# JavaScript

When the internet was taking off in the mid-90s it was still running over very slow connections. Every time a page was loaded it was necessary to re-send all of the data which could be incredibly inefficient. A solution was found which made it possible to do some of the updates on the user's end, removing some of the need to reload pages.

That solution developed into what we know today as **JavaScript**. Every modern web browser includes a JavaScript **engine**, enabling us to run code inside our browsers which will affect the websites we view. JavaScript has evolved to become something far more powerful, with engines such as NodeJS developed to enable us to run JavaScript outside the browser. There are many frameworks which make it easier for users to develop websites and make JavaScript a core part of them.

JavaScript in the browser is *event-driven*, meaning the code is executed in response to the user's actions. We will implement it here by adding a button to our page which, when clicked, will put the site into "dark mode" by modifying the CSS to change the colours of the background and the text.

## Importing JavaScript

We don't write JavaScript code directly in the HTML, instead we import it into the file in a similar way to the CSS. In the start point for this session we have a file called `app.js` which we import in the `head` section:

```
<head>

    <!-- Other metadata goes here -->

    <script defer src="app.js"></script>

</head>
```

The `defer` keyword tells the browser to load the file but to pause the execution of it until the content in the `body` has loaded. The code we are going to write in `app.js` will be acting on the elements defined in `body`, but JavaScript code runs *very* quickly - much quicker than a browser can load HTML. If we don't tell the browser to wait we'll run into errors as we try to modify something that doesn't exist yet. Alternatively we could place the

`script` tag at the bottom of the `body` to ensure the HTML loads first, or we could add a line to the JavaScript file to pause the execution there.

# Adding the Button

Our JavaScript file is going to do three things:

1. Identify the HTML elements we want to interact with
2. Add an **event listener** to each of them
3. Define a function which should be executed when the event is triggered

Before we can tell our JavaScript to look for a dark mode button we'll need to add it to the HTML. We'll put it in the header underneath the nav bar.

```
<header>
    <!-- title and nav bar -->
    <button id="dark-mode-button">Dark Mode</button>
</header>
```

We've given the button an `id` attribute but we're not applying any CSS. By setting this property we'll be able to pick it out using JavaScript, which we do using a function called `getElementById`.

```
darkModeButton = document.getElementById("dark-mode-button")
```

There are a few new things happening here which we'll go through one by one. Firstly let's look at the `document` keyword. When we run JavaScript in the browser we need some way to "hook in" to the HTML being rendered and the JavaScript engine allows us to do this using `document`. The `getElementById` function lets us pick out a particular piece of HTML according to an `id` which we specify, in this case the button we just added. Finally we store the element in a **variable** which enables us to refer back to it later in the code.

We're going to click the button and expect something to happen and we need to define what that "something" should be. We'll write a **function** which will specify the behaviour we want to see when the button is clicked. Functions are reusable chunks of code, meaning we could add this behaviour to other buttons as well without having to rewrite everything. We won't start modifying the CSS just yet, for now we'll print a message to let us see that everything's working correctly.

```
handleButtonClick = function (){
    console.log("button clicked")
}
```

Lastly we need to make the connection between our button and this behaviour. Remember the variable we used earlier to store the reference to the button in? We'll apply a special function called an **event listener** to it to make that link. An event listener needs two pieces of information: the type of event we're listening out for and what to do when that event occurs.

```
darkModeButton.addEventListener("click", handleButtonClick)
```

**Events** are just what they sound like - things that happen when we interact with our site. There are many different events which we can respond to. Some, such as mouse movement, are happening all the time as a natural by-product of using the site. Others only "fire" on specific interactions, some only when those interactions happen on specific elements. In this case we are listening for a "click" event, ie. the user clicking the left mouse button when the cursor is over the selected element. When that click happens the code we wrote in the `handleButtonClick` function will be executed.

We can test it out now! The message we wrote will be displayed in the browser's *console* which we can access by right-clicking on the page, selecting "inspect", then clicking the "console" tab at the top of the panel which opens.

# Modifying the Page

We can do a *lot* more than just print things to the console with JavaScript. We'll take things a step further here by using it to modify what's displayed on the page, specifically the CSS. We can do even more than that if we want to: we could add or remove elements, run complex background tasks or even interact with another program if we put the right tools in place.

We'll start by changing the background colour of the page's `body`. Accessing this is much easier than finding our button as it's a built-in property of the `document`. The body has a `style` property (as does every element) which has further properties describing each of the CSS attributes we may want to change. If we do want to change anything we can simply assign a new value to it.

```
handleButtonClick = function (){
    body = document.body
    body.style.backgroundColor = "#091d1e"
}
```

Now when we click the button our page changes colour! We can't change it back, but we're not stuck with it. Any changes we make are not persistant, meaning that when we click the refresh button in the browser the page will reload and undo all of our changes. Changing the background colour has made the text pretty hard to read, so let's change that too.

```
handleButtonClick = function (){
    body = document.body
    body.style.backgroundColor = "#091d1e"
    body.style.color = "#aaaaaa"
}
```

That's better but it's still not perfect. Our black borders are really difficult to see against the darker background so we need to change them. We'll start with the title section and we can use our old friend `getElementById` from before to help us here.

```
handleButtonClick = function (){
    // body changes

    title = document.getElementById("title-section")
    title.style.border = "1px solid #aaaaaa"
}
```

That's one of them taken care of, but we still need to deal with the borders around the posts. This will be a little more complicated as we have to deal with three elements instead of one. There are tools available to help us though, one of which is the `getElementsByClassName` function. Just like `getElementById` it searches for a particular CSS selector among the HTML, only this time it gets *every* element which matches.

```
handleButtonClick = function (){
    // body changes

    // title changes

    posts = document.getElementsByClassName("post")
}
```

We can't just modify the `border` property and hope that all of the elements pick up the change, but we can use a structure known as a **for loop** to look at each element in turn and modify the properties individually.

```
handleButtonClick = function (){
    // body changes

    // title changes

    posts = document.getElementsByClassName("post")
    for (post of posts){
        post.style.border = "1px solid #aaaaaa"
    }
}
```

Now each post has the correct colour on its border. We've lost the colour differential we had with the post titles, but we can get it back. They have neither a class nor ids so we can't use the functions we've seen so far, but we can use the `getElementsByTagName` function to search for all elements of a given type. Once we've found them we can use a for loop again to set the `color` property for each one.

```
handleButtonClick = function (){
    // body changes

    // title changes

    // post borders

    postHeadings = document.getElementsByTagName("h3")
    for (heading of postHeading){
        heading.style.color = "#466876"
    }
}
```

Now our site can switch into dark mode at the press of a button!

# Building on the Basics

Once we know how to write event-driven programs like this we're on the way to building some seriously impressive applications. We've got the tools we need to build more sites like this where some information is displayed without being loaded from anywhere else (known as *static* sites), the next step could be to add a back end with a database where we can keep track of which posts are our favourites. We could add a form to let the user add more posts, we could add our missing pages from the nav bar, or we could start researching JavaScript frameworks and re-do the whole thing using one of them!