CLASSES

WE WILL COVER THE FOLLOWING TOPICS:

- What are Classes?
- Why should we use them?
- New words
- What does a class look like?
- Naming conventions
- Methods
- The Constructor method
- Static methods

WHAT ARE CLASSES?

Classes are...

...a concept in OOP (object oriented programming)

...a template or blueprint for an object

...quite hard to get your head around the first time!

WHY SHOULD WE USE ES6 CLASSES?

Re-usability

Classes are designed to be re-used

A class is essentially a template - we can't usually use it directly. We must create a copy (instance). We work with the copy.

Helps us organise our code

A class essentially is a group of functions (methods) and properties

This group shares something in common - the Class

Think of it like a folder. You would only include in that folder things that belong together.

Just one more thing...
Classes (in JavaScript) were added in ES6
Before we used **prototypes** (not covered in this course)
In JavaScript, Classes are basically built on top of **prototypes**


RESTRICTED
AREA
NO UNAUTHORIZED
NO
TRESPASSIN

NEW WORDS WE MUST LEARN

instantiate (verb) - to make a copy of something
instance (noun) - refers to the copy
method (noun) - a special function which is attached to an object. Describes some behaviour on that object.

WHAT DOES A CLASS LOOK LIKE IN JAVASCRIPT?

```
class Animal {
}
```

Classes are just like objects - we can store methods and properties on them
In fact, we could represent our classes as objects
However classes give us a few extra advantages (more on this later)

```
class Animal {

    roar() {
        console.log("Hear me roar!")
    }

}
```

We don't use classes directly, we must instantiate (copy) them
We copy using the **new** keyword

```
class Animal {

    roar() {
        console.log("Hear me roar!")
    }

}

const dog = new Animal();
```

```
class Animal {

    roar() {
        console.log("Hear me roar!")
    }

}

const dog = new Animal();
const cat = new Animal();
const horse = new Animal();
```

NAMING CONVENTIONS WHEN WRITING A CLASS

Class names should be capitalised!

`Animal {}` ✔

not

`animal {}` ❌

Instances should NOT be capitalised!

`const dog = new Animal()` ✔

not

`const Dog = new Animal()` ❌

METHODS

A class consists of methods and properties
Methods are essentially just functions. We call them methods because they belong to an object (the class).

THE CONSTRUCTOR METHOD

Classes can include a special method called the **constructor**

This method is special, because it runs automatically when the class is instantiated

```
class Animal {

    constructor() {
        console.log("I am being
instantiated");
    }
}

const dog = new Animal();
```

We can also use the constructor to set properties
We must use **this** to refer to itself
Constructor is optional - only use it if you have to!

```
class Animal {

    constructor(props) {
        this.species = "Dog";
        console.log("I am being
instantiated");
    }
}

const dog = new Animal();
```

We can also pass in properties via the **constructor**
Remember the **constructor** is basically just a function

```
class Animal {

    constructor(props) {
        this.species = props.species;
        console.log("I am being
instantiated");
    }
}

const dog = new Animal({ species: "Dog"
});
const cat = new Animal({ species: "Cat"
});

console.log(dog.species); // "Dog"
console.log(cat.species); // "Cat"
```

Let's add a method now

```
class Animal {

    constructor(noise) {
        this.noise = noise;
    }

    playNoise() {
        console.log(this.noise);
    }
}

const dog = new Animal("woof!");

dog.playNoise(); // "woof!"
```