Remove Elements



❑ Remove Elements:

- **.remove()** - Removes the selected element (and its child elements).
- **.empty()** - Removes the child elements from the selected element.
- Example:

```
//removes the selected div and its child elements.
```

```
$("#div1").remove();
```

```
//remove() method also accepts one parameter, which allows you to filter the  
elements to be removed.
```

```
$("p").remove(".italic");
```

```
//removes the child elements of the selected div.
```

```
$("#div1").empty();
```

- JQuery Dom & HTML Manipulation Reference:
http://www.w3schools.com/jquery/jquery_ref_html.asp
- JQuery DOM Traversing Reference:
http://www.w3schools.com/jquery/jquery_ref_traversing.asp

JQuery Advanced Event handling



❑ .bind() & unbind()

Method	Example
.bind(eventType [, eventData], handler)	Attach a handler to an event for the elements.
	<ul style="list-style-type: none">- eventType: A string containing one or more DOM event types, such as "click" or "submit,".- eventData: An object containing data that will be passed to the event handler.- handler: A function to execute each time the event is triggered.
.unbind(eventType [, handler])	Remove a previously-attached event handler from the elements.
	<ul style="list-style-type: none">- eventType: A string containing a JavaScript event type, such as click or submit.- handler: The function that is to be no longer executed.

JQuery Advanced Event handling (Cont.)



❑ .bind() & unbind() (cont.):

- Bind an event handler to element:

```
$( "#foo" ).bind( "click", function() {  
    alert( "User clicked on 'foo.'" );  
});
```

- Bind Multiple Events:

```
$( "#foo" ).bind( "mouseenter mouseleave", function() {  
    $( this ).toggleClass( "entered" );  
});
```

- Passing Event Data:

```
//var message = "Spoon!";  
var message = $("#t1").val();  
$( "#foo" ).bind( "click", {msg: message, title:"msgTitle"},  
function( event ) {  
    alert( event.data.msg );  
});
```

JQuery Advanced Event handling (Cont.)



❑ .bind() & unbind() (cont.):

- Unbind all event handlers from element:

```
$( "#foo" ).unbind();
```

- Unbind event handler for click event from element:

```
$( "#foo" ).unbind( "click" );
```

- Unbind event handler for click event for specific handler from element:

```
$( "#foo" ).unbind( "click" , fun1);
```

- Starting from jQuery 1.7, the .on() and .off() methods are preferred to attach and remove event handlers on elements

JQuery Advanced Event handling (Cont.)



❑ .delegate() & .undelegate()

Method	Example
.delegate(selector, eventType, handler)	Attach a handler to one or more events for all elements that match the selector, now or in the future (even for elements that isn't created yet) , based on a specific set of root elements.
	<ul style="list-style-type: none">- selector: A selector to filter the elements that trigger the event.- eventType: A string containing one or more space-separated JavaScript event types, such as "click" or "keydown," or custom event names.- handler: A function to execute at the time the event is triggered.
.undelegate() .undelegate(selector, eventType[,handler])	Remove a handler from the event for all elements which match the current selector, based upon a specific set of root elements.
	<ul style="list-style-type: none">- selector: A selector which will be used to filter the event results.- eventType: A string containing a JavaScript event type, such as "click" or "keydown"- handler: A function to execute at the time the event is triggered.

jQuery Advanced Event handling (Cont.)



❑ .delegate() & .undelegate() (cont.):

- As of jQuery 1.7, .delegate() has been superseded by the .on() method. For earlier versions, however, it remains the most effective means to use event delegation.

```
1 // jQuery 1.4.3+
2 $( elements ).delegate( selector, events, data, handler );
3 // jQuery 1.7+
4 $( elements ).on( events, selector, data, handler );
```

- Using delegate to handle events:

```
$( "table" ).delegate( "td", "click", function() {
    $( this ).toggleClass( "chosen" );
});
```

- Equivalent to:

```
$( "table" ).on( "click", "td", function() {
    $( this ).toggleClass( "chosen" );
});
```

JQuery Advanced Event handling (Cont.)



❑ .delegate() & .undelegate() (cont.)

- delegate() & on(), attaches a click event handler to all selected elements - **even new ones.**

//all new created paragraphs will have the same event handler

```
$( "body" ).delegate( "p", "click", function() {  
    $( this ).after( "<p>Another paragraph!</p>" );  
});
```

//handle click event for all inputs of type button

```
$( "form" ).delegate( ":button", "click", function(){//do any});
```

- Undelegate()

//undelegate all events

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

//undelegate event handler of click event

```
$( "p" ).undelegate( "click" );
```

//Unbind all click events delegated for buttons

```
$( "form" ).undelegate( ":button ", "click" );
```

//Unbind click events for specific function for buttons

```
$( "form" ).undelegate( ":button ", "click" , fun1);
```

JQuery Advanced Event handling (Cont.)



❑ .delegate() & .undelegate() (cont.):

- Demo on delegate() & on()
 - <http://api.jquery.com/delegate/>
 - <http://api.jquery.com/on/>
- The **.off()** method removes event handlers that were attached with **.on()** [Equivalent to undelegate()]

//Remove all event handlers from all paragraphs:

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

//Remove all delegated click handlers from all paragraphs:

```
$( "p" ).off( "click", "*" );
```

// Remove specific event handler from p

```
$( "body" ).off( "click", "p", foo );
```

- Demo on undelegate() & off()
 - <http://api.jquery.com/undelegate/>
 - <http://api.jquery.com/off/>
- Reference for Event Handler Attachment:
 - <http://api.jquery.com/category/events/event-handler-attachment/>

Event Object



❑ Event Object:

```
$( document ).on( 'click', function( event ) {  
    console.log( event.type ); // The event type, eg. "click"  
    console.log( event.which ); // The button or key that was pressed.  
    console.log( event.target ); // The originating element.  
    console.log( event.pageX ); // The document mouse X coordinate.  
    console.log( event.pageY ); // The document mouse Y coordinate.  
});
```

❑ Preventing Default action:

```
$( 'a' ).on( 'click', function( event ) {  
    // Prevent the default action.  
    event.preventDefault();  
    // Log stuff.  
    console.log( 'I was just clicked!' );  
});
```

jQuery UI





❑ JQuery UI provides a comprehensive set of:

- Effects
 - ➔ (Hide, Show, animate,...).
- Interactions
 - ➔ (Draggable, Droppable, Resizable, Sortable,..).
- UI Widgets
 - ➔ (Accordion, Menu, Dialog, tabs,...).

❑ JQuery plugins

- Carousel, Image Zoom,....

DEMO!

jQuery UI --- Effects



□ Effects :

- jQuery provides a simple interface for doing various kind of amazing effects.
- jQuery methods allow us to quickly apply commonly used effects, which fall into 2 categories:
 - JQuery Effect Methods
 - UI Library Based Effects

jQuery UI --- Effects(Cont.)



□ jQuery Effect Methods:

- jQuery supports us with basic Effect methods which can be used in:
 - Showing and Hiding elements
 - Toggling the elements
 - Fading
 - Sliding
 - Animation

jQuery UI --- Effects(Cont.)



□ Showing & Hiding Elements:

```
[selector].show ( speed , [callback] );
```

- speed → "slow", "normal", or "fast".
- callback → a function to be executed whenever the animation completes.

```
[selector].hide ( speed, [callback] );
```

jQuery UI --- Effects(Cont.)



❑ Toggling Elements:

- jQuery provides methods to toggle the display state of elements between revealed or hidden.
- If the element is initially displayed, it will be hidden; if hidden, it will be shown.

```
[selector].toggle ([speed][, callback])
```

jQuery UI --- Effects(Cont.)



❑ Fading Functions:

fadeIn(speed, [callback])	Fade in all matched elements by adjusting their opacity and firing an optional callback after completion.
fadeOut(speed, [callback])	Fade out all matched elements by adjusting their opacity to 0, then setting display to "none" and firing an optional callback after completion.
fadeTo(speed, opacity, [callback])	Fade the opacity of all matched elements to a specified opacity and firing an optional callback after completion.

jQuery UI --- Effects(Cont.)



□ Slide Functions:

slideDown(speed, [callback])	Reveal all matched elements by adjusting their height and firing an optional callback after completion.
slideUp(speed, [callback])	Hide all matched elements by adjusting their height and firing an optional callback after completion.
slideToggle(speed, [callback])	Toggle the visibility of all matched elements by adjusting their height and firing an optional callback after completion.

jQuery UI --- Effects(Cont.)



□ Animations Functions:

- The animate() method performs a custom animation of a set of CSS properties.

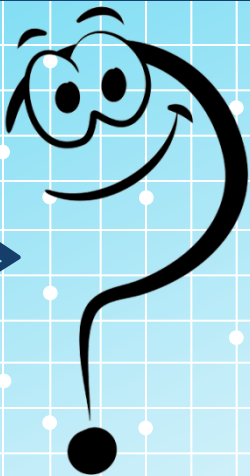
```
selector.animate( params, [duration, easing, callback] );
```


<script>



JavaScript

</script>

<SCRIPT>  </SCRIPT>

<script>document.writeln("Thank
You!")</script>