WELCOME TO CFG
YOUR INTRODUCTION TO JAVASCRIPT

CODE FIRST GIRLS

TECH SHOULDN'T JUST BE A BOYS CLUB.

COURSE JOURNEY

INTRO JAVASCRIPT	CONDITIONS & LOGIC	THE DOM	INTRO REACT	REACT COMPONENTS	STATES & EVENTS	STYLING COMPONENTS	PROJECT PRESENTATION
MODULE 01	MODULE 02	MODULE 03	MODULE 04	MODULE 05	MODULE 06	MODULE 07	MODULE 08

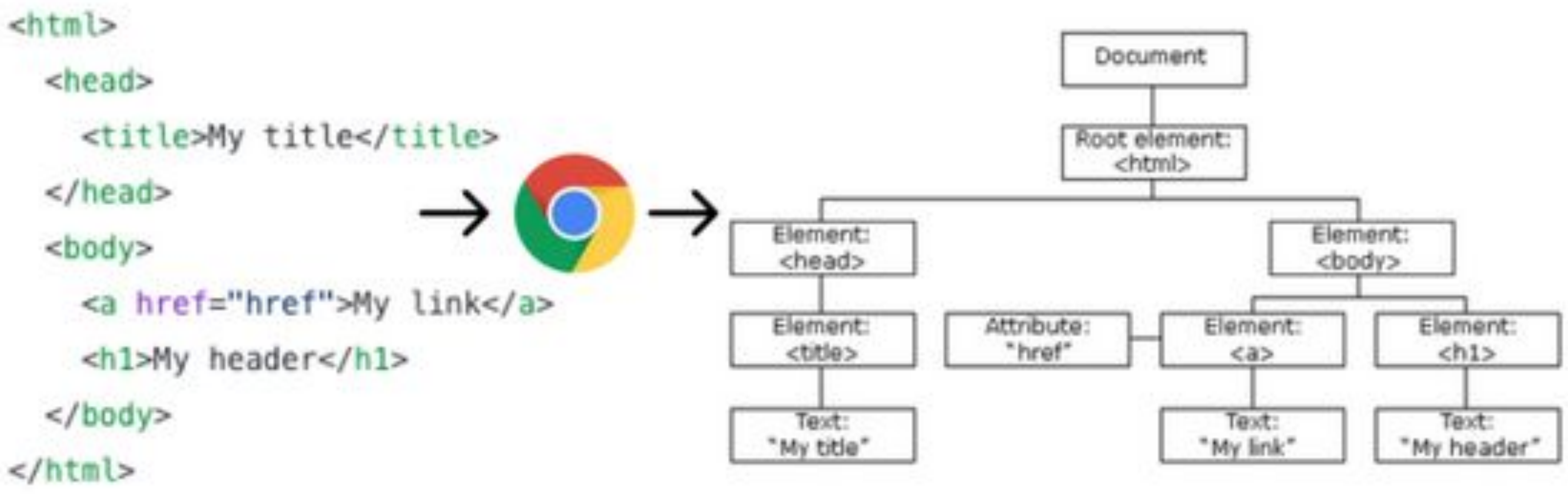
DOM MANIPULATION WITH JAVASCRIPT

One of the most exciting things about learning Javascript is something called DOM Manipulation. The HTML DOM is the **Document Object Model**, the interface created when a webpage first loads that contains all of the information relevant to the construction of that page.

Through this model, Javascript gains the power to alter, or manipulate elements, styles and information. **The DOM is a standard format for how we can get, change, add or delete elements.**

This also includes creating events, for example, adding interaction whenever a button is clicked, animating an element or storing data given to us by a user. **In short, it brings our website to life!**

WHAT IS THE DOCUMENT OBJECT MODEL?



```
<html>
  <head>
    <title>My title</title>
  </head>
  <body>
    <a href="href">My link</a>
    <h1>My header</h1>
  </body>
</html>
```

ACCESSING THE DOM

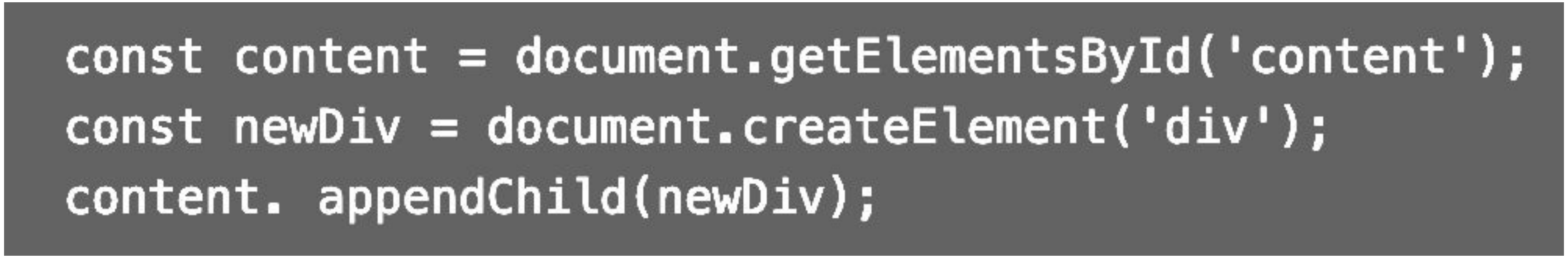
Accessing DOM elements is the first step in manipulating them using JavaScript. There are several different ways of accessing DOM elements, with different advantages and disadvantages for each. A couple of examples are:

- **getElementById( )**
  This is one of the most commonly used methods. It allows you to access an element by its unique id attribute. For example, if you have an html element called 'myId' you can access it using let element = **document.getElementById("myId")**

- **querySelector( ) and querySelectorAll( )**
  Query Selectors are similar to getElementById, but use CSS-style selectors to access elements. For example, if you want to access the first <p> element with a class of "myClass", you can use let paragraph = **document.querySelector("p.myClass")**

ACCESSING THE DOM

We then save these elements in our JavaScript code, so that we can refer to them and access them later on. In the example below, we have saved the html object with the ID 'content' in a variable with the same name. Then, we've created a new div and added it to the content element. This is an example of using DOM manipulation to create new html elements within your code.



```
const content = document.getElementsById('content');
const newDiv = document.createElement('div');
content. appendChild(newDiv);
```

NOW LET'S PRACTICE TOGETHER
UNDERSTANDING THE DOM

2 MINS

Exercise 1.0
Let's revise the whole process! Create an HTML file and a JS file and link them to each other!

7 MINS

Exercise 1.1
Using ONLY Javascript, create a paragraph ('p') called **paragraph** with the following qualities and attach to the DOM:

1. The inner text should say something about you
2. The font size should be 18px
3. The font family should be sans-serif
4. The width of the element should be 100px
5. The border should be 1px thick, solid and orange
6. The padding should be 30px

Feel free to use this CSS Cheatsheet and work out how to add these changes!

GROUP EXERCISE

WHAT ELSE CAN WE DO WITH THE DOM?

Once we understand that the Document Object Model is like a tree of objects, we can start to consider how to change this tree structure. Just like in real life, the DOM tree can react to changes. Some of the most common ways we can change our DOM is through the following:

- **Change all the HTML attributes and elements**
- **Create new HTML elements and attributes**
- **Remove HTML elements and attributes**
- **Change the CSS styling of elements**
- **React to events on the page**

MANIPULATING DOM ELEMENTS

```
 document.getElementById(“blue-circle”).style.color = “red”;


document.getElementById(“message-box”).innerHTML = “Added

text”;
    document.getElementById(“pop-up”).style.visibility =

    "hidden";
```

NOW LET'S PRACTICE TOGETHER

UNDERSTANDING THE DOM

2 MINS

Exercise 1.3
Let's take this a step further. Add a div in the index file and give it the ID **content**

5 MINS

Exercise 1.4
Using ONLY Javascript, create a header ('h1') called **newHeader** and place it **INSIDE of the div we created in 1.3** with the following qualities and attach to the DOM:

1. The inner text should say "I'm in!"
2. The border should be 1px thick, dashed and blue
3. The padding should be 50px

Feel free to use this CSS Cheatsheet and work out how to add these changes!

You will notice that the div appears before the paragraph we created before. That is simply because the JS file link is added to the end of our HTML and the DIV is added before it!

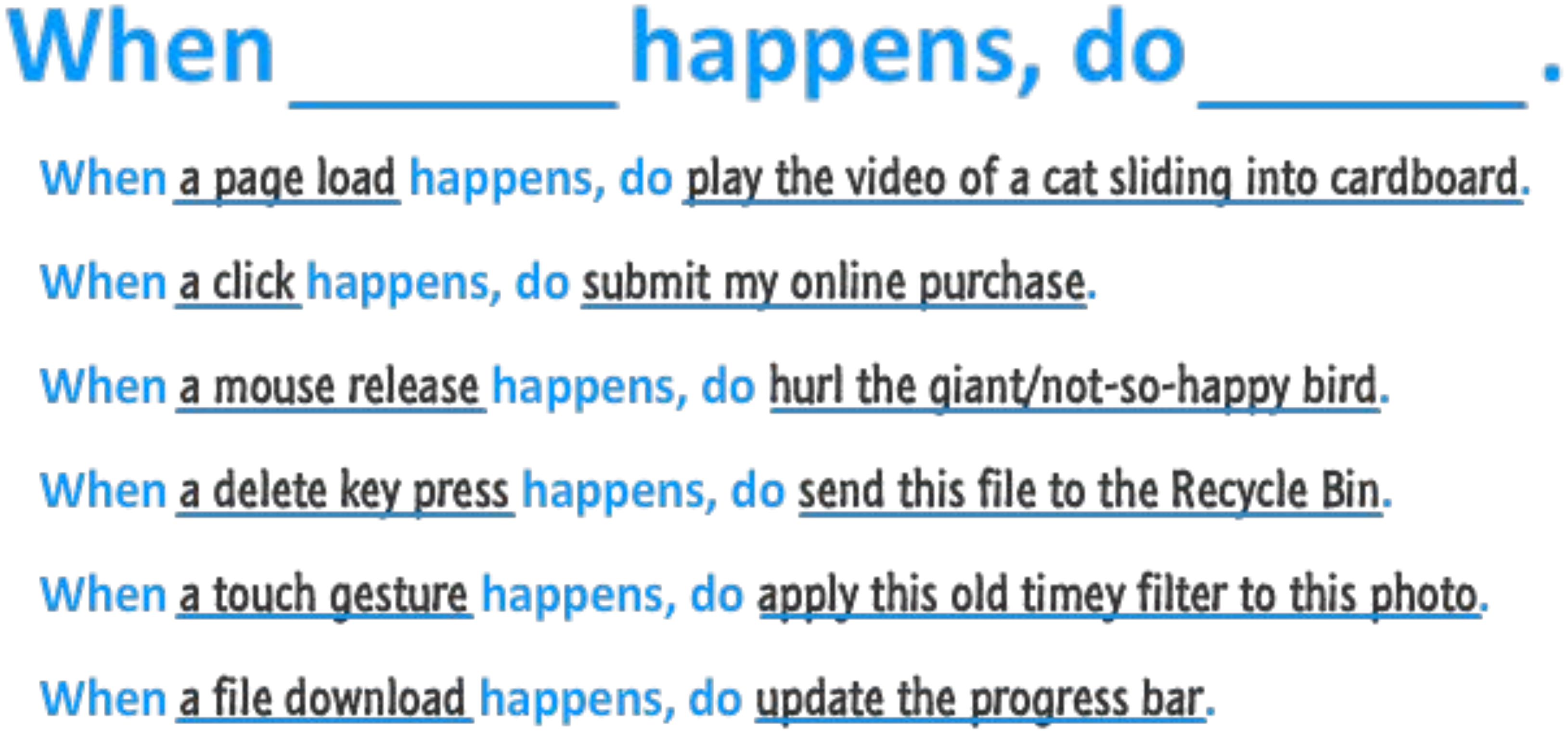
GROUP EXERCISE

DOM EVENTS

DOM events are actions or occurrences that happen in the document, such as a button being clicked or an element hovered over. These **events can trigger bits of JavaScript code**, which then in turn **change the structure of our DOM**. This lets us create a level of interaction between the user and the interface - for example, sending information to a database, changing an image or adapting the style of certain elements.

Common examples of **DOM events include clicks, mouseover movements, submit and load actions as well as key presses**. These will be some of the most used interactions you encounter when developing your websites.

DOM Events



When ______ happens, do ______.

When a page load happens, do play the video of a cat sliding into cardboard.

When a click happens, do submit my online purchase.

When a mouse release happens, do hurl the giant/not-so-happy bird.

When a delete key press happens, do send this file to the Recycle Bin.

When a touch gesture happens, do apply this old timey filter to this photo.

When a file download happens, do update the progress bar.

Event Listeners

Event listeners are an important part of of DOM manipulation. They're exactly what the name suggests - lines of code we create to listen for certain events to happen, and then trigger reactions. There is a two part process for creating an event listener:

- We need to specify the element we're watching - in this case, called **"ourElement"**:

  We do this by using the **document.getElementById("ourElement")**

- We add our event listener, and specify the kind of event we're looking for, as well as what we want that to trigger:

  .addEventListener("click", displayTime);

Here, we are activating the displayTime function when this is clicked. Put that all together:

document.getElementById("ourElement").addEventListener("click", displayTime);

NOW LET'S PRACTICE TOGETHER

DEFINING DOM ELEMENTS

5 MINS

Exercise 2.0
Open our HTML file called index-3 in the session-3-exercise-2 folder. You'll see that it has a button and paragraph element.

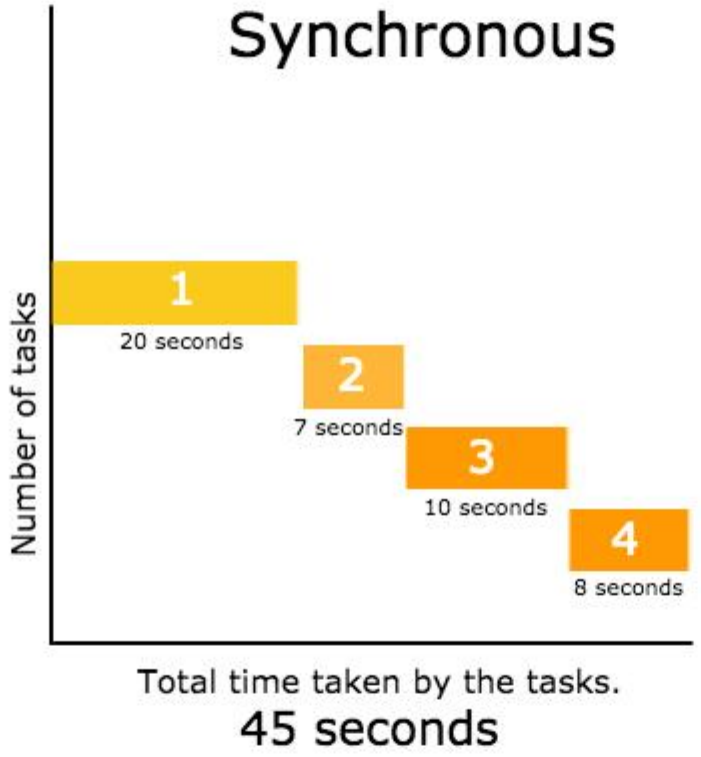
Exercise 2.1
We're going to select the button and paragraph elements using the **document.getElementbyId( )** method.

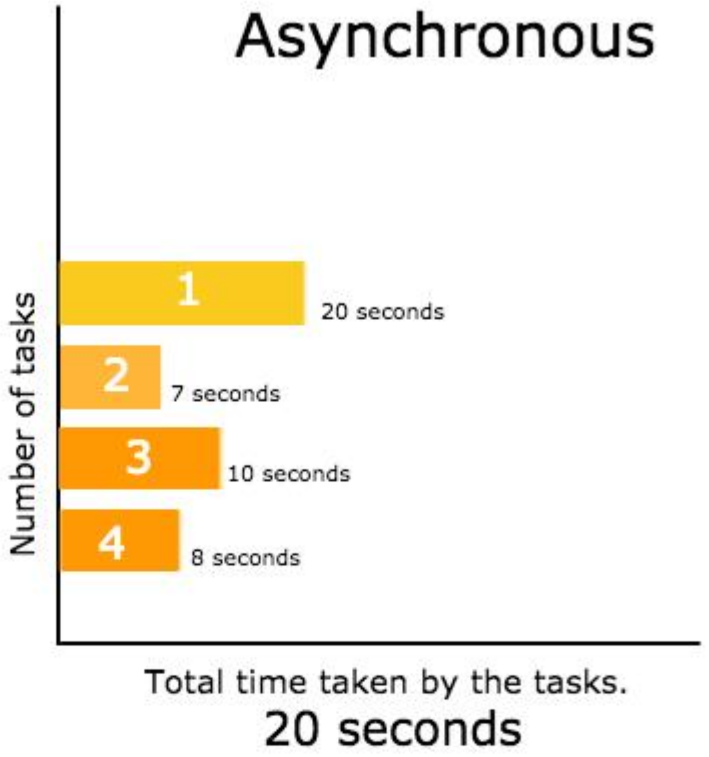
5 MINS

Exercise 2.2
We're going to add an event listener to the button that changes the text of the paragraph element when clicked.

Exercise 2.3
Open the file in a web browser and click the button. Watch as the text changes!

Asynchronous vs Synchronous JavaScript





Asynchronous vs Synchronous JavaScript

The concepts of synchronous and asynchronous code can be challenging to understand at first. However, as developers progress in their learning journey and begin to explore more advanced topics, such as **building web applications with libraries like React**, they will inevitably encounter asynchronous code. This is because modern **web applications often rely on asynchronous operations**, such as fetching data from a server or performing complex calculations in the background.

While the concept of asynchronous code may seem daunting at first, it's important to remember that **most modern JavaScript libraries and frameworks provide abstractions that simplify the process of working with asynchronous code**. For example, React provides a **built-in useState hook** that makes it easy to manage component state, even when state updates are triggered asynchronously.

HOMEWORK

+ **Homework Task – Customising using the DOM**

In the session-3-homework folder, you'll find starter HTML and CSS files. Using your new found DOM knowledge, create a JS file and link it to the current folder and manipulate the DOM customise Mabel's website!