DOM in JQuery

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 append() Method

The jQuery append() method inserts content AT THE END of the selected HTML elements.

```
$("p").append("Some appended text.");
```

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.
min.js"></script>
<script>
$(document).ready(function(){
  $("#btn1").click(function(){
    $("p").append(" <b>Appended text</b>.");
  });

  $("#btn2").click(function(){
    $("ol").append("<li>Appended item</li>");
  });
});
</script>
</head>
<body>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

<ol>//ordered list
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ol>

<button id="btn1">Append text</button>
<button id="btn2">Append list items</button>

</body>
</html>
```

jQuery prepend() Method

The jQuery prepend() method inserts content AT THE BEGINNING of the selected HTML elements.

$("p").prepend("Some prepended text.");

```
<!DOCTYPE html>
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.
min.js"></script>
<script>
$(document).ready(function(){
  $("#btn1").click(function(){
    $("p").prepend("<b>Prepended text</b>. ");
  });
  $("#btn2").click(function(){
    $("ol").prepend("<li>Prepended item</li>");
  });
});
</script>
</head>
<body>

<p>This is a paragraph.</p>
<p>This is another paragraph.</p>

<ol>
  <li>List item 1</li>
  <li>List item 2</li>
  <li>List item 3</li>
</ol>

<button id="btn1">Prepend text</button>
<button id="btn2">Prepend list item</button>

</body>
</html>
```

Add Several New Elements With append() and prepend()

In both examples above, we have only inserted some text/HTML at the beginning/end of the selected HTML elements.

However, both the append() and prepend() methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML (like we have done in the examples above), with jQuery, or with JavaScript code and DOM elements.

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) :

```
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
}
```

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
function appendText() {
  var txt1 = "<p>Text.</p>";       // Create text with HTML
  var txt2 = $("<p></p>").text("Text.");  // Create text with jQuery
  var txt3 = document.createElement("p");
  txt3.innerHTML = "Text.";        // Create text with DOM
  $("body").append(txt1, txt2, txt3);   // Append new elements
}
</script>
</head>
<body>

<p>This is a paragraph.</p>
<button onclick="appendText()">Append text</button>

</body>
</html>
```

jQuery after() and before() Methods

The jQuery after() method inserts content AFTER the selected HTML elements.

The jQuery before() method inserts content BEFORE the selected HTML elements.

```
$("img").after("Some text after");

$("img").before("Some text before");
```

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#btn1").click(function(){
    $("img").before("<b>Before</b>");
  });

  $("#btn2").click(function(){
    $("img").after("<i>After</i>");
  });
});
</script>
</head>
<body>
<img src="/images/file1.gif" alt="jQuery" width="100" height="140"><br><br>
<button id="btn1">Insert before</button>
<button id="btn2">Insert after</button>
</body>
</html>
```

Add Several New Elements With after() and before()

Also, both the **after() and before()** methods can take an infinite number of new elements as parameters. The new elements can be generated with text/HTML (like we have done in the example above), with jQuery, or with JavaScript code and DOM elements.

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

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

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
function afterText() {
  var txt1 = "<b>I </b>";           // Create element with HTML
  var txt2 = $("<i></i>").text("love ");  // Create with jQuery
  var txt3 = document.createElement("b");   // Create with DOM
  txt3.innerHTML = "jQuery!";
  $("img").after(txt1, txt2, txt3);    // Insert new elements after img
}
</script>
</head>
<body>

<img src="/images/file1.gif" alt="jQuery" width="100" height="140">

<p>Click the button to insert text after the image.</p>

<button onclick="afterText()">Insert after</button>

</body>
</html>
```

jQuery - Remove 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.

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

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").remove();
  });
});
</script>
</head>
<body>
<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">
This is some text in the div.
<p>This is a paragraph in the div.</p>
<p>This is another paragraph in the div.</p>
</div>
<br>
<button>Remove div element</button>
</body>
</html>
```

jQuery empty() Method

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

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

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#div1").empty();
  });
});
</script>
</head>
<body>
<div id="div1" style="height:100px;width:300px;border:1px solid black;background-color:yellow;">
This is some text in the div.
<p>This is a paragraph in the div.</p>
<p>This is another paragraph in the div.</p>
</div>
<br>
<button>Empty the div element</button>

</body>
</html>
```

Filter the Elements to be Removed

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

The parameter can be any of the jQuery selector syntaxes.

The following example removes all <p> elements with class="test"

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

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").remove(".test");
  });
});
</script>
<style>
.test {
  color: red;
  font-size: 20px;
}
</style>
</head><body>
<p>This is a paragraph.</p>
<p class="test">This is another paragraph.</p>
<p class="test">This is another paragraph.</p>
<button>Remove all p elements with class="test"</button>
</body></html>
```

This example removes all <p> elements with class="test" and class="demo":

```
$("p").remove(".test, .demo");
```

```
<html>
<head>
<script
src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").remove(".test, .demo");
  });
});
</script>
<style>
.test {
  color: red;
  font-size: 20px;
}
.demo {
  color: green;
  font-size: 25px;
}
</style>
</head>
<body>

<p>This is a paragraph.</p>
<p class="test">This is p element with class="test".</p>
<p class="test">This is p element with class="test".</p>
<p class="demo">This is p element with class="demo".</p>

<button>Remove all p elements with class="test" and
class="demo"</button>

</body>
</html>
```

jQuery Effects - Animation

jQuery Animations - The animate() Method

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

$(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:

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

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({left: '250px'});
  });
});
</script>
</head>
<body>

<button>Start Animation</button>

<p>By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!</p>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>
</html>
```

jQuery animate() - Manipulate Multiple Properties

Notice that multiple properties can be animated at the same time:

```
$("button").click(function(){
  $("div").animate({
    left: '250px',
    opacity: '0.5',
    height: '150px',
    width: '150px'
  });
});
```

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({
      left: '250px',
      opacity: '0.5',
      height: '150px',
      width: '150px'
    });
  });
});
</script>
</head>
<body>
<button>Start Animation</button>
<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>
</body>
</html>
```

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:

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

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({
      left: '250px',
      height: '+=150px',
      width: '+=150px'
    });
  });
});
</script>
</head>
<body>
<button>Start Animation</button>
<p>By default, all HTML elements have a static position, and cannot be moved. To manipulate the position, remember to first set the CSS position property of the element to relative, fixed, or absolute!</p>
<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>
</body>
</html>
```

jQuery animate() - Using Pre-defined Values

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

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

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("div").animate({
      height: 'toggle'
    });
  });
});
</script>
</head>
<body>

<p>Click the button multiple times to toggle the animation.</p>

<button>Start Animation</button>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>
</html>
```

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:

```
$("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");
});
```

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("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");
  });
});
</script>
</head>
<body>

<button>Start Animation</button>

<div style="background:#98bf21;height:100px;width:100px;position:absolute;"></div>

</body>
</html>
```

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

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

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    var div = $("div");
    div.animate({left: '100px'}, "slow");
    div.animate({fontSize: '3em'}, "slow");
  });
});
</script>
</head>
<body>

<button>Start Animation</button>

<div style="background:#98bf21;height:100px;width:200px;position:absolute;">HELLO</div>

</body>
</html>
```

jQuery Stop Animations

jQuery stop() Method

The jQuery stop() method is used to stop an animation or effect before it is finished.

The stop() method works for all jQuery effect functions, including sliding, fading and custom animations.

$(selector).stop(stopAll,goToEnd);

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("#flip").click(function(){
    $("#panel").slideDown(5000);
  });
  $("#stop").click(function(){
    $("#panel").stop();
  });
});
</script>
<style>
#panel, #flip {
  padding: 5px;
  font-size: 18px;
  text-align: center;
  background-color: #555;
  color: white;
  border: solid 1px #666;
  border-radius: 3px;
}

#panel {
  padding: 50px;
  display: none;
}
</style>
</head>
<body>

<button id="stop">Stop sliding</button>

<div id="flip">Click to slide down panel</div>
<div id="panel">Hello world!</div>

</body>
</html>
```

jQuery Callback Functions

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);**

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide("slow", function(){
      alert("The paragraph is now hidden");
    });
  });
});
</script>
</head>
<body>

<button>Hide</button>

<p>This is a paragraph with little content.</p>

</body>
</html>
```

Example with Callback

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("p").hide(1000);
    alert("The paragraph is now hidden");
  });
});
</script>
</head>
<body>

<button>Hide</button>

<p>This is a paragraph with little content.</p>

</body>
</html>
```

Example without Callback

jQuery - Chaining

jQuery Method Chaining

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.

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:

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#p1").css("color", "red").slideUp(2000).slideDown(2000);
  });
});
</script>
</head>
<body>

<p id="p1">jQuery is fun!!</p>

<button>Click me</button>

</body>
</html>
```

```
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/3.5.1/jquery.min.js"></script>
<script>
$(document).ready(function(){
  $("button").click(function(){
    $("#p1").css("color", "red")
      .slideUp(2000)
      .slideDown(2000);
  });
});
</script>
</head>
<body>

<p id="p1">jQuery is fun!!</p>

<button>Click me</button>

</body>
</html>
```