ES6 SYNTAX

" ES6 is a version of JavaScript. It represents the end of ***old*** *JavaScript and the start of* ***new*** *JavaScript "*

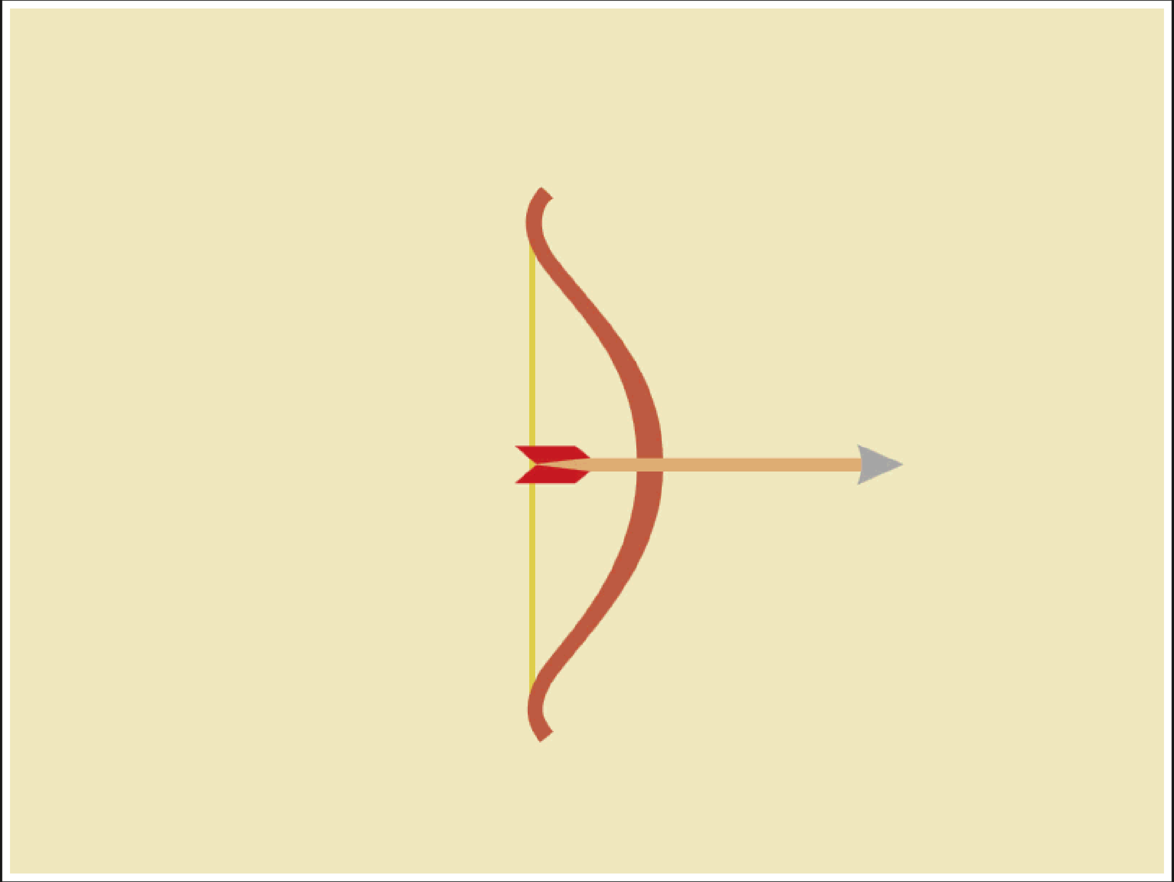
JAVASCRIPT VERSIONS

ES1	…	ES5	ES6	ES7	ES8	ES9	E
1997		2009/2011	2015	2016	2017	2018	20

ES6 syntax

1. Arrow functions
2. Destructuring
3. Organising our code (ES6 Modules)

Arrow Functions



Arrow functions are:

- A shorter way of writing functions
- A way of avoiding issues with context *"this"* - more on this later

An example with named functions

```
function foobar() {
}
```

```
const foobar = () => {
}
```

An example with anonymous functions

```
[21, 12, 13].forEach(function() {
});
```

```
[21, 12, 13].forEach(() => {
});
```

Destructuring

" Destructuring ... is a JavaScript expression that makes it possible to unpack values from arrays, or properties from objects, into distinct variables. " - MDN Docs

" Transforming the way data is structured (in an array or an object) "

Destructuring arrays

```js
const subtitle = [ 12, 'The monkeys are listening...' ];

const id = subtitle[0];
const text = subtitle[1];
```

Destructuring arrays

```
const subtitle = [ 12, 'The monkeys are listening...' ];

const id = subtitle[0];
const text = subtitle[1];
```

```
const subtitle = [ 12, 'The monkeys are listening...' ];

const [ id, text ] = subtitle;
```

Destructuring arrays

```javascript
const subtitle = [ 12, 'The monkeys are listening...' ];

const id = subtitle[0];
const text = subtitle[1];
```

```javascript
const subtitle = [ 12, 'The monkeys are listening...' ];

const [ id, text ] = subtitle;
```

`id` and `text` are new local variables

So far we've been assigning individual values into variables

So far we've been assigning individual values into variables

What if we wanted to assign more than one value into a variable?

So far we've been assigning individual values into variables

What if we wanted to assign more than one value into a variable?

We can do that with the rest operator, represented by 3 dots (...)

In the following example, we will group the last 3 values in the array together

```
const subtitle = [ 12, 'The monkeys', 'are', 'listening' ];
```

In the following example, we will group the last 3 values in the array together

```
const subtitle = [ 12, 'The monkeys', 'are', 'listening' ];
```

In the following example, we will group the last 3 values in the array together

```
const [ id, ...text ] = subtitle;

console.log(text); // [ 'The monkeys', 'are', 'listening' ]
```

Destructuring objects

```
const subtitle = {
        id: 12,
        title: 'The monkeys are listening...'
};

const { id, title } = subtitle;
```

Destructuring objects

```
const subtitle = {
        id: 12,
        title: 'The monkeys are listening...'
};

const { id, title } = subtitle;
```

`id` and `text` are new local variables

Organising our code with ES6 Modules

The programs we've been working with so far have been pretty small.

What happens though, if our code spans hundreds of lines long?

Here's a good of why we should organise our code.

With ES6 modules we can split our code into multiple files

With ES6 modules we can split our code into multiple files

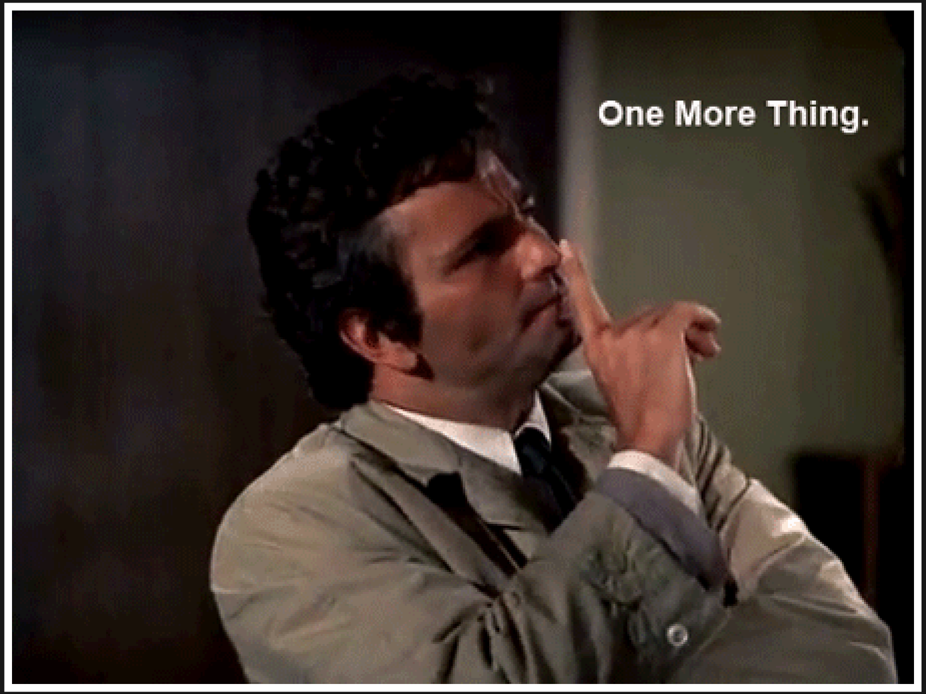
We use the `import` and `export` keywords to handle our code

```javascript
// file library.js

export function doSomething() {
        // some logic
}
```

```js
// file library.js

export function doSomething() {
        // some logic
}
```

```js
// file app.js

import { doSomething } from './library.js';

doSomething();
```

Before we end...



Even though ES6 is "modern", it's still useful to know how to read older versions of JavaScript

Even though ES6 is "modern", it's still useful to know how to read older versions of JavaScript

You may come across code written in ES5 or earlier