JQUERY

The on() Method

The on() method attaches one or more event handlers for the selected elements.

Attach a click event to a <p> element:

Example

```
$("p").on("click", function(){
    $(this).hide();
});
```

Meet - ebe-xuhk-jas - Google Chrome

The hide() and show() Method

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

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

The hide() and show() Method

Syntax:

- $(selector).hide(speed,callback);
- $(selector).show(speed,callback);
- The optional speed parameter specifies the speed of the hiding/showing, and can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the hide() or show() method completes (you will learn more about callback functions in a later chapter).
- The following example demonstrates the speed parameter with hide():

Example

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

jQuery toggle()

With jQuery, you can toggle between the hide() and show() methods with the toggle() method.

Shown elements are hidden and hidden elements are shown:

Example

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

Syntax:

```
$(selector).toggle(speed,callback);
```

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

The optional callback parameter is a function to be executed after toggle() completes.

jQuery Fading Methods

With jQuery you can fade elements in and out of visibility.

jQuery Fading Methods: With jQuery you can fade an element in and out of visibility.

jQuery has the following fade methods:

- fadeIn()
- fadeOut()
- fadeToggle()
- fadeTo()

jQuery fadeIn() Method

The jQuery fadeIn() method is used to fade in a hidden element.

Syntax:

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

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the fading completes.

The following example demonstrates the fadeIn() method with different parameters:

Example

```
$("button").click(function(){
    $("#div1").fadeIn();
    $("#div2").fadeIn("slow");
    $("#div3").fadeIn(3000);
});
```

jQuery Sliding Methods

With jQuery you can create a sliding effect on elements.

jQuery has the following slide methods:

- slideDown()
- slideUp()
- slideToggle()

jQuery slideDown() Method

The jQuery slideDown() method is used to slide down an element.

Syntax:

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

The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.

The optional callback parameter is a function to be executed after the sliding completes

The following example demonstrates the slideDown() method:

Example

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

jQuery Animations - The animate() Method

- The jQuery animate() method is used to create custom animations.

Syntax:

```
$(selector).animate({params},speed,callback);
```

- The required params parameter defines the CSS properties to be animated
- The optional speed parameter specifies the duration of the effect. It can take the following values: "slow", "fast", or milliseconds.
- The optional callback parameter is a function to be executed after the animation completes.
- The following example demonstrates a simple use of the animate() method; it moves a element to the right, until it has reached a left property of 250px:

Example

```
$("button").click(function(){
   $("div").animate({left: '250px'});
});
```

jQuery animate() - Using Relative Values

It is also possible to define relative values (the value is then relative to the element's current value). This is done by putting += or -= in front of the value

Example

```
$("button").click(function(){
  $("div").animate({
    left: '250px',
    height: '+=150px',
    width: '+=150px'
  }); });
```

jQuery animate() - Using Pre-defined Values

You can even specify a property's animation value as "show", "hide", or "toggle":

Example

```
$("button").click(function(){
  $("div").animate({
    height: 'toggle'
  });
});
```

jQuery animate() - Uses Queue Functionality

By default, jQuery comes with queue functionality for animations.

This means that if you write multiple animate() calls after each other, jQuery creates an "internal" queue with these method calls. Then it runs the animate calls ONE by ONE.

So, if you want to perform different animations after each other, we take advantage of the queue functionality:

Example 1

```
$("button").click(function(){
    var div = $("div");
    div.animate({height: '300px', opacity: '0.4'}, "slow");
    div.animate({width: '300px', opacity: '0.8'}, "slow");
    div.animate({height: '100px', opacity: '0.4'}, "slow");
    div.animate({width: '100px', opacity: '0.8'}, "slow");
});
```

jQuery animate()

The example below first moves the element to the right, and then increases the font size of the text:

Example 2

```
$("button").click(function(){
    var div = $("div");
    div.animate({left: '100px'}, "slow");
    div.animate({fontSize: '3em'}, "slow");
});
```

jQuery Callback Functions

- JavaScript statements are executed line by line. However, with effects, the next line of code can be run even though the effect is not finished. This can create errors.
- To prevent this, you can create a callback function.
- A callback function is executed after the current effect is finished.
- Typical syntax: $(selector).hide(speed,callback);

jQuery Callback Functions

Examples

The example below has a callback parameter that is a function that will be executed after the hide effect is completed:

Example with Callback

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

The example below has no callback parameter, and the alert box will be displayed before the hide effect is completed:

Example without Callback

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

Until now we have been writing jQuery statements one at a time (one after the other).

However, there is a technique called chaining, that allows us to run multiple jQuery commands, one after the other, on the same element(s).

Tip: This way, browsers do not have to find the same element(s) more than once.

To chain an action, you simply append the action to the previous action.

jQuery Method Chaining

The following example chains together the css(), slideUp(), and slideDown() methods. The "p1" element first changes to red, then it slides up, and then it slides down:

Example

```
$("#p1").css("color", "red").slideUp(2000).slideDown(2000);
```

We could also have added more method calls if needed.

Tip: When chaining, the line of code could become quite long. However, jQuery is not very strict on the syntax; you can format it like you want, including line breaks and indentations.

jQuery Method Chaining

This also works just fine:

Example

```
$("#p1").css("color", "red")
  .slideUp(2000)
  .slideDown(2000);
```

jQuery methods for DOM manipulation are:

- text() - Sets or returns the text content of selected elements
- html() - Sets or returns the content of selected elements (including HTML markup)
- val() - Sets or returns the value of form fields

The following example demonstrates how to get content with the jQuery text() and html() methods: Example

```
$("#btn1").click(function(){
  alert("Text: " + $("#test").text());
});
$("#btn2").click(function(){
  alert("HTML: " + $("#test").html());
});
```

The following example demonstrates how to get the value of an input field with the jQuery val() method:

Example

```
$("#btn1").click(function(){
  alert("Value: " + $("#test").val());
});
```

A Callback Function for text(), html(), and val()

Example

```
$("#btn1").click(function(){
   $("#test1").text(function(i, origText){
      return "Old text: " + origText + " New text: Hello world!
      (index: " + i + ")";
   });
});

$("#btn2").click(function(){
   $("#test2").html(function(i, origText){
      return "Old html: " + origText + " New html: Hello <b>world!</b>
      (index: " + i + ")";
   });
});
```

Add New HTML Content

We will look at four jQuery methods that are used to add new content:

append() - Inserts content at the end of the selected elements

prepend() - Inserts content at the beginning of the selected elements

after() - Inserts content after the selected elements

before() - Inserts content before the selected elements

jQuery.pdf - Personal - Microsoft Edge

Add New HTML Content

In the following example, we create several new elements. The elements are created with text/HTML, jQuery, and JavaScript/DOM. Then we append the new elements to the text with the append() method (this would have worked for prepend() too) :

Example

```
function appendText() {
    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.";
    $("body").append(txt1, txt2, txt3);      // Append the new elements
}
```

jQuery to remove existing HTML elements.

Remove Elements/Content

To remove elements and content, there are mainly two jQuery methods:

- remove() - Removes the selected element (and its child elements)
- empty() - Removes the child elements from the selected element

jQuery remove() Method

The jQuery remove() method removes the selected element(s) and its child elements.

Example

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

jQuery empty() Method

The jQuery empty() method removes the child elements of the selected element(s).

Example

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