Contents

[Environment Set-Up 2](#_Toc178175160)

[Initialising the TypeScript project 2](#_Toc178175161)

[Compiling source files 2](#_Toc178175162)

[Deploying to Web 3](#_Toc178175163)

[TypeScript 8](#_Toc178175164)

# Environment Set-Up

## Initialising the TypeScript project

Start by generating the **tsconfig.json** using **tsc –init**; this will allow us to compile our **.ts** files to **.js** as we go.

A computer screen shot of a black screen

Description automatically generated

## Compiling source files

In order to compile our source files during the development process we will make use of a library called **tsc-watch**. **tsc-watch** will monitor the project for the changes and continuously recompile the **.ts** files to **.js** files, allowing them to be executed by the browser.



A black background with white text

Description automatically generated

At this point there should be 18 errors:

A black screen with red text

Description automatically generated

These will gradually be resolved as we complete the TypeScript in the project.

## Deploying to Web

To aid in deploying our website we will make use of a build tool called **parcel**. Parcel requires very little setup, simply install it as a dev dependencyA screenshot of a computer screen

Description automatically generated

You should now see **Parcel** in your **package.json** A screen shot of a computer code

Description automatically generated

Next you must modify the **start** script in your **package.json** to use **Parcel**

**A screen shot of a computer

Description automatically generated**

Once the **start** script is created simply run the app with **npm start**. By default your src files will be built to /dist and your app will be hosted on **localhost:1234**A screen shot of a computer

Description automatically generated

A screenshot of a computer program

Description automatically generated

At this point you should be able to see a working app at **localhost:1234**.

A white rectangular object with a blue line

Description automatically generated

At the moment there is no data and no place to save it if there was. You should find a **data.json** file in the starter code, we will now use **json-server** to spin up a basic REST API on port 3000 using **data.json** to store our data.

A screenshot of a computer

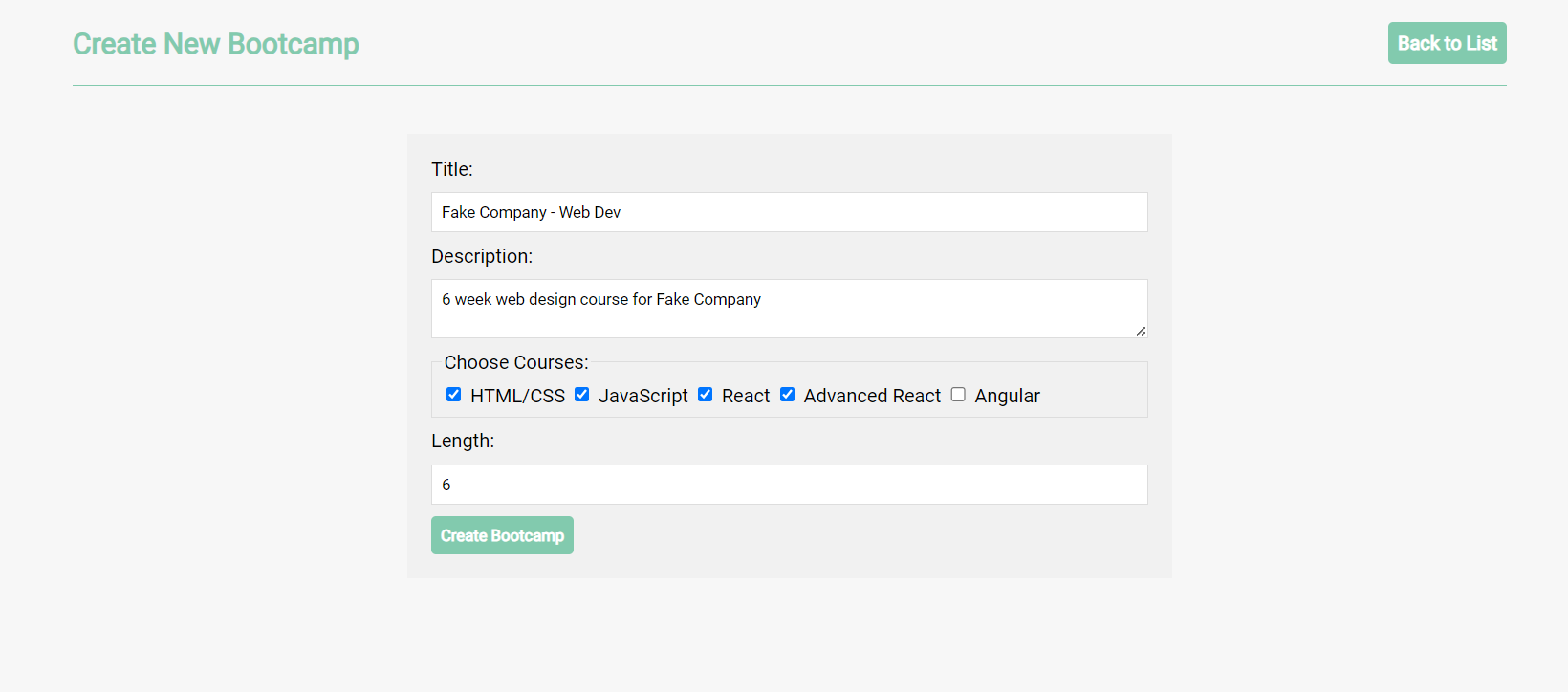
Description automatically generated

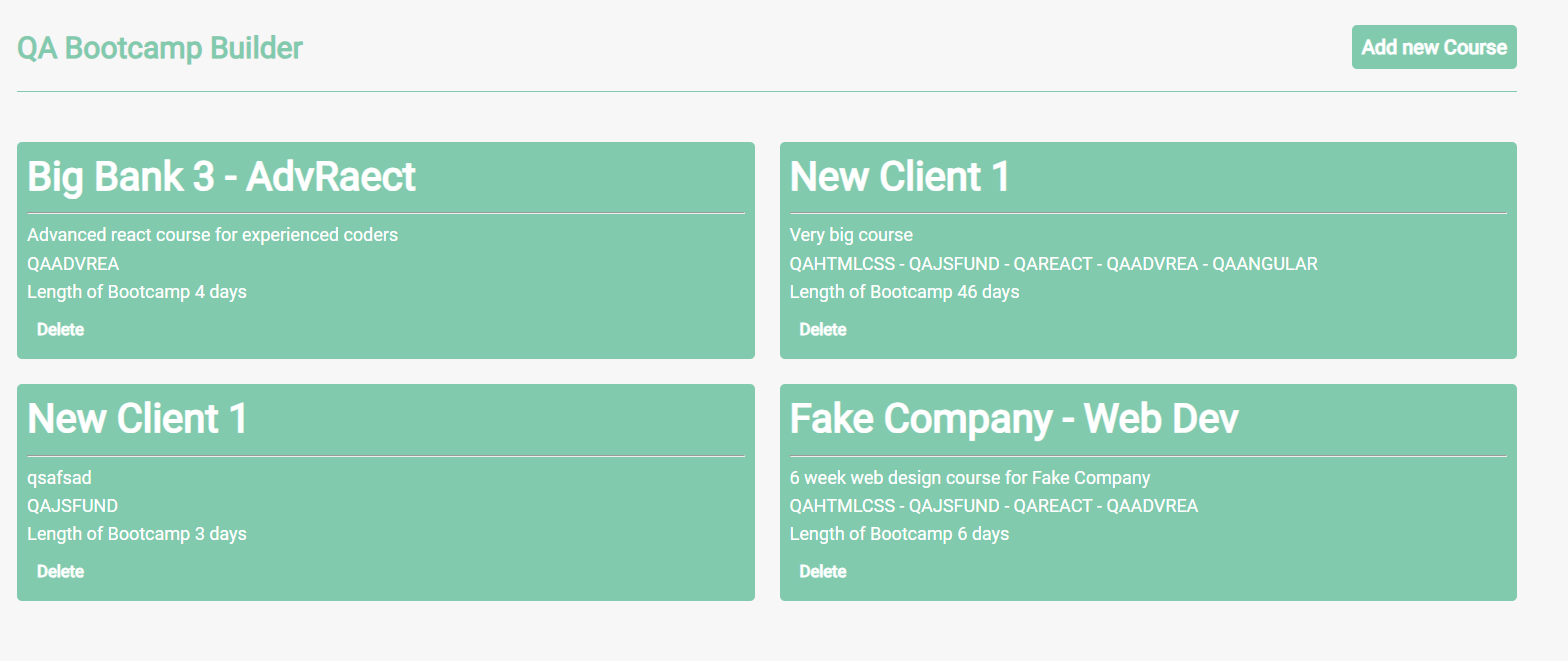
If you check the site again you should now see some test bootcamps.

A screenshot of a computer

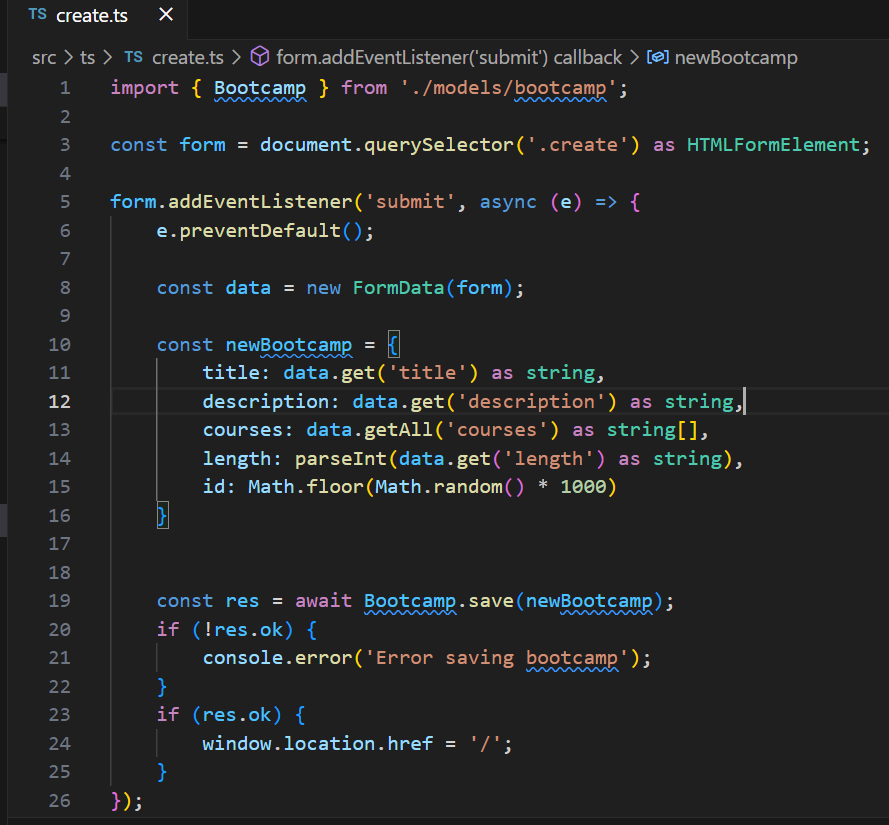
Description automatically generated

To check your setup, try creating a new bootcamp using the button in the top-right corner of the page.



You should be able to see your new bootcamp back on the home page.

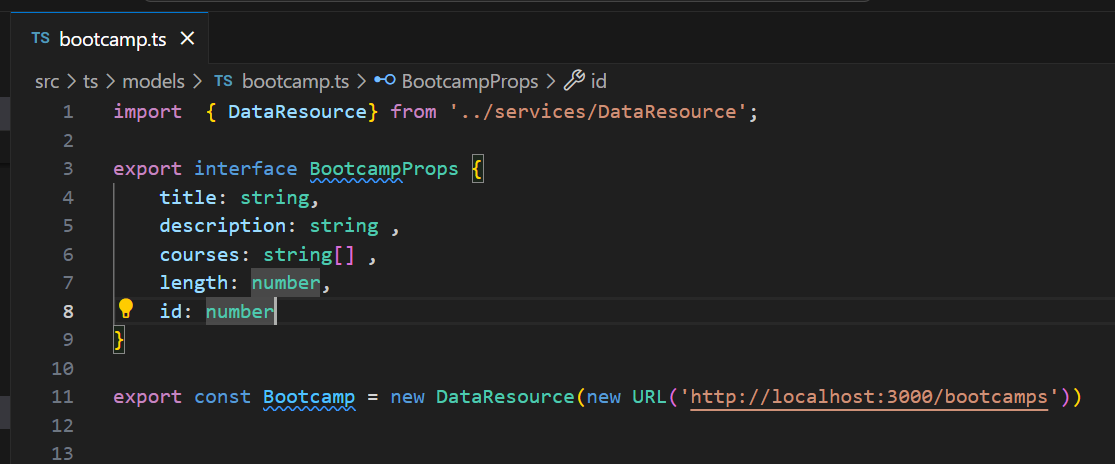
# TypeScript

We shall start by adding types to **create.ts**. First, we should specify that form is an **HTMLFormElement** and then set the types of the form data.

Next, we will add types to the **DataResource** class. Start by making the **endpoint** field a URL, then make both of the **id**s **number**s. Currently we have no appropriate type for the data parameter of the save method so we shall create one in the next step.

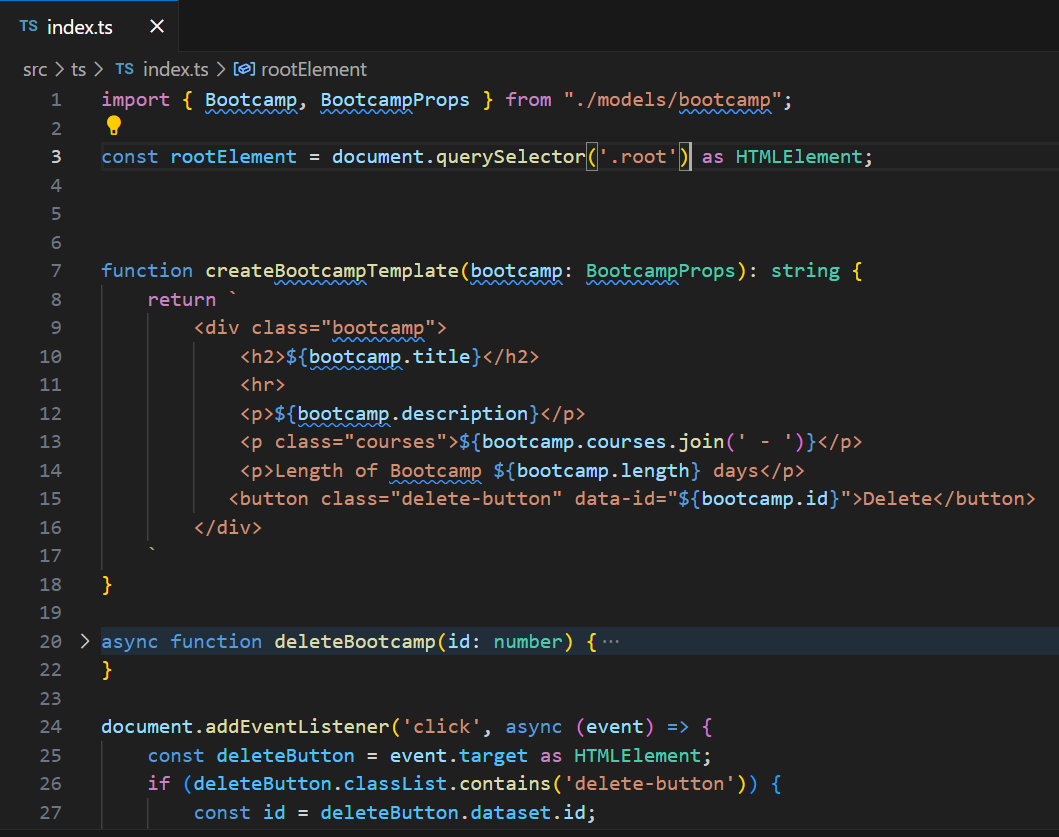
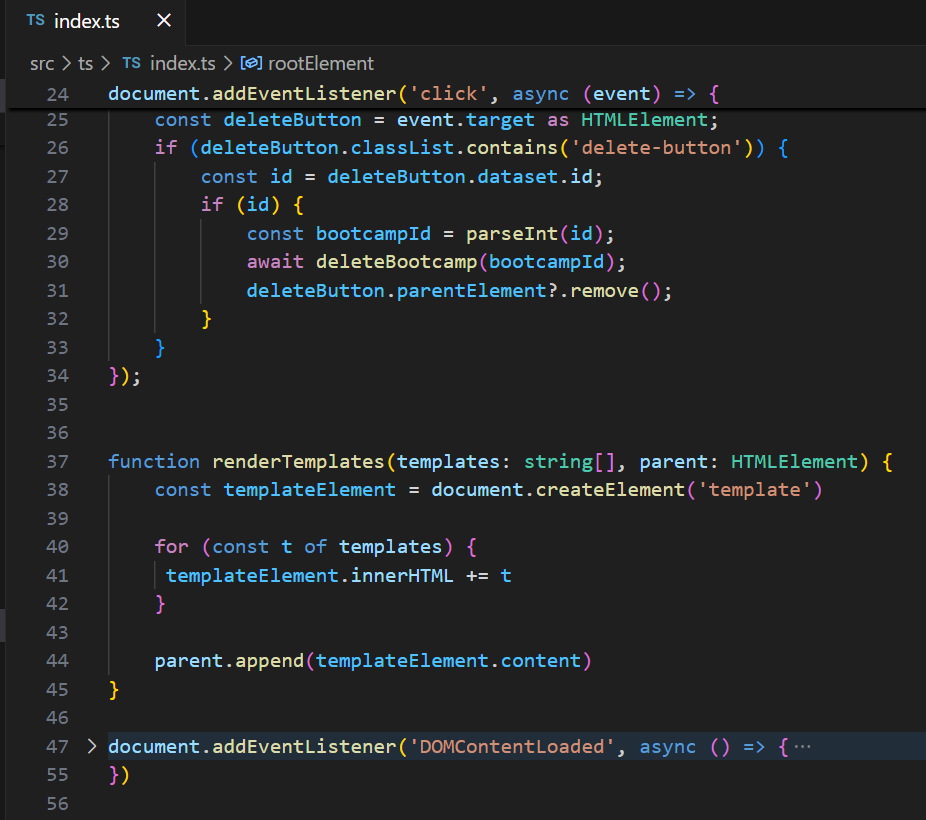
A screen shot of a computer program

Description automatically generated

In **bootcamp.ts** we will create an interface that will provide type checking for the data submitted from our bootcamp form. Make sure to export this interface so it can be used in the rest of the project.

Now import this interface into **DataResource.ts** so you can set the type of the **data** parameter.Also import it into **create.ts** to set the type of **newBootcamp.**A screen shot of a computer program

Description automatically generated

The last place we will need to import the interface into is **index.ts.** Use the interface to set the type of the **bootcamp** parameter.While we’re looking at **index.ts** we shall add types to the rest of the variables. Set the type of **event.target** and **parent** to be **HTMLElement** and **templates** to be **string[]**.Now that we’ve finished adding types to the project it’s to to add some reusability. At the moment **DataResource** only works with BootcampProps but in the future we may want to reuse it with a different model. In order to achieve this we should make **DataResource** generic.A screen shot of a computer program

Description automatically generated

Remember to set the type of the **DataResource** instance in **bootcamp.ts**.A screen shot of a computer program

Description automatically generated