WELCOME TO CFG
YOUR INTRODUCTION TO WEB DEVELOPMENT

CODE FIRST GIRLS

TECH SHOULDN'T JUST BE A BOYS CLUB.

COURSE JOURNEY

HTML	CSS	Recap Project design	JavaScript + Objects & the DOM	Github pages Frameworks	Project presentations Careers in web development
MODULE 01	MODULE 02	MODULE 03	MODULE 04	MODULE 05	MODULE 06

PART 1: OBJECTS

OBJECTS

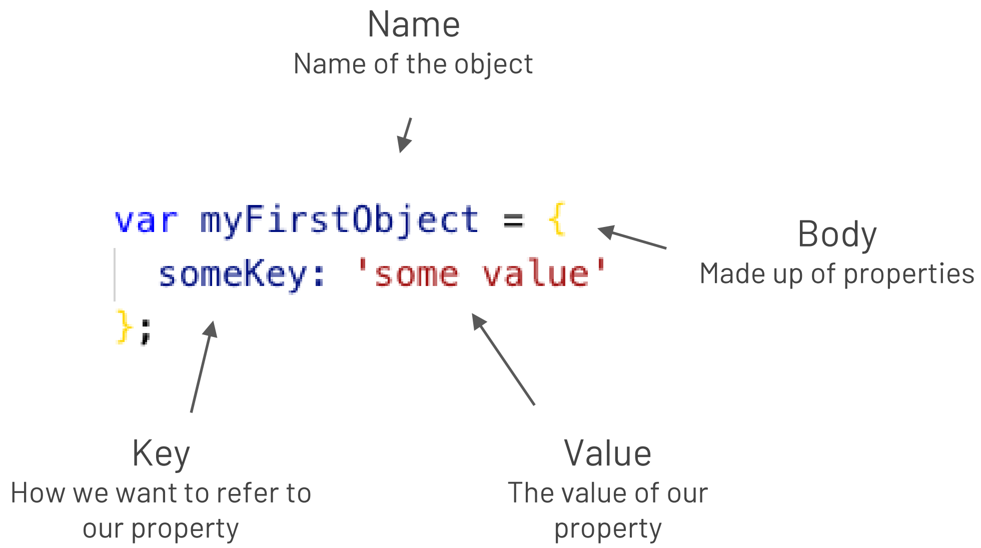
Similar to arrays, except instead of using numbers and square brackets […], **values** are assigned to **keys** within **curly brackets** {…}

Powerful way of storing information

This type of object is called an **object literal**

Syntactically very similar to CSS declarations

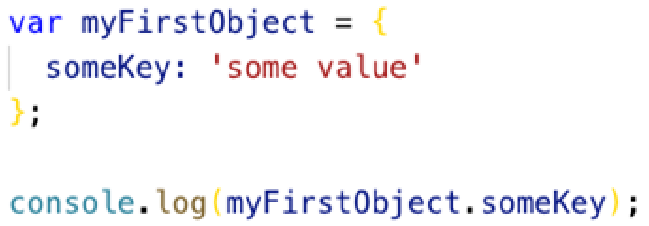
Objects have **properties** which are **key-value** pairs


Name
Name of the object
Body
Made up of properties
Key
How we want to refer to our property
Value
The value of our property

```
var myFirstObject = {
  someKey: 'some value'
};
```

ACCESSING VALUES

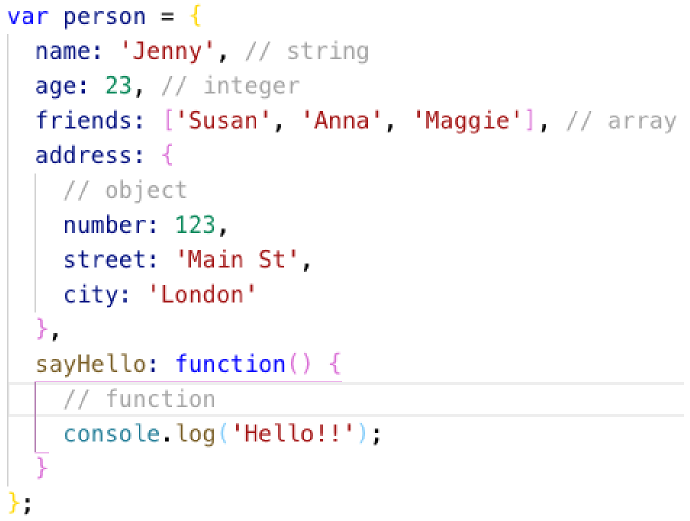
We can access the values within our objects using the "Dot" notation - object.key



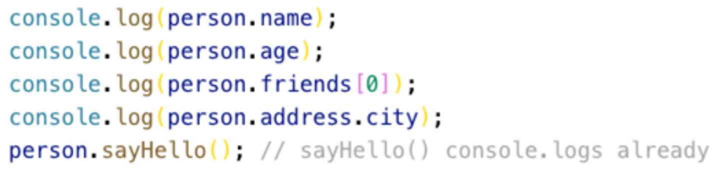
```
var myFirstObject = {
  someKey: 'some value'
};

console.log(myFirstObject.someKey);
```

ANOTHER TYPE OF OBJECT..

- This looks complicated but it’s not
- Think of it as a group of variables that belong to one thing
- **Value** can be anything (string, number, array, object, function etc)
- We treat the end values the same way we would normally, but since they’re now a property we have to access them **through** the object
- For example, we know friends is an array, but since its now a property we access it with **person.friends** before adding the square brackets
- When a **value** is a **function** that does something - it’s called a **method**



```
var person = {
  name: 'Jenny', // string
  age: 23, // integer
  friends: ['Susan', 'Anna', 'Maggie'], // array
  address: {
    // object
    number: 123,
    street: 'Main St',
    city: 'London'
  },
  sayHello: function() {
    // function
    console.log('Hello!!');
  }
};
```



```
console.log(person.name);
console.log(person.age);
console.log(person.friends[0]);
console.log(person.address.city);
person.sayHello(); // sayHello() console.logs already
```

PRACTICE

YOU CAN WORK ON YOUR OWN OR IN YOUR TEAMS

Exercise 6.1

Pick another topic (Show, Car, House, Movie, Book etc) and create another object that uses all the same data types as the previous example

OBJECTS RECAP

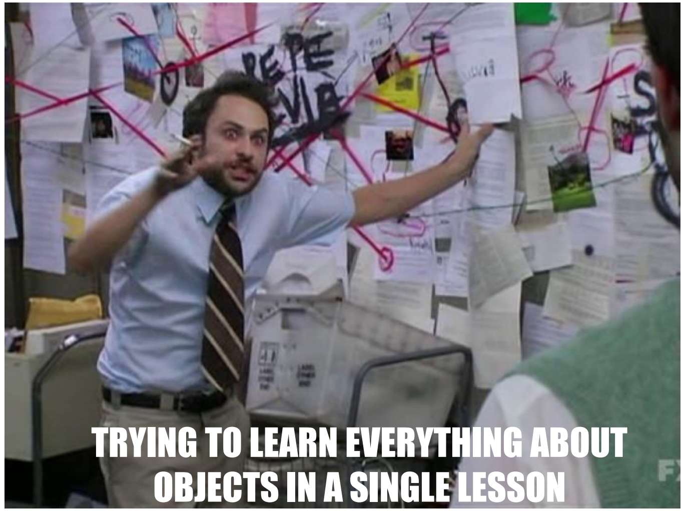
+ There is much, much more to Objects, and they play an integral part of coding across every language

+ An object is declared using curly brackets and is made up of properties

+ Properties are made up of **key:value** pairs

+ Values can be anything except empty

+ Values are accessed with: Dot notation eg:**person.age**



PART 2: THE DOM

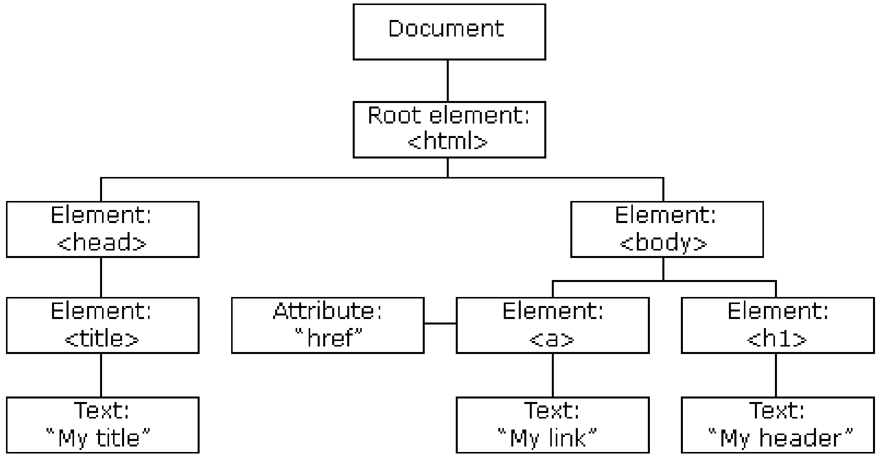
DOCUMENT OBJECT MODEL (DOM)

Under the hood, browsers treat our code as one enormous object

The **window** object is the topmost level

One of the window's main properties is the document, which is what we will focus on

It is through these properties that we can use Javascript to interact and modify our HTML and CSS



LOOKING AT THE DOM

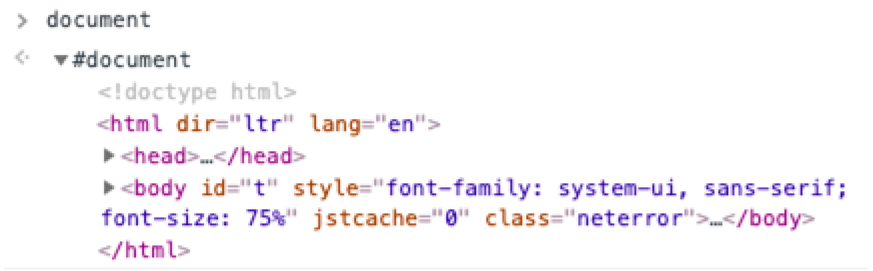
Open your Chrome dev tools

In the console, simply type **document**

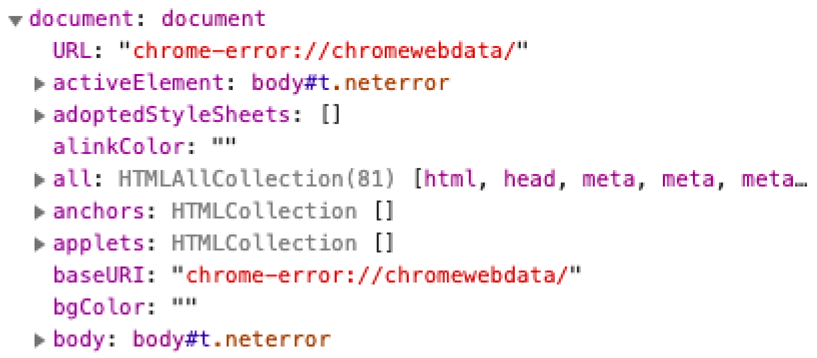
This will return you the **document node** where our HTML lives

However, to see what's behind the scenes, type **window** into the console then scroll down until you find the **document** key

Inside **document** you will see all of the properties associated with the document which we have access to via JavaScript!



```
> document
<· ▼#document
      <!doctype html>
      <html dir="ltr" lang="en">
       ▶<head>…</head>
       ▶<body id="t" style="font-family: system-ui, sans-serif;
      font-size: 75%" jstcache="0" class="neterror">…</body>
      </html>
```

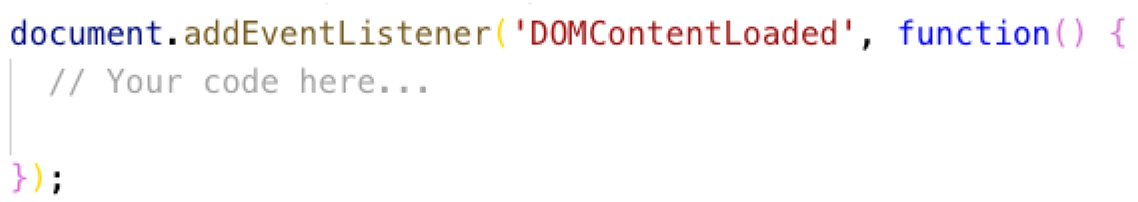


```
▼document: document
   URL: "chrome-error://chromewebdata/"
  ▶activeElement: body#t.neterror
  ▶adoptedStyleSheets: []
   alinkColor: ""
  ▶all: HTMLAllCollection(81) [html, head, meta, meta, meta…
  ▶anchors: HTMLCollection []
  ▶applets: HTMLCollection []
   baseURI: "chrome-error://chromewebdata/"
   bgColor: ""
  ▶body: body#t.neterror
```

GETTING STARTED

Open up **theDOM.js**

Because the browser reads HTML from top to bottom (head before body), we have to tell our Javascript to wait until all elements in the body have finished loading before we run it



```
document.addEventListener('DOMContentLoaded', function() {
  // Your code here...

});
```

GETTING ELEMENTS

We can get elements in a number of ways:

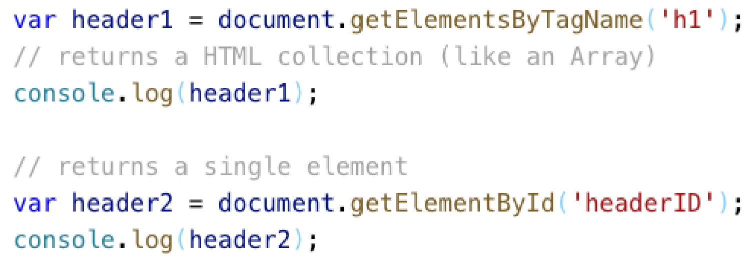
document.getElementsByTagName('h1')

document.getElementsByClassName('someClassName')

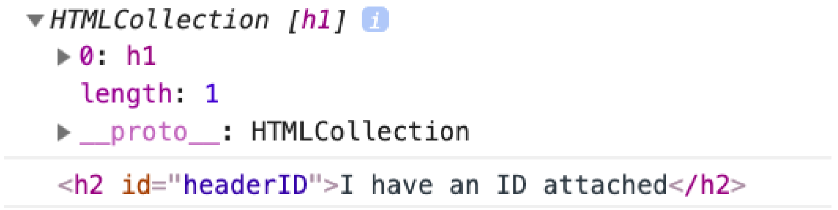
document.getElementsByName('h1')

document.getElementById('someID')

Open the first item in the HTMLCollection and you will see all the properties of the element



```
var header1 = document.getElementsByTagName('h1');
// returns a HTML collection (like an Array)
console.log(header1);

// returns a single element
var header2 = document.getElementById('headerID');
console.log(header2);
```



```
▼HTMLCollection [h1]
  ▶0: h1
   length: 1
  ▶__proto__: HTMLCollection
  <h2 id="headerID">I have an ID attached</h2>
```

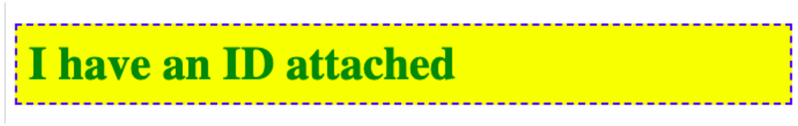
THE DOM – CHANGING CSS

Once we have a single element from the DOM, we can access and edit the **style** property

```
// Because it's a collection, we have to use [] to get the individual element
header1[0].style.color = 'blue';
// ID returns a single element so we can access style directly
header2.style.color = 'green';
```

We can change almost every CSS property in this way

```
header2.style.fontSize = '40px';
header2.style.background = 'yellow';
header2.style.padding = '10px';
header2.style.border = '2px dashed blue';
```



I have an ID attached

CREATING ELEMENTS

Use the createElement method to create any new element you want

Use innerText (or innerHTML) to give it some content

Append it to the body (or another element)

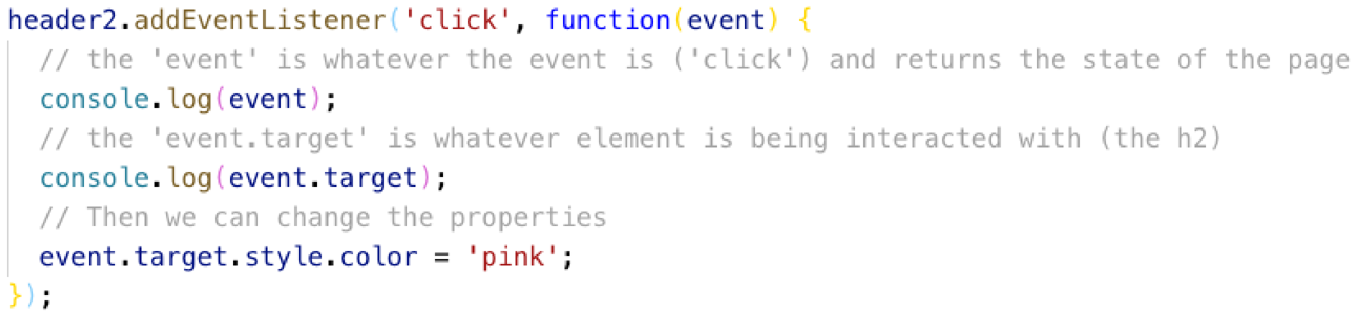
```
// use the createElement method
var newParagraph = document.createElement('p');
// add some text
newParagraph.innerText = 'I have just been created with Javascript!';
// append to the body
document.body.appendChild(newParagraph);
```

EVENTS

Events (like click, hover, drag, submit etc) are a cornerstone of front-end development.

An event has 3 parts:

The method - **.addEventListener**
The name - **'click'**, **'submit'** etc
The function - the code to execute: the **event** argument is the element you're interacting with



```
header2.addEventListener('click', function(event) {
  // the 'event' is whatever the event is ('click') and returns the state of the page
  console.log(event);
  // the 'event.target' is whatever element is being interacted with (the h2)
  console.log(event.target);
  // Then we can change the properties
  event.target.style.color = 'pink';
});
```

PRACTICE

YOU CAN WORK ON YOUR OWN OR IN YOUR TEAMS

Exercise 6.2

Using ONLY Javascript, create another paragraph (‘p’) called **paragraph2** with the following qualities and attach to the DOM:

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

PRACTICE

YOU CAN WORK ON YOUR OWN OR IN YOUR TEAMS

Exercise 6.3

In the paragraph 2 you created earlier, add an event listener for:

A mouse enter event where the font color changes to a color of your choice

PRACTICE

PLEASE WORK IN YOUR PROJECT TEAMS

Exercise 6.4

+ Find the folder: exercise-starter-code in Slack
+ Download the code, unzip and move it to a similar location
+ Open the entire folder (not individual files) in VScode
+ Go through the code and follow along with the instructions in app.js

HOMEWORK

+ Review your project ideas and determine what DOM events you think your page might need

If it has a form / newsletter, pay particular attention to object literals and the 'submit' event

THANK YOU

COME READY TO WORK ON YOUR PROJECTS IN THE NEXT SESSION

CODE FIRST GIRLS