The DOM and jQuery functions and selectors

Lesson 3

Plan for this lesson

Introduction to the DOM

- Code along
- More about manipulating the DOM

JavaScript Frameworks

- Angular
- Backbone.js
- jQuery
- Node.js

jQuery

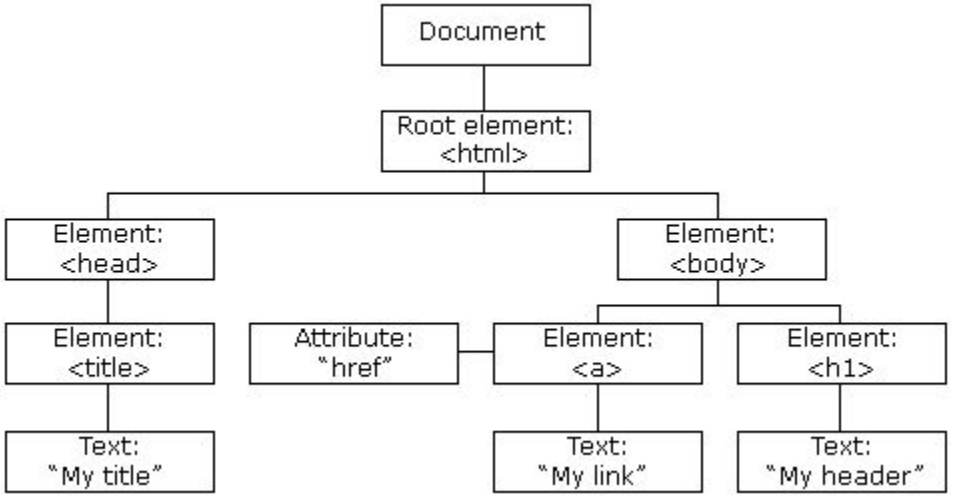
- Code along

The DOM

The DOM What is the DOM

When a web page is loaded, the browser creates a **D**ocument **O**bject **M**odel of the page.

The **HTML DOM** model is constructed as a tree of Objects:



The DOM What is the DOM

The HTML DOM is a standard **object** model and **programming interface** for HTML. It defines:

- The HTML elements as **objects**
- The **properties** of all HTML elements
- The **methods** to access all HTML elements
- The **events** for all HTML elements

In other words: **The HTML DOM is a standard for how to get, change, add, or delete HTML elements**.

- HTML DOM methods are actions you can perform (on HTML Elements).
- HTML DOM properties are values (of HTML Elements) that you can set or change.

The DOM What is the DOM

The HTML DOM can be accessed with JavaScript (and with other programming languages).

In the DOM, all HTML elements are defined as **objects**.

The programming interface is the properties and methods of each object.

A **property** is a value that you can get or set (like changing the content of an HTML element).

A **method** is an action you can do (like add or deleting an HTML element).

The DOM What can JavaScript do to the DOM

Via the object model JavaScript has a model to manipulate and thus create dynamic HTML:

- JavaScript can change all the HTML elements in the page
- JavaScript can change all the HTML attributes in the page
- JavaScript can change all the CSS styles in the page
- JavaScript can remove existing HTML elements and attributes
- JavaScript can add new HTML elements and attributes
- JavaScript can react to all existing HTML events in the page
- JavaScript can create new HTML events in the page

Code example

```
<html>
<body>

<p id="demo"></p>

<script>
document.getElementById("demo").innerHTML = "Hello World!";
</script>

</body>
</html>
```

We've used this before many times, so nothing new here.

We get the content of an element is by using the **innerHTML** property, which is useful for getting or replacing the content of HTML elements.

The innerHTML property can be used to get or change any HTML element, including <html> and <body>.

The DOM Finding and changing HTML elements

document.getElementById(id)	Find an element by element id
document.getElementsByTagName(name)	Find elements by tag name
document.getElementsByClassName(name)	Find elements by class name
element.innerHTML = new html content	Change the inner HTML of an element
element.attribute = new value	Change the attribute value of an HTML element
element.setAttribute(attribute, value)	Change the attribute value of an HTML element
element.style.property = new style	Change the style of an HTML element

The DOM Adding and deleting HTML elements and event handlers

document.createElement(element)	Create an HTML element
document.removeChild(element)	Remove an HTML element
document.appendChild(element)	Add an HTML element
document.replaceChild(element)	Replace an HTML element
document.write(text)	Write into the HTML output stream

document.getElementById(id).onclick = function(){code}

Adding event handler code to an onclick event

Code along

Adding HTML elements

```
<!DOCTYPE html>
<html>
<body>

<h1>JavaScript while</h1>

<p id="demo"></p>

<script>
document.body.onload = addElement;

function addElement () {
  // create a new div element and give it some content
  var newDiv = document.createElement("div");
  var newContent = document.createTextNode("Hi there and greetings!");
  newDiv.appendChild(newContent); //add the text node to the newly created div.

  // add the newly created element and its content into the DOM
  var currentDiv = document.getElementById("div1");
  document.body.insertBefore(newDiv, currentDiv);
}
</script>

  <div id="div1">The text above has been created dynamically.</div>
</body>
</html>
```

The DOM Finding HTML elements

document.anchors	Returns all <a> elements that have a name attribute
document.baseURI	Returns the absolute base URI of the document
document.body	Returns the <body> element
document.cookie	Returns the document's cookie
document.doctype	Returns the document's doctype
document.documentElement	Returns the <html> element
document.documentMode	Returns the mode used by the browser
document.documentURI	Returns the URI of the document
document.domain	Returns the domain name of the document server
document.embeds	Returns all <embed> elements

The DOM Finding HTML elements

document.forms	Returns all <form> elements
document.head	Returns the <head> element
document.images	Returns all <img> elements
document.implementation	Returns the DOM implementation
document.inputEncoding	Returns the document's encoding (character set)
document.lastModified	Returns the date and time the document was updated
document.links	Returns all <area> and <a> elements that have a href attribute
document.readyState	Returns the (loading) status of the document

The DOM Finding HTML elements

document.referrer	Returns the URI of the referrer (the linking document)
document.scripts	Returns all <script> elements
document.strictErrorChecking	Returns if error checking is enforced
document.title	Returns the <title> element
document.URL	Returns the complete URL of the document

The DOM Finding HTML elements by ID

The easiest way to find an HTML element in the DOM, is by using the element id.

This example finds the element with id="intro":

```
<!DOCTYPE html>
<html>
<body>
<p id="intro">Hello World!</p>
<p>This example demonstrates the <b>getElementById</b> method!</p>
<p id="demo"></p>
<script>
var myElement = document.getElementById("intro");
document.getElementById("demo").innerHTML =
"The text from the intro paragraph is " + myElement.innerHTML;
</script>
</body>
</html>
```

The DOM Finding HTML elements by Tag Name

This example finds the element with id="main", and then finds all <p> elements inside "main":

```
<!DOCTYPE html>
<html>
<body>
<p>Hello World!</p>
<div id="main">
<p>The DOM is very useful.</p>
<p>This example demonstrates the <b>getElementsByTagName</b> method</p>
</div>
<p id="demo"></p>
<script>
    var x = document.getElementById("main");
    var y = x.getElementsByTagName("p");
    document.getElementById("demo").innerHTML =
    'The first paragraph (index 0) inside "main" is: ' + y[0].innerHTML;
</script>
</body>
</html>
```

The DOM Finding HTML elements by Class Name

If you want to find all HTML elements with the same class name, use getElementsByClassName().

This example returns a list of all elements with class="intro".

```
<!DOCTYPE html>
<html>
<body>
<p>Hello World!</p>
<p class="intro">The DOM is very useful.</p>
<p class="intro">This example demonstrates the <b>getElementsByClassName</b> method.</p>
<p id="demo"></p>
<script>
    var x = document.getElementsByClassName("intro");
    document.getElementById("demo").innerHTML =
    'The first paragraph (index 0) with class="intro": ' + x[0].innerHTML;
</script>
</body>
</html>
```

The DOM Changing the HTML output stream

In JavaScript, document.write() can be used to write directly to the HTML output stream.

```
<!DOCTYPE html>
<html>
<body>
<script>
    document.write(Date());
</script>
</body>
</html>
```

The DOM Changing the HTML output stream

The easiest way to modify the content of an HTML element is by using the **innerHTML** property.

To change the content of an HTML element, use the syntax:

document.getElementById(id).innerHTML = new HTML

```
<!DOCTYPE html>
<html>
<body>
<p id="p1">Hello World!</p>
<script>
    document.getElementById("p1").innerHTML = "New text!";
</script>
<p>The paragraph above was changed by a script.</p>
</body>
</html>
```

The DOM Changing the HTML output stream

To change the value of an HTML attribute, use this syntax:

document.getElementById(id).attribute=new value

```
<!DOCTYPE html>
<html>
<body>
<img id="image" src="dom_img1.jpg" width="620" height="400">
<script>
    document.getElementById("image").src = "dom_img2.jpg";
</script>
<p>The original image was dom_img1.jpg, but the script changed it to dom_img2.jpg</p>
</body>
</html>
```

JavaScript Frameworks

JavaScript Frameworks

The aim of general frameworks is to balance the differences between browsers by creating a new, unified API to perform general tasks like DOM manipulation and Ajax functionality.

Modern web development has a focus on responsiveness, resilience, scalability and accuracy. Our solutions should be responsive to real-time demands: we want our systems to be resilient against peak performance and the kind of demands that come from unknown sources, and we want our projects to be scalable so that when the time comes, we can easily upgrade or downgrade our software for optimal performance. The following JavaScript frameworks have been built with reactive web development in mind, and they have all been used in thousands of solutions out in the wild.

JavaScript Frameworks AngularJS

Angular is an MVC-type framework. It offers two-way data binding between models and views. This data binding allows for an automatic update on both sides whenever there is a data change. It enables you to build reusable View Components. It provides a services framework to allow easily backend-frontend server communication.

```
8   app.controller("PanelController", function(){
9       this.tab = 1;
10
11      this.selectTab = function(setTab) {
12          this.tab = setTab;
13      };
14      this.isSelected = function(checkTab){
15          return this.tab === checkTab;
16      };
17  });
```

JavaScript Frameworks Backbone

Backbone is especially popular with teams looking for a simple structure for their small web applications without bringing in a large framework like Angular or Ember.

Backbone provides a full MVC framework along with routing. The Models allow for key-value binding and events for handling data changes. Models (and Collections) can connect to RESTful APIs. The Views have declarative event handling, and the Router does an excellent job of handling your URL and state management. Everything you need to build a Single Page Application without offering too much and without unnecessary complexity.

It is a cornerstone in the Appcelerator Titanium mobile app development tool.

JavaScript Frameworks jQuery

jQuery is the most used JavaScript library in the world for a reason. It makes DOM traversal, event handling, animation, AJAX much simpler and easier across all browsers.

```
$(document).ready(function(){
    $("button").click(function(){
        $("p:first").addClass("intro");
    });
});
```

We'll look much closer at jQuery later this lesson and in the rest of the course.

JavaScript Frameworks Node.js

The main purpose of the framework is to help build interaction vigorous web apps like community sites, content streaming websites, feature heavy one-page apps, and other apps that rely on heavy data interaction.

It is a cornerstone of the Appcelerator Titanium mobile app development tool.

jQuery

jQuery Manipulate the DOM and CSS

The primary emphasis of the framework is the ability to use CSS selectors to select and work with DOM objects.

Like most frameworks you can download it from the offical jQuery project site (http://jquery.com). You can also use Googles free CDN.

It is available as both compressed (minimized) or uncompressed. Since you do not want to change the code of the framework in most cases the smaller minimized version would be preferable because of size.

Code along

jQuery part one

Code from the code along HTML

```html
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" type="text/css" href="1_1.css">
</head>
<body>

<h1>First heading</h1>

<p>First paragraph.</p>
<p>Second paragraph.</p>

<button>Make red</button>

</body>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<script type="text/javascript" src="1_1.js"></script>
</html>
```

HTML structure. Omit the jquery library first since we'll be using pure JavaScript first to see the difference.

We put the js at the bottom since we do not need it straight away. Here we put the js in a separate file.

Code from the code along JavaScript

```
document.getElementById("makered").onclick = changeColor;

function changeColor() {
    var x = document.getElementsByTagName("p");
    var i;
    for (i = 0; i < x.length; i++) {
        x[0].className += ' intro';
    }
}
```

Using pure javascript to change the color of the p elements.

We put it in a separate file so we do not need HTML tags.

Code from the code along jQuery

```
$(document).ready(function(){
    $("button").click(function(){
        $("p:first").addClass("intro");
    });
});
```

In JavaScript $ has no special significance (no more than a or Q anyway). It is just an uninformative variable name.
In jQuery the $ is an alias for the function called jQuery, so your code can be written like this with the exact same results: jQuery(document).ready(function() {...

A page can't be manipulated safely until the document is "ready." jQuery detects this state of readiness for you. Code included inside $(document).ready() will only run once the page Document Object Model (DOM) is ready for JavaScript code to execute.

addClass adds the specified class(es) to each element in the set of matched elements.

Using jQuery to change the color of the first p element by adding a CSS class to it.

We put it in a separate file so we do not need HTML tags.

Code from the code along

Removing a class is just as easy as adding it

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").removeClass("intro");
    });
});
```

Checking if an element has a specific class

```
$(document).ready(function(){
    $("button").click(function(){
        alert($("p").hasClass("outro"));
    });
});

```

Code from the code along JavaScript

```
document.getElementById("makered").onclick = changeColor;

function changeColor() {
    var x = document.getElementsByTagName("p");
    var i;
    for (i = 0; i < x.length; i++) {
        x[0].className += ' intro';
    }
}
```

Using pure javascript to change the color of the p elements.

We put it in a separate file so we do not need HTML tags.

Code along

Break

Code from the code along HTML

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
<link rel="stylesheet" type="text/css" href="2_1.css">
</head>
<body>

<h1>First heading</h1>

<p class="intro">First paragraph.</p>
<p>Second paragraph.</p>

<button>Change content of all p elements</button>

</body>
<script type="text/javascript" src="2_1.js"></script>
</html>
```

We haven't really changed anything since the first jQuery examples, so just copy that. Since we won't be manipulating CSS in these examples the file has been omitted.

Code from the code along jQuery

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").html("Hello <b>world!</b>");
    });
});
```

.html(htmlString) sets the HTML contents of each element in the set of matched elements.

Using jQuery to change the content of the p elements.

Code from the code along HTML

```html
<!DOCTYPE html>
<html>
<head>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
</head>
<body>
    <h1>First heading</h1>
    <p class="intro">First paragraph.</p>
    <p>Second paragraph.</p>
    <ol>
        <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 item</button>
</body>
<script type="text/javascript" src="2_2.js"></script>
</html>
```

We haven't really changed anything since the first jQuery examples, so just copy that. Since we won't be manipulating CSS in these examples the file has been omitted.

Code from the code along jQuery

```
$(document).ready(function(){
    $("#btn1").click(function(){
        $("p").append(" <b>Appended text</b>.");
    });
    $("#btn2").click(function(){
        $("ol").append("<li>Appended item</li>");
    });
});
```

.append(content) inserts content, specified by the parameter, to the end of each element.

```
$(document).ready(function(){
    $("#btn1").click(function(){
        $("p").prepend(" <b>Prepended text</b>.");
    });
    $("#btn2").click(function(){
        $("ol").prepend("<li>Prepended item</li>");
    });
});
```

.prepend(content) inserts content, specified by the parameter, to the beginning of each element.

Using jQuery to add to the content of the p and ol elements.

Code from the code along jQuery

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").after("<p>Hello world!</p>");
    });
});
```

.after(content) inserts content, specified by the parameter, after each element.

```
$(document).ready(function(){
    $("button").click(function(){
        $("p").before("<p>Hello world!</p>");
    });
});
```

.before(content) inserts content, specified by the parameter, before each element.

Using jQuery to add to content around the p elements.

Code along

Break

Code from the code along HTML and jQuery

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
</head>
<body>
    <h1>First heading</h1>
    <input type="text">
    <p>Write something in the input field, and then press enter or click outside the field.</p>
</body>
<script type="text/javascript" src="3_1.js"></script>
</html>
```

```javascript
$(document).ready(function(){
    $("input").change(function(){
        alert("The text has been changed.");
    });
});
```

.change () binds an event handler to the "change" JavaScript event, or triggers that event on an element.

Using jQuery to register changes.

Code from the code along HTML and jQuery

```
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
</head>
<body>
    <h1>First heading</h1>
    <p>Click on this paragraph.</p>
</body>
<script type="text/javascript" src="3_2.js"></script>
</html>
```

```
$(document).ready(function(){
    $("p").click(function(){
        alert("The paragraph was clicked.");
    });
});
```

.click () binds an event handler to the "click" JavaScript event, or triggers that event on an element. Technically inline binding (onclick) is considered depreciated, so event-listeners is the “proper” modern way to register events.

Using jQuery to activate on clicks.

Code from the code along HTML and jQuery

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
</head>
<body>
    <h1>First heading</h1>
    <p>Hover the mouse pointer over this paragraph.</p>
</body>
<script type="text/javascript" src="3_3.js"></script>
</html>
```

```javascript
$(document).ready(function(){
    $("p").hover(function(){
        $(this).css("background-color", "yellow");
        }, function(){
        $(this).css("background-color", "pink");
    });
});
```

.hover () binds one or two handlers to the matched elements, to be executed when the mouse pointer enters and leaves the elements.

Using jQuery to activate effect on hover.

Code from the code along HTML and jQuery

```html
<!DOCTYPE html>
<html>
<head>
<script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></script>
</head>
<body>
<h1>First heading</h1>
    Enter your name: <input type="text">
    <p>Enter your name in the input field above. It will change background color on keydown and keyup.</p>
</body>
<script type="text/javascript" src="3_4.js"></script>
</html>
```

```javascript
$(document).ready(function(){
    $("input").keydown(function(){
        $("input").css("background-color", "yellow");
    });
    $("input").keyup(function(){
        $("input").css("background-color", "pink");
    });
});
```

.input () selects all input, textarea, select and button elements.

Using jQuery to react on input.

Sources and further reading

- http://www.w3schools.com/js/js_htmldom.asp
- https://developer.mozilla.org/en-US/docs/Web/API/Document/createElement
- https://colorlib.com/wp/javascript-frameworks/
- https://www.sitepoint.com/top-javascript-frameworks-libraries-tools-use/
- http://iul.dk/2016/09/mvc/ (in Danish)
- https://learn.jquery.com/using-jquery-core/document-ready/
- https://api.jquery.com/addclass/
- http://api.jquery.com/html/
- http://api.jquery.com/append/
- http://api.jquery.com/prepend/
- http://api.jquery.com/after/
- http://api.jquery.com/before/
- http://api.jquery.com/change/
- http://api.jquery.com/click/
- http://stackoverflow.com/questions/12627443/jquery-click-vs-onclick (relevant debate)
- http://api.jquery.com/hover/
- http://api.jquery.com/input-selector/