CLASSES

Classes are...

Classes are...

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

Classes are...

- ...a concept in OOP (object oriented programming)
- ...a template or blueprint for an object

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!

Just one more thing...

Just one more thing...

Classes (in JavaScript) were added in ES6

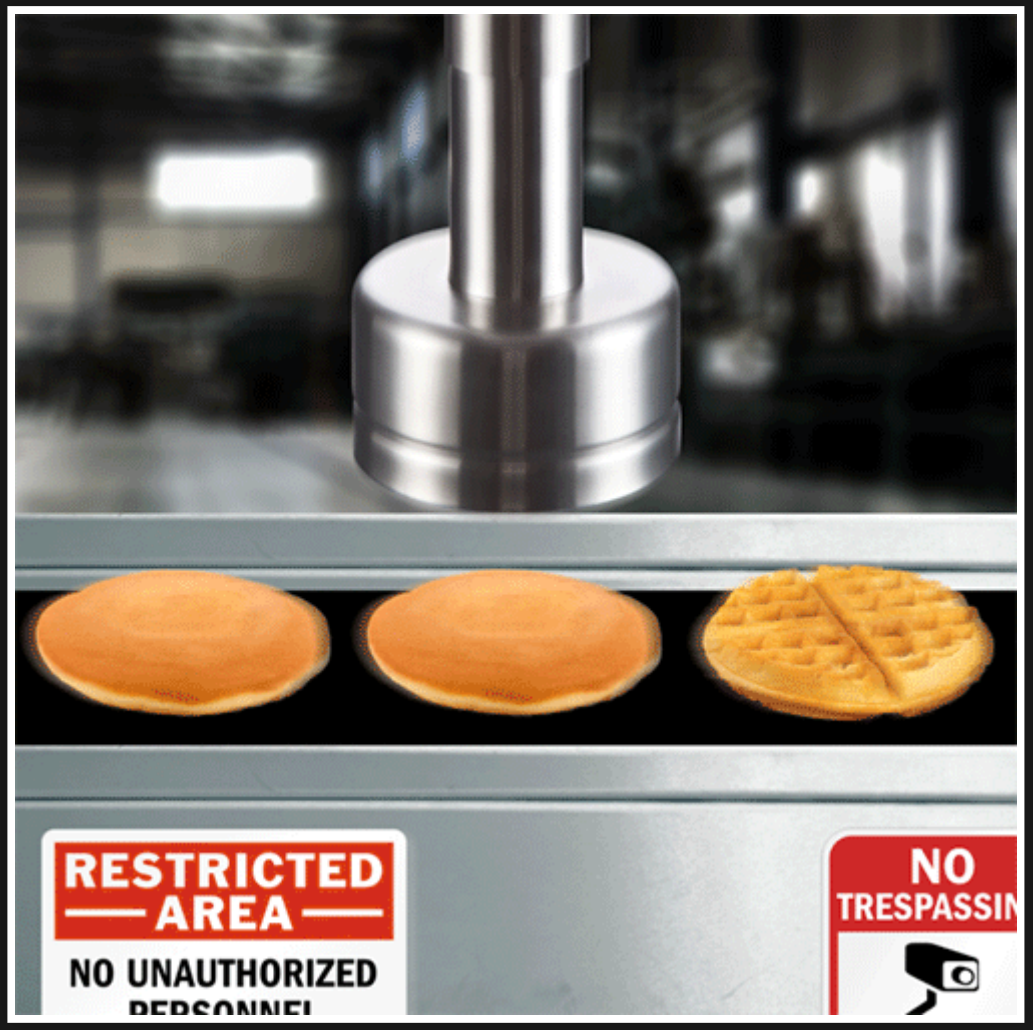
Just one more thing...

Classes (in JavaScript) were added in ES6

Before we used **prototypes** (not covered in this course)

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.


RESTRICTED
AREA
NO UNAUTHORIZED
NO
TRESPASSIN

What does a class look like in JavaScript?

```
1 class Animal {
2 }
```

Classes are just like objects - we can store methods and properties on them

Classes are just like objects - we can store methods and properties on them

In fact, we could represent our classes as objects.

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)

We don't use classes directly, we must instantiate (copy) them

We don't use classes directly, we must instantiate (copy) them

We copy using the new keyword

```
1 class Animal {
2 }
3
4 const dog = new Animal();
```

```
class Animal {
}

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

Naming conventions when writing a class

Naming conventions when writing a class

Class names should be capitalised!

```
Animal {} ✔️
```

not

```
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() ❌
```

Classes can include a special method called the **constructor**, which is called 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 can also use the constructor to set properties

We must use **this** to refer to the itself

```
1  class Animal {
2
3          constructor() {
4                  this.noise = "woof!";
5          }
6  }
7
8  const dog = new Animal();
9
10 console.log(dog.noise); // "woof!"
```

constructor is optional!

```
class Animal {

        constructor() {
                this.noise = "woof!";
        }
}

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

console.log(dog.noise); // "woof!"
console.log(cat.noise); // "woof!" 😹
```

We can also pass in properties via the **constructor**

We can also pass in properties via the **constructor**

Remember the **constructor** is basically just a function

```
class Animal {

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

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

console.log(dog.noise); // "woof!"
```

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!"
```

I Am Devloper

@iamdevloper

manager: we need to design an admin system for a veterinary centre

dev: ok, this is it, remember your training

class Dog extends Animal {}