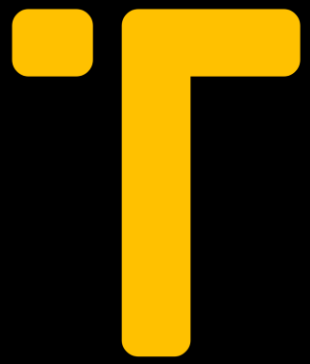

TECHSWITCH

JavaScript and the DOM

Softwire

JavaScript

You've all seen some JavaScript before, so I'm not going to explain that.

Today's talk is going to be about how to apply that to working in the browser!

The DOM

The DOM (or Document Object Model) represents the structure of HTML documents. JavaScript running in a browser is able to view the DOM, change it or add new content.

This allows us to use JavaScript to change the webpage in response to a user's actions.

Remember that in general, if we could do something without JS, then that's often going to be a good idea!

Things written in JS are likely to

- Be more complex
- Be more fragile
- Work for fewer people

TECHSWITCH
Softwire

Events

Every time ‘stuff happens’ to HTML elements, the browser emits ‘events’ which provide details about what has happened.

For example

- When the browser finishes loading a page, a ‘load’ event is fired.
- When a button is clicked, a ‘click’ event is fired.
- When a form is submitted, a ‘submit’ event is fired.

JavaScript can ‘listen’ to these events, and we can write code to respond to them.

Common DOM Methods

The document

JavaScript running in the browser will always have access to a global variable called document.

This variable represents the entirety of the HTML that the browser is currently rendering.
This is the object that allows us to interact with the elements on the page.

```
> document
◂ ▼#document
    <!doctype html>
    <html lang="en-GB" class="b-pw-1280 no-touch orb-js id-svg orb-more-loaded n-no-touchevents id-svg se-no-touch" id="responsive-news">
    ▶<head>…</head>
     <!-- front_page_domestic_personalised -->
     <!--[if lt IE 7]>      <body class="lt-ie10 lt-ie9 lt-ie8 lt-ie7">  <![endif]-->
     <!--[if IE 7]>         <body class="ie7 lt-ie10 lt-ie9 lt-ie8">     <![endif]-->
     <!--[if IE 8]>         <body class="ie8 lt-ie10 lt-ie9">            <![endif]-->
     <!--[if IE 9]>         <body class="ie9 lt-ie10">                   <![endif]-->
     <!--[if gt IE 9]><!-->
    ▶<body>…</body>
    </html>
> |
```

TECHSWITCH
Softwire

Finding Elements

We can find elements in the DOM using very similar techniques to what we saw yesterday with CSS.

The document has the following methods that allow us to find elements by the tags, classes or IDs defined in the HTML.

```
document.getElementById("my-id");
document.getElementsByClassName("my-class-name");
document.getElementsByTagName("h1");
```

Note that getting by ID returns 1 element (as IDs are unique), but the other methods return a collection of elements.

Reading Properties

Once we've found the element that we care about, we can inspect it to find its details.

```
const classes = element.className;
const position = element.getBoundingClientRect();
const fontSize = element.style.fontSize;
```

As with most things on the web, your browser's dev tools are a great place to experiment with how this works.

TECHSWITCH
Softwire

Changing an element

We can also update our element.

Effectively what's happening here is that JavaScript is editing the HTML document.
Therefore, anything that we could do in HTML we can also do in JS.

```
// I can add classes.
element.classList.add("my-new-class");

// Or directly set inline styles
element.style.fontSize = "24px";

// Or change any of the attributes that the element has
element.setAttribute("type", "text");
```

Listening to Events

We will probably also want to be able to respond to events.
There are 2 ways to do this.

1. **Add an event listener in JS**

```
const button = document.getElementById("my-button");
button.addEventListener("click", function() {
    console.log('I was clicked')
});
```

2. **Reference a function from the HTML**

```
function whenClicked() {
    console.log("I was clicked");
}
```

```
<button onclick="whenClicked()">Click Me</button>
```

TECHSWITCH
Softwire

Adding to the DOM

We can also generate entirely new elements to add to the document!

```
const myList = document.getElementById("my-list");

const newListItem = document.createElement("li");
const textContent = document.createTextNode("Hello TechSwitch");

newListItem.appendChild(textContent);
myList.appendChild(newListItem);
```

We can also delete things.

```
const element = document.getElementById("thing");
element.remove();
```

JQuery

Cross browser support has historically been a huge problem for JS.
It's still a problem today (why won't IE die???) but in general JS implementations are a lot more inconsistent.

We needed some way to be able to write code once and have some hope that it would work on most browsers.

For manipulating the DOM, the answer was a library called JQuery.

The idea was that you could just call the simple library methods, and then JQuery would have the responsibility of figuring out all the annoying cross-browser stuff.

This worked brilliantly.

As a result, JQuery became a default standard of web development.

JQuery

Today, with the improved state of JS, we often don't need JQuery (we'll never use it on this course).

For example, the basic DOM methods we've been looking at today should all 'JustWork' on all major browsers, so we don't need to use a library method to replace them.

But I still think it's worth mentioning, as you'll see it around quite often

- In older codebases
- (possible) in libraries that you depend on
- In stack overflow answers
- In tutorials.

Luckily, it's easy to spot. Any JS with a '$' is using JQuery.

```
$('#myId').hide();
$(element).onClick(function() {
    // stuff
})
```