12. Client Side Javascript & JQuery

Github Repository | Vimeo Video Recording

Topics to cover

- [X] Client side JS
  - [x] 1. Browser objects
  - [x] 2. Intro to the DOM
  - [x] 3. The document object
  - [x] 4. Event handling and the event object
- [X] 5. Intro to JQuery
  - [x] Why does it exist?
  - [x] How much additional behavior does it add to the browser?
  - [x] Library or framework? Why?
  - [x] Why is it important to learn / use jQuery?
  - [x] Element creation with jQuery
  - [x] Event handling with jQuery

1. Browser objects

In order to access and/or modify elements on your website (or even on the browser itself), modern browsers offer different objects we can access through JavaScript:

`window` object - Represents the window that holds the DOM and BOM objects - Each tab in a browser is a *window* with its own `window` object - Contains (among other things) information about the size of the window and screen

`navigator` object - Part of the BOM objects - Contains information about the browser such as browser version, browser name, and geographic location

`document` object - Represents the DOM - Can be seen in the browser console by running `console.dir(document)`

We call the objects directly, or as properties of the window object:

```
// you can also access navigator and document as props on the window
object
window.navigator === navigator; // true
window.document === document; // true
```

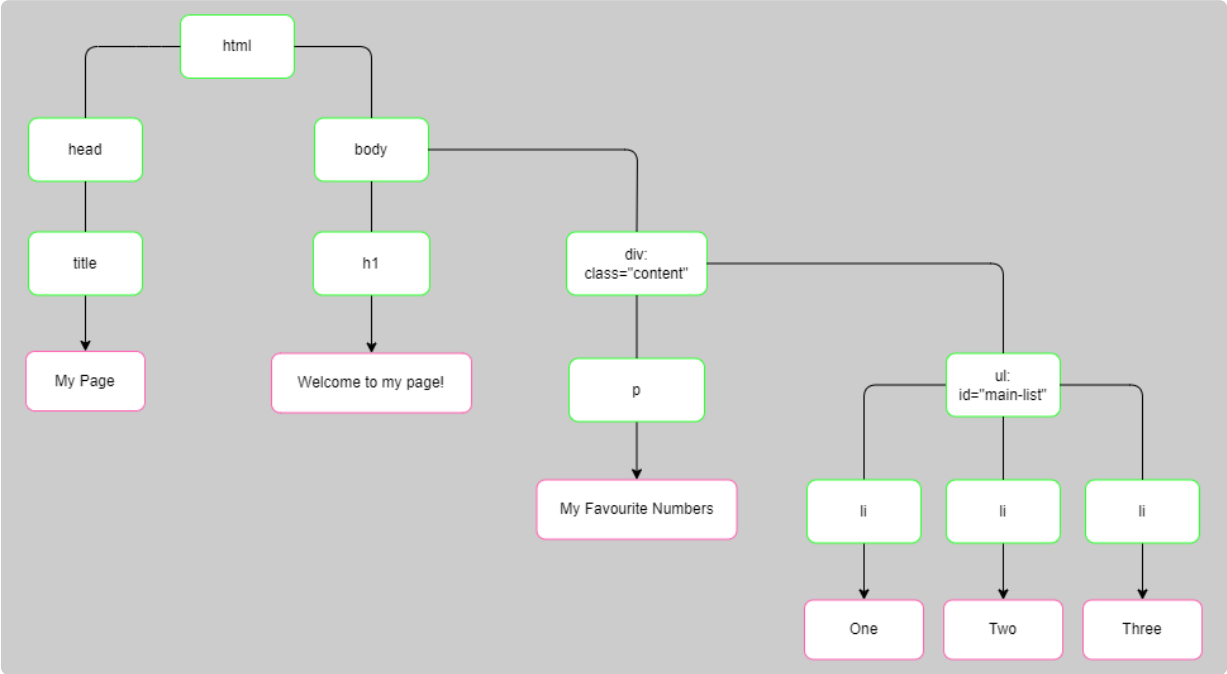
2. Intro to the DOM

From MDN:

> The Document Object Model (DOM) is a programming interface for HTML and XML documents. It represents the page so that programs can change the document structure, style, and content. The DOM represents the document as nodes and objects. That way, programming languages can connect to the page.
> Our HTML is turned into a data structure that we call the **D**ocument **O**bject **M**odel (**DOM**).

- The DOM is stored as a `tree` data structure with parent/child/sibling relationships between various parts of the page
- The DOM allows us to interact with the web page using JavaScript
- eg. The browser turns your HTML into the DOM

```
<html>
  <head>
    <title>My Page</title>
  </head>
  <body>
    <h1>Welcome to my page!</h1>
    <div class="content">
      <p>My Favourite Numbers</p>
      <ul id="main-list">
        <li>One</li>
        <li>Two</li>
        <li>Three</li>
      </ul>
    </div>
  </body>
</html>
```



Green outline = HTML element; Pink outline = text node

3. The `document` object

When an HTML document is loaded into a web browser, it becomes a `document` object, which: * acts as the root node of the HTML document. * is a property of the window object. * is accessed with: `window.document` or just `document`

[You can find a list of properties and methods of the `document` object, with explanations and examples on the w3school's website](#).

4. JavaScript Events

Events are "things" that get triggered when some action has occurred in the website (*eg. a button is clicked, the mouse pointer is moved, a key is pressed*). When JavaScript is used in HTML pages, **JavaScript can "react" on these events**.

Handling events

We can attach code (usually in the form of a `callback` function) to run when a specific event occurs... we call this **attaching an event listener to an element**:

```
// we can use anonymous functions...
document
  .querySelector('button')
  .addEventListener('click', (event) => {
  // do something
  });

// or named functions
const eventHandler = function (event) {
  // do something
};
document
  .querySelector('button')
  .addEventListener('click', eventHandler);
```

There are a *lot* of events. Each event is represented by an `Event` object which is passed as the argument to the callback function.

The event object

The `Event` object contains useful information about the specific event that occurred

```
// console.log the mouse x and y coordinates whenever the body is clicked
const clickHandler = function (event) {
  console.log(event.clientX, event.clientY);
};
document
  .querySelector('body')
  .addEventListener('click', clickHandler);

// we can also remove event handlers using a similar API
document
  .querySelector('body')
  .removeEventListener('click', clickHandler);
```

5. jQuery

---

jQuery is a JavaScript library that provides a simple API for DOM manipulation, event handling, and AJAX requests * Can be referenced with `jQuery` or the `$` * Typically brought into a project using a Content Delivery Network (**CDN**) * **CDN**s store code and other files that can be brought into our projects as the browser loads our page

```
<script src="https://code.jquery.com/jquery-3.4.1.js"></script>
```

You can also download the latest compressed-production-ready version on jQuery and add it to your project files.

jQuery Uses CSS selectors for finding elements in the DOM

```
// html element
$('h1');
// class
$('.my-class');
// id
$('#my-id');
// nested element
$('.my-class span');
```

Creating Elements with jQuery

We can use the same jQuery interface to create DOM elements by passing in the opening tag of an HTML element

```
const newDiv = $('<div>');
const newImg = $('<img>');

// note the greater than and less than
$('img') !== $('<img>');
```

We can add attributes, event listeners, and even child elements to our created elements and then append to somewhere in the DOM

```
// create a new image and give it a src attribute
const newImg = $('<img>').attr('src', '/path/to/image.png');

// append the new image to the body element
$('body').append(newImg);
```

Event Handling with jQuery

We can also easily attach event listeners to DOM events using jQuery

```
// using the .on method
$('button').on('click', clickHandler);

// there are several shorthand methods for common DOM events
$('button').click(clickHandler);
$('form').submit(submitHandler);
$('input').focus(focusHandler);
```

Document Ready

We usually want to wait for the document to finish being loaded before our JavaScript runs. jQuery gives us a simple interface for wrapping our code. Once the document has finished loading, our callback is called:

```
$(document).ready(() => {
  // this callback runs once the document is "ready"
  console.log('ready');
});

// or
$(() => {
  // passing a callback to jQuery is a shorthand for $(document).ready()
  console.log('ready');
});
```

All the JavaScript and jQuery code can be wrote inside the callback passed into the ready function. This way you can ensure that your document is loaded.

Useful Links

- MDN: The DOM
- W3C DOM Standard
- MDN: Browser Events
- What is the `$` in DevTools?
- jQuery Docs

View on Compass | Leave Feedback