## String Interpolation

Start to make our application a bit more re-usable by storing the setup and punchline of our joke as properties on our component class instead of hardcoded in the HTML.

**Learning Outcomes**

* Understand classes in more detail, how to define them and how to use them.
* What is string interpolation and the moustache syntax {{ }}



**Motivation**

In the previous chapter we created our first Angular application, a very simple one with only one component called JokeComponent with a tag of joke.

Now whenever we want to display that joke in our application we simply add the tag <joke></joke> in our HTML.

But it’s not very re-usable, it just shows the same joke over and over again, a more re-usable JokeComponent would be one where the developer can use different jokes.

Firstly lets add some properties or our class, like so:

class JokeComponent {

setup: string;

punchline: string;

}

We are saying that this class has two properties, setup and punchline, both of them can only hold strings.

The code :string is something called a type, and it’s a core part of TypeScript, something you probably are not used to if you’ve never worked with typed languages before.



We will discuss Types in more detail in the next section but in summary if we ever try to make these properties hold anything other than a string TypeScript will throw an error.

**Classes**

Remember I said that classes are blueprints for objects, or in other words *instructions* for how to create an object. To actually create an object using a class we use the new keyword, like so:

let joke = new JokeComponent()

joke is an object created using the JokeComponent class, another word for an object created using a class is a *class instance* or just *instance*.

Since we have added some properties to the JokeComponent class the class instance that is created also has those properties.

console.log(joke.setup);

console.log(joke.punchline);

If we really did try to run the above code we would see that nothing gets printed out to the console, that’s because the setup and punchline properties have not been initialised, they are blank.

With classes how we initialise properties when we instantiate an object is via a special class function called a constructor.

class JokeComponent {

setup: string;

punchline: string;

constructor() {

this.setup = "What did the cheese say when it looked in the mirror?";

this.punchline = "Halloumi (Hello Me)";

}

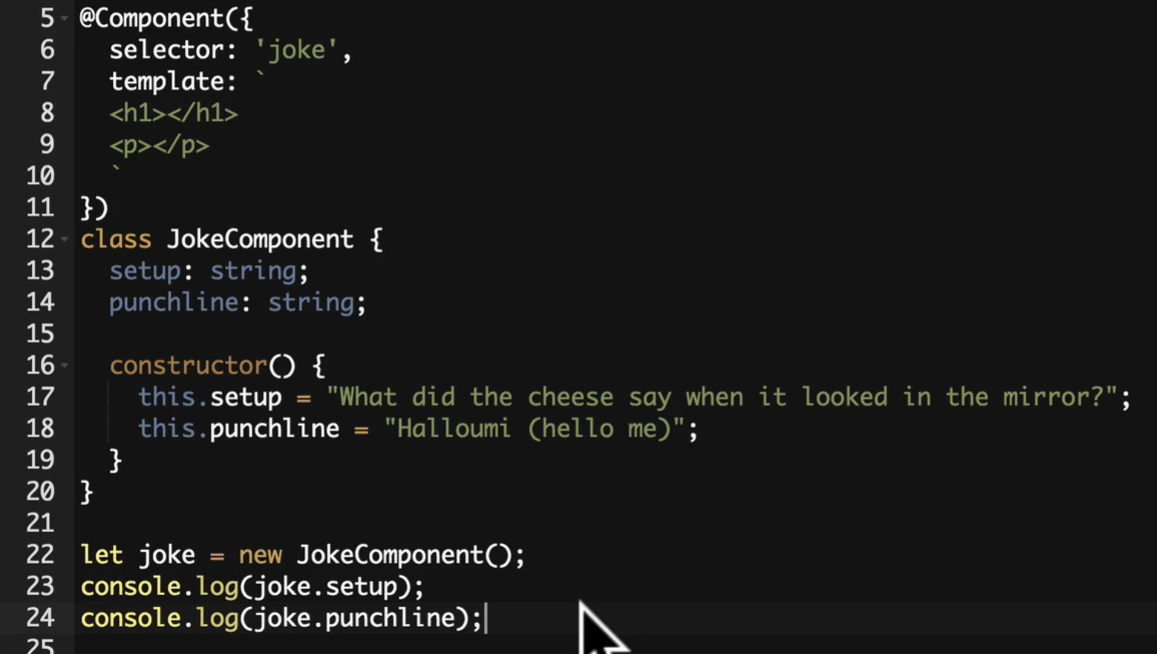
}

Now when we instantiate the class the constructor function is called and this initialises the setup and punchline properties.

In the interest of brevity I’m initialising our properties in the constructor, the

recommended approach with initialising a component is to use *Component*

*Lifecycle Hooks*, again more on that later.



**String Interpolation**

Currently the setup and punchline is hardcoded into the HTML template of our JokeComponent. We need to have the template output the contents of our properties instead.

We can do that in the template by using the special {{ }} syntax, also known as *moustache syntax*.

The {{ }} contains JavaScript which is run by Angular and the output inserted in the HTML.

So if we put {{ 1 + 1 }} in the template the number 2 would be inserted into the HTML.

The template knows about the JokeComponent class it’s attached to so in-between the {{ }} we can also read properties or even call functions on our JokeComponent and have the output inserted in the

HTML.

We just want to display the values of the setup and punchline properties so we just use those, like so:

<

h1>{{setup}}</h

1>

<

p>{{punchline}}</p

>

**Summary**

We’ve explained how a class is a blueprint for an object and we can create a specific instance of a class using the new keyword.

The class instance can have properties and we can bind those properties to portions of our template by using string interpolation via the {{ }} syntax.

**Listing**

*index.html*

<!

DOCTYPE html

>

<!--

suppress ALL

-->

<

html

>

<

head

>

<link rel="stylesheet"

href="https://maxcdn.bootstrapcdn.com/bootstrap/4.0.0-

alpha.4/css/bootstrap.min.css">

<script src="https://unpkg.com/core-js/client/shim.min.js"></script>

<script src="https://unpkg.com/zone.js@0.7.4?main=browser"></script>

<script src="https://unpkg.com/systemjs@0.19.39/dist/system.src.js"></script>

<script src="systemjs.config.js"></script>

<script>

System.import('script.ts').catch(function (err) {

console.error(err);

});

</script>

<

/head

>

<

body class="container m-t-1"

>

<

joke></joke

>

<

>

/body

<

/html

>

*script.ts*

import {platformBrowserDynamic} from '@angular/platform-browser-dynamic';

import {NgModule} from '@angular/core';

import {BrowserModule} from '@angular/platform-browser';

import {Component} from '@angular/core';

@Component({

selector: 'joke',

template: `

<h1>{{setup}}</h1>

<p>{{punchline}}</p>

`

})

class JokeComponent {

setup: string;

punchline: string;

constructor() {

this.setup = "What did the cheese say when it looked in the mirror?";

this.punchline = "Halloumi (Hello Me)";

}

}

@NgModule({

imports: [BrowserModule],

declarations: [JokeComponent],

bootstrap: [JokeComponent]

})

export class AppModule {

}

platformBrowserDynamic().bootstrapModule(AppModule);