

## Using Angular in Visual Studio Code

[Angular](#) is a popular web development platform developed and maintained by Google. Angular uses [TypeScript](#) as its main programming language. The Visual Studio Code editor supports TypeScript IntelliSense and code navigation out of the box, so you can do Angular development without installing any other extension.

**Note:** To help get you started with Angular development, you can use the [Angular profile template](#) that includes useful extensions, settings, and code snippets.

### [Welcome to Angular](#)

We'll be using the [Angular CLI](#) for this tutorial. To install and use the command line interface as well as run the Angular application server, you'll need the [Node.js](#) JavaScript runtime and [npm](#) (the Node.js package manager) installed. npm is included with Node.js which you can install from [Node.js downloads](#).

**Tip:** To test that you have Node.js and npm correctly installed on your machine, you can type `node -v` and `npm --version`.

To install the Angular CLI, in a terminal or command prompt type:

```
npm install -g @angular/cli
```

This may take a few minutes to install. You can now create a new Angular application by typing:

```
ng new my-app
```

`my-app` is the name of the folder for your application. The `ng new` command prompts you with options for the generated application. Accept the defaults by pressing the Enter key. This may take a few minutes to create the Angular application in [TypeScript](#) and install its dependencies.

Let's quickly run our Angular application by navigating to the new folder and typing `ng serve` to start the web server and open the application in a browser:

```
cd my-app
```

```
ng serve
```

You should see "Welcome to app!!!" on <http://localhost:4200> in your browser. We'll leave the web server running while we look at the application with VS Code.

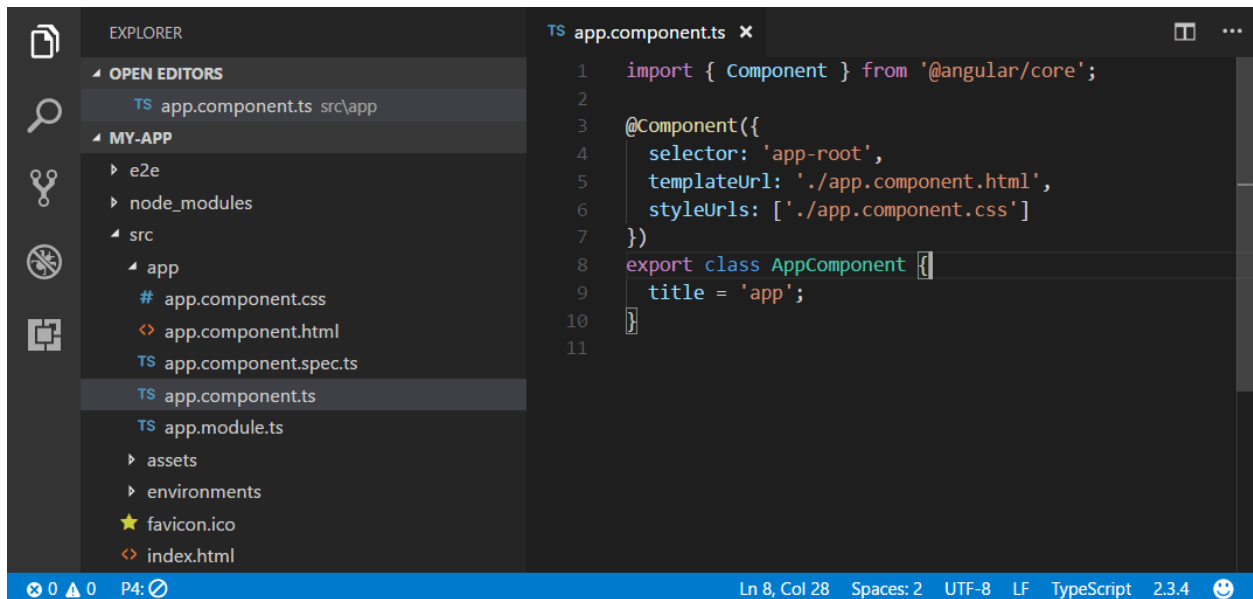
To open your Angular application in VS Code, open another terminal (or command prompt) and navigate to the `my-app` folder and type `code .`:

```
cd my-app
```

```
code .
```

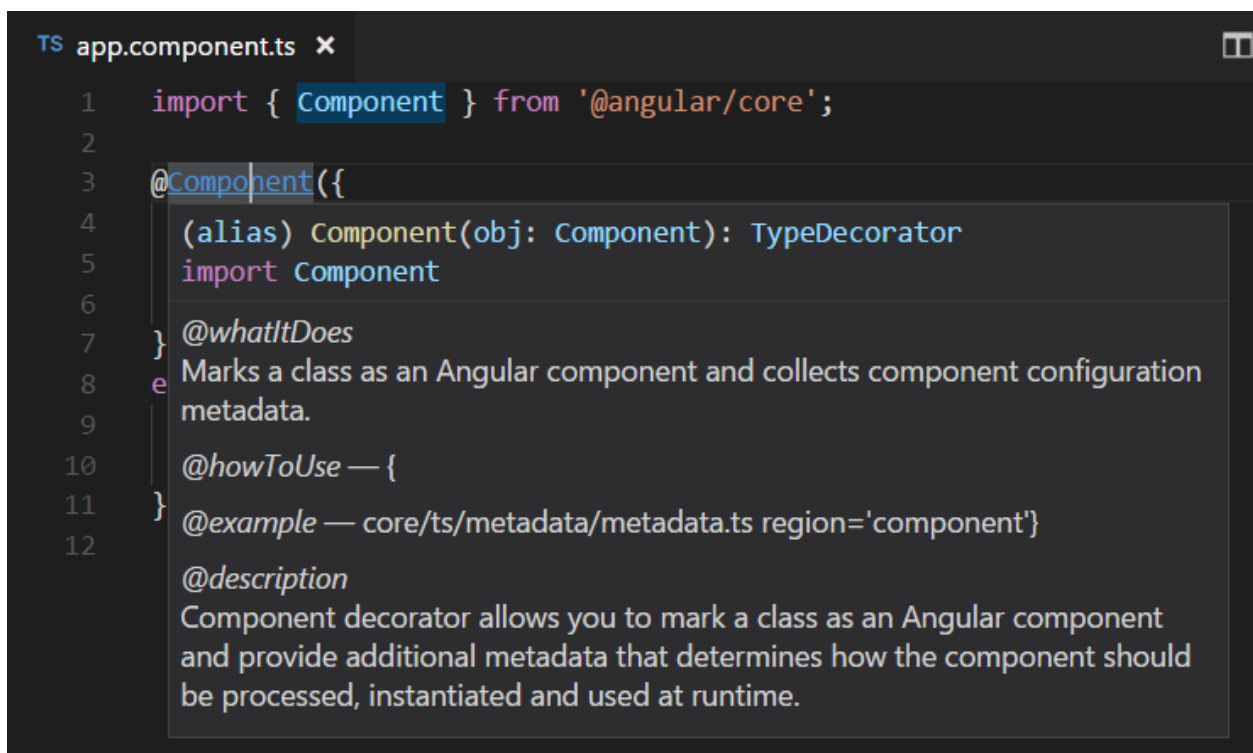
### [Syntax highlighting and bracket matching](#)

Now expand the `src\app` folder and select the `app.component.ts` file. You'll notice that VS Code has syntax highlighting for the various source code elements and, if you put the cursor on a parenthesis, the matching bracket is also selected.

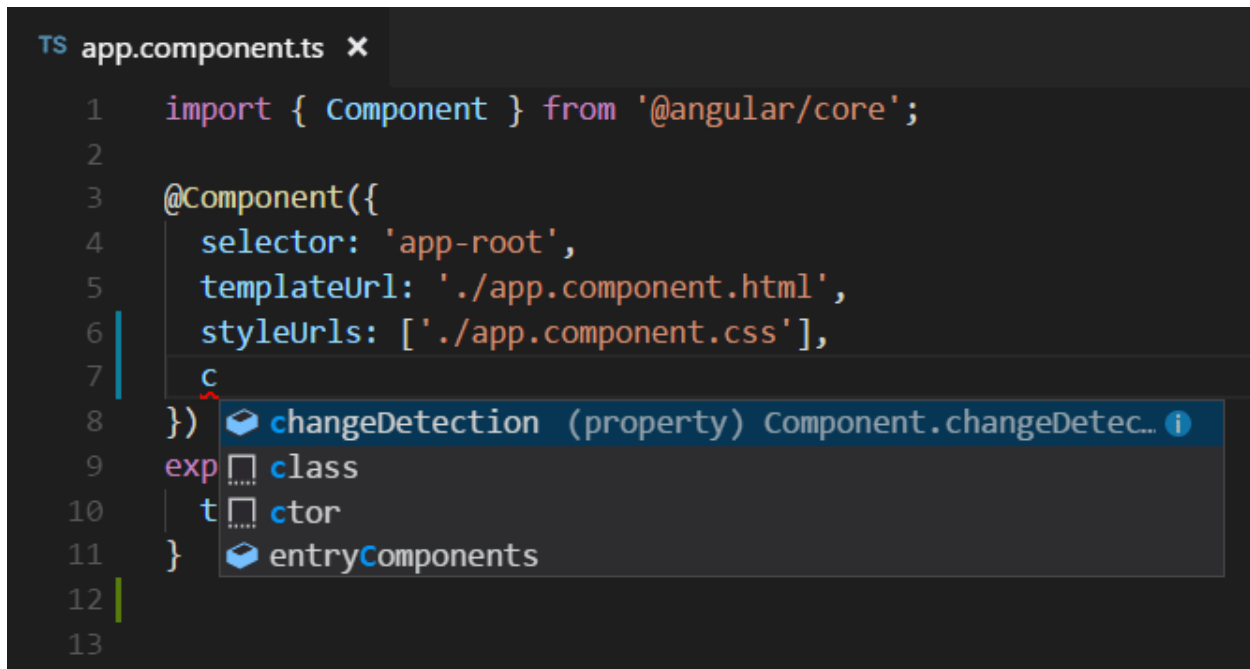


### IntelliSense

As you hover your mouse over text in the file, you'll see that VS Code gives you information about key items in your source code. Items such as variables, classes and Angular decorators are a few examples where you'll be presented with this information.



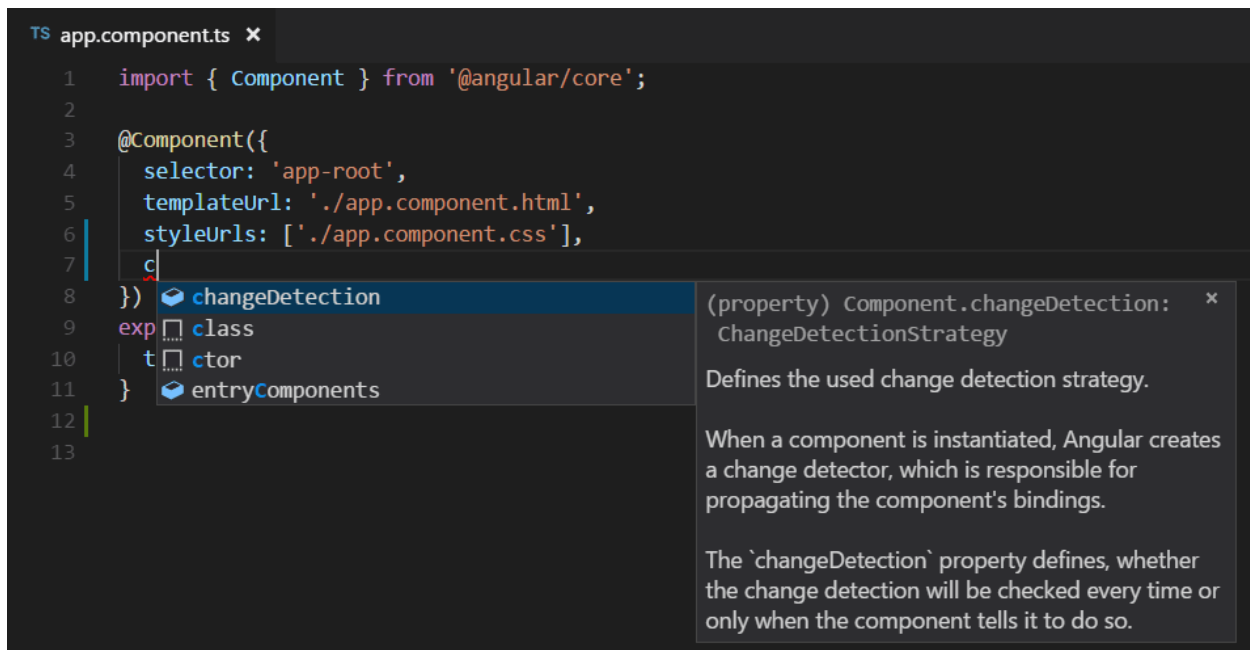
As you start typing in `app.component.ts`, you'll see smart suggestions and code snippets.



```
TS app.component.ts x
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css'],
7    c
8  })
9  exp class
10 t ctor
11 }
12
13
```

The screenshot shows the VS Code editor with a file named `app.component.ts`. The code defines an Angular component using the `@Component` decorator. The decorator object has properties for `selector`, `templateUrl`, and `styleUrls`. The cursor is positioned at the end of the decorator object, after the closing curly brace. A dropdown menu of suggestions is visible, showing `changeDetection` (property), `Component.changeDetec...`, `class`, `ctor`, and `entryComponents`. An information icon (i) is visible next to the `changeDetection` suggestion.

You can click the information button (i) to see a flyout with more documentation.



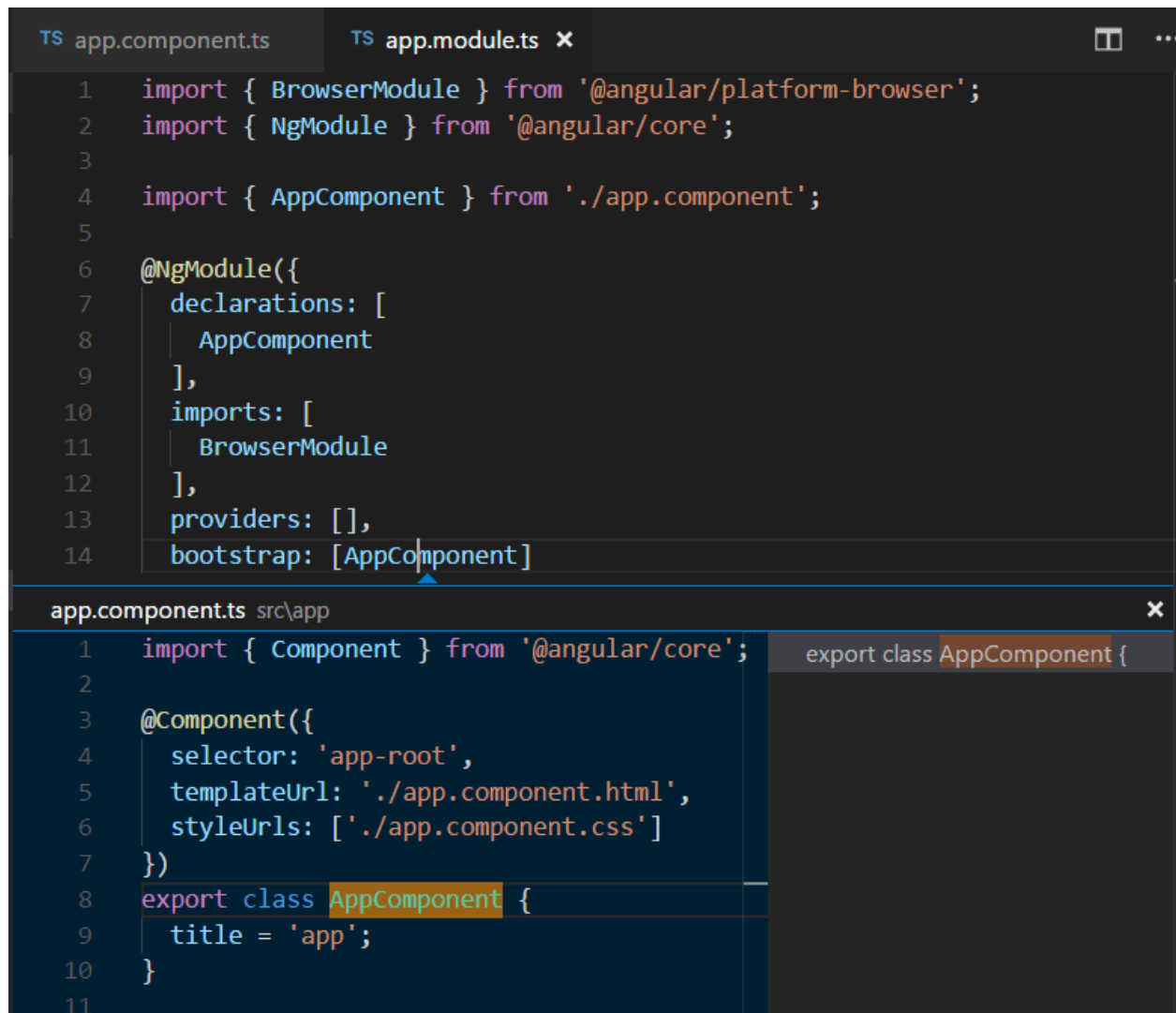
```
TS app.component.ts x
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-root',
5    templateUrl: './app.component.html',
6    styleUrls: ['./app.component.css'],
7    c
8  })
9  exp class
10 t ctor
11 }
12
13
```

The screenshot shows the same code as the previous image, but with the `changeDetection` suggestion selected. A flyout window is open to the right, displaying the documentation for the `changeDetection` property. The flyout title is `(property) Component.changeDetection: ChangeDetectionStrategy`. The text in the flyout states: "Defines the used change detection strategy. When a component is instantiated, Angular creates a change detector, which is responsible for propagating the component's bindings. The 'changeDetection' property defines, whether the change detection will be checked every time or only when the component tells it to do so."

VS Code uses the TypeScript language service for code intelligence ([IntelliSense](#)) and it has a feature called [Automatic Type Acquisition](#) (ATA). ATA pulls down the npm Type Declaration files (\*.d.ts) for the npm modules referenced in the package.json.

[Go to Definition, Peek definition](#)

Through the TypeScript language service, VS Code can also provide type definition information in the editor through **Go to Definition** (F12) or **Peek Definition** (Alt+F12). Open the app.module.ts file and put the cursor over AppComponent in the bootstrap property declaration, right click and select **Peek Definition**. A [Peek window](#) will open showing the AppComponent definition from app.component.ts.



Press Escape to close the Peek window.

### [Hello World](#)

Let's update the sample application to "Hello World". Go back to the app.component.ts file and change the title string in AppComponent to "Hello World".

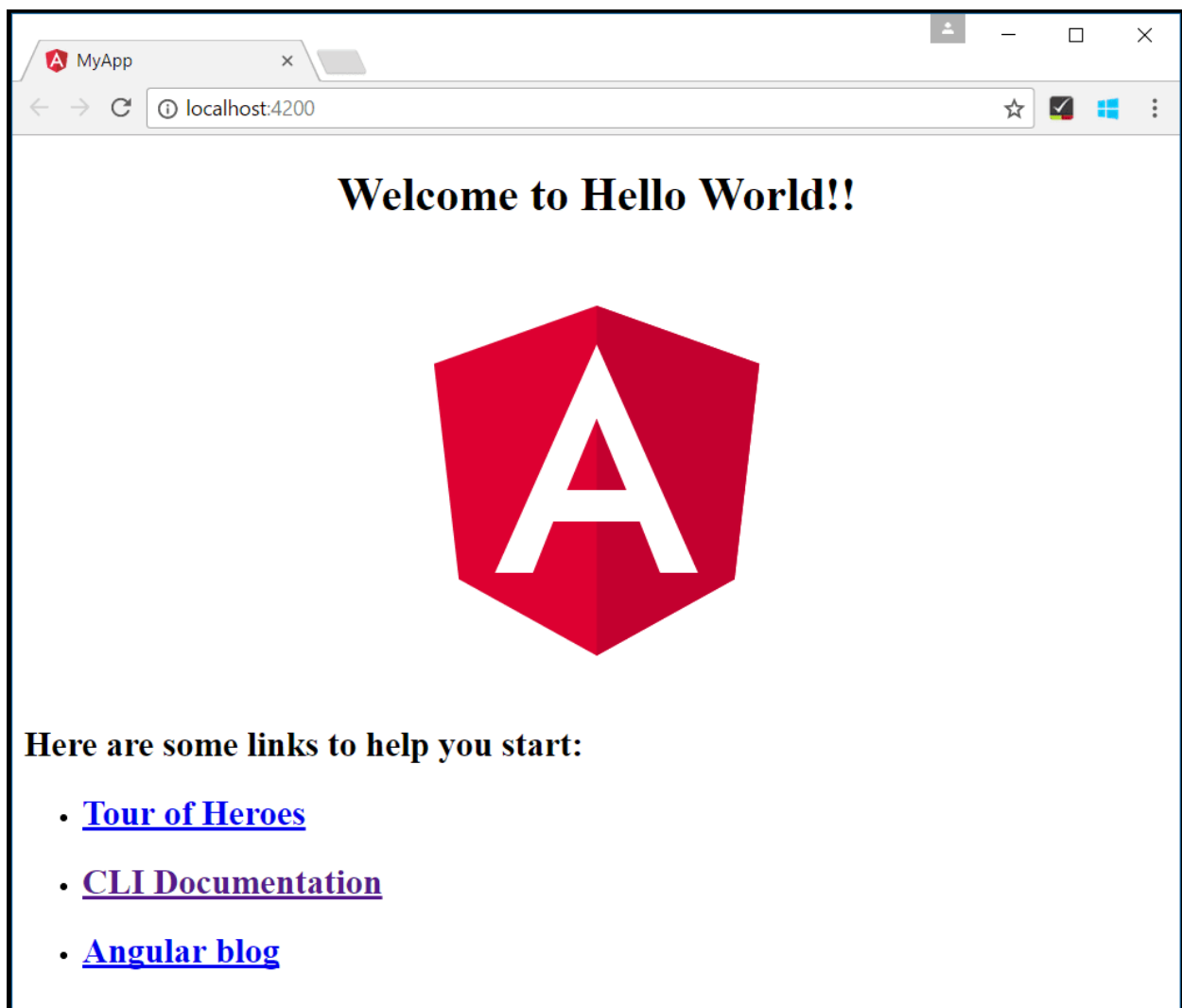
```
import { Component } from '@angular/core';
```

```
@Component({
```

```
selector: 'app-root',  
templateUrl: './app.component.html',  
styleUrls: ['./app.component.css']  
})  
  
export class AppComponent {  
  title = 'Hello World';  
}
```

Once you save the app.component.ts file, the running instance of the server will update the web page and you'll see "Welcome to Hello World!!".

**Tip:** VS Code supports Auto Save, which by default saves your files after a delay. Check the **Auto Save** option in the **File** menu to turn on Auto Save or directly configure the files.autoSave user [setting](#).



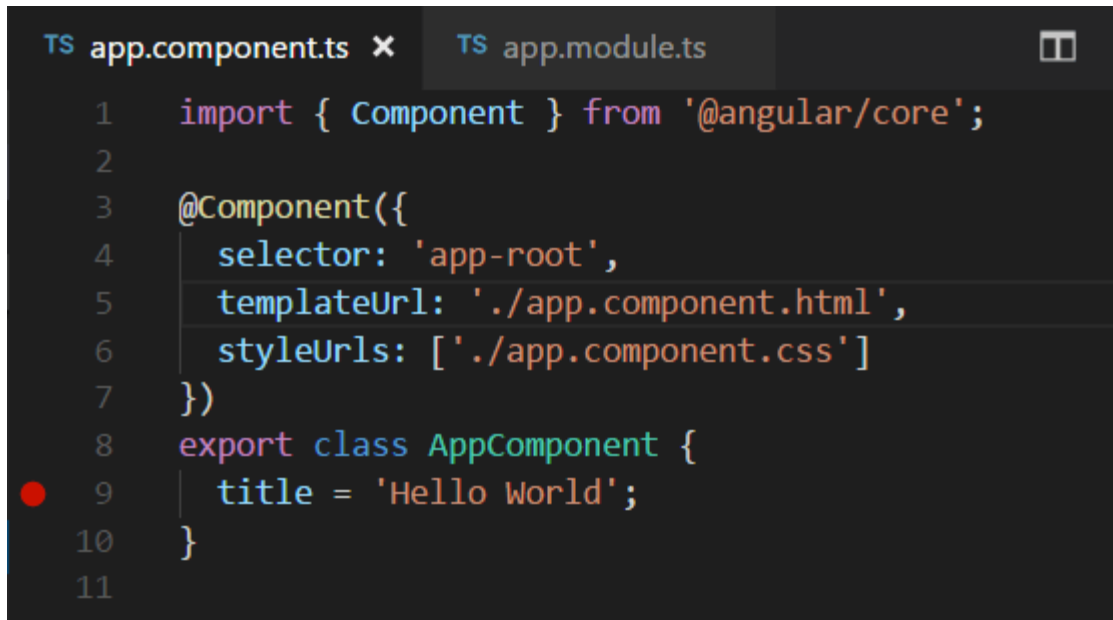
## [Debugging Angular](#)

To debug the client side Angular code, we'll use the built-in JavaScript debugger.

Note: This tutorial assumes you have the Edge browser installed. If you want to debug using Chrome, replace the launch type with chrome. There is also a debugger for the [Firefox](#) browser.

### [Set a breakpoint](#)

To set a breakpoint in app.component.ts, click on the gutter to the left of the line numbers. This will set a breakpoint which will be visible as a red circle.



### [Configure the debugger](#)

We need to initially configure the [debugger](#). To do so, go to the **Run and Debug** view (Ctrl+Shift+D) and select the **create a launch.json file** link to create a launch.json debugger configuration file. Choose **Web App (Edge)** from the **Select debugger** dropdown list. This will create a launch.json file in a new .vscode folder in your project which includes a configuration to launch the website.

We need to make one change for our example: change the port of the url from 8080 to 4200. Your launch.json should look like this:

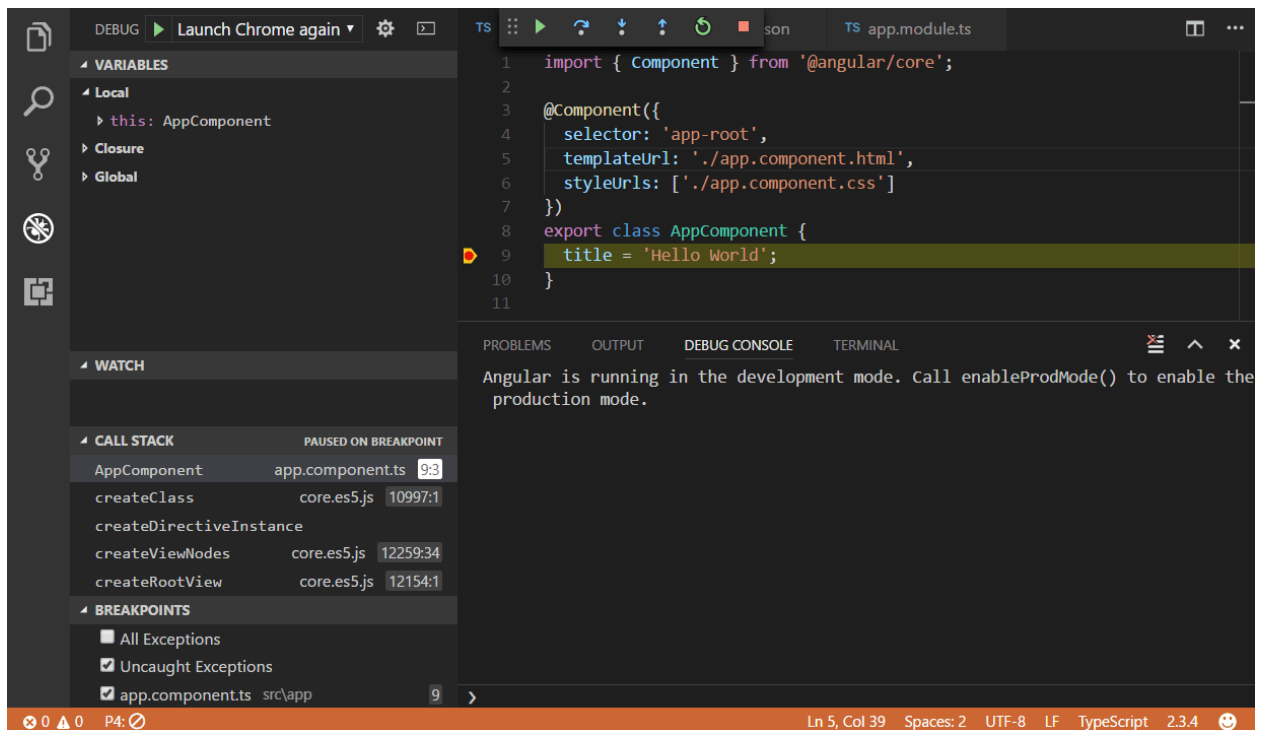
```
{
  "version": "0.2.0",
  "configurations": [
    {
      "type": "msedge",
      "request": "launch",
```

```

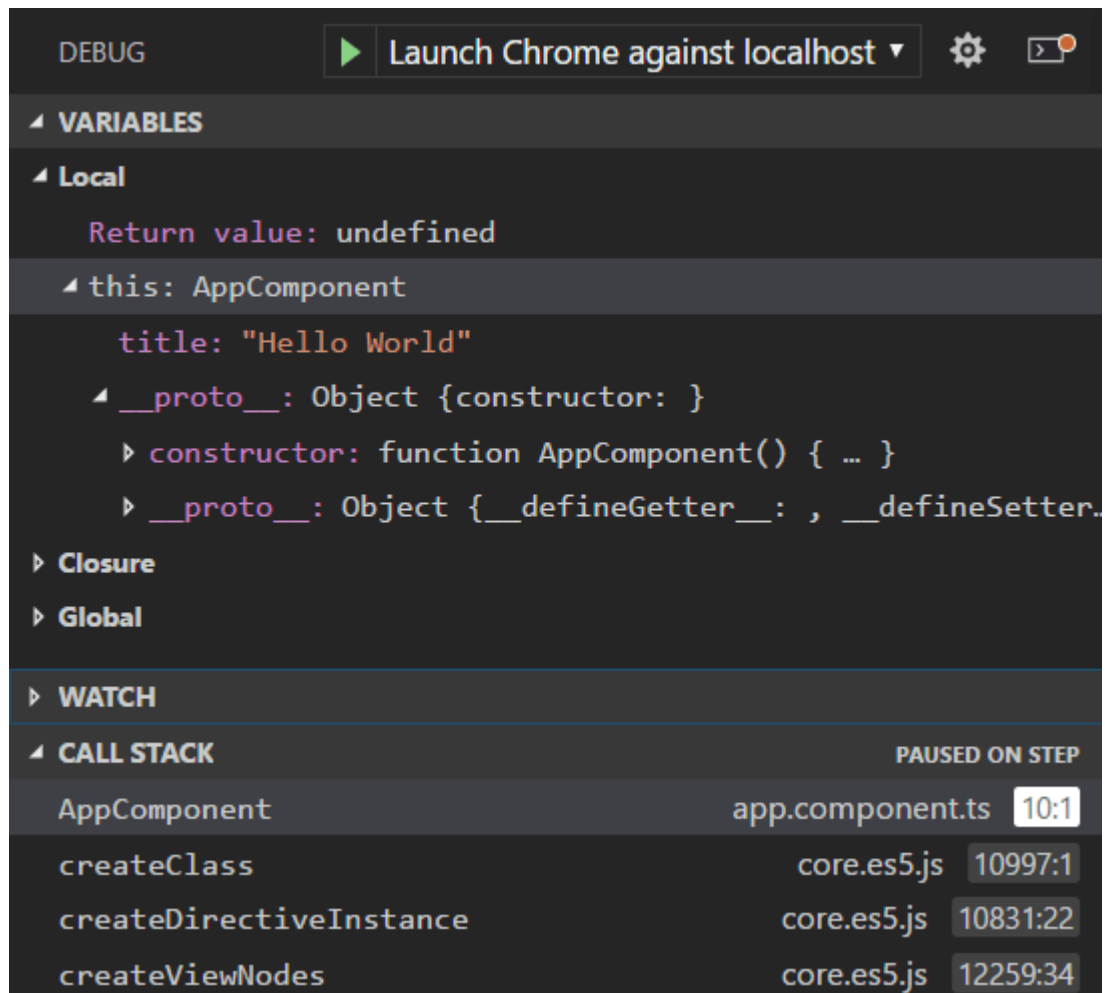
"name": "Launch Edge against localhost",
"url": "http://localhost:4200",
"webRoot": "${workspaceFolder}"
}
]
}

```

Press F5 or the green arrow to launch the debugger and open a new browser instance. The source code where the breakpoint is set runs on startup before the debugger was attached so we won't hit the breakpoint until we refresh the web page. Refresh the page and you should hit your breakpoint.



You can step through your source code (F10), inspect variables such as `AppComponent`, and see the call stack of the client side Angular application.



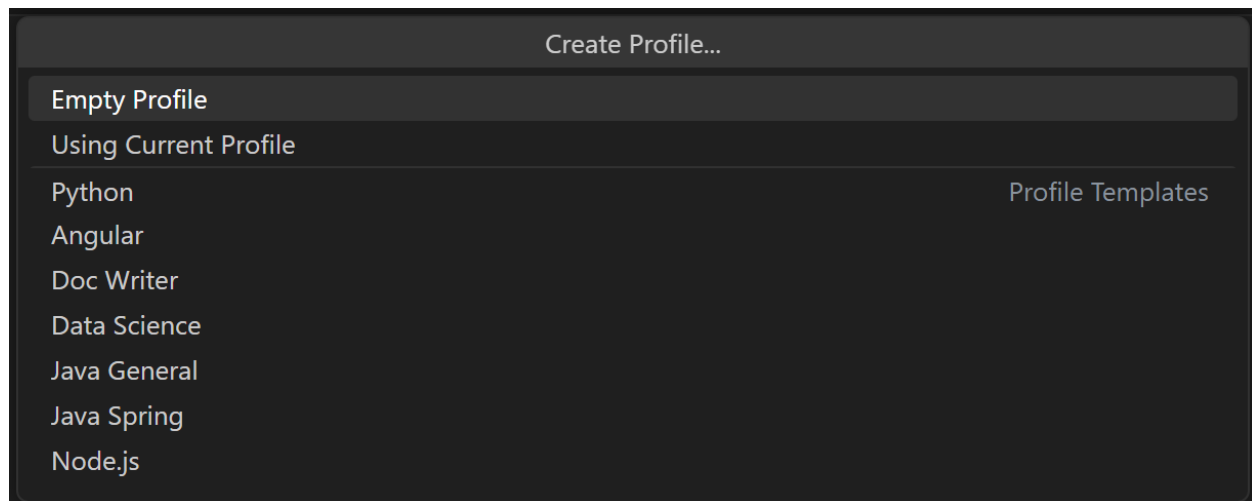
For more information about the debugger and its available options, check out our documentation on [browser debugging](#).

### [Angular profile template](#)

[Profiles](#) let you quickly switch your extensions, settings, and UI layout depending on your current project or task. To help you get started with Angular development, you can use the [Angular profile template](#), which is a curated profile with useful extensions and settings. You can use the profile template as is or use it as a starting point to customize further for your own workflows.

You select a profile template through the **Profiles > Create Profile...** dropdown:





Once you select a profile template, you can review the settings and extensions, and remove individual items if you don't want to include them in your new profile. After creating the new profile based on the template, changes made to settings, extensions, or UI are persisted in your profile.

### [Popular Starter Kits](#)

In this tutorial, we used the Angular CLI to create a simple Angular application. There are lots of great samples and starter kits available to help build your first Angular application.

### [Recipes](#)

The VS Code team has created [recipes](#) for more complex debugging scenarios. There you'll find the [Debugging with Angular CLI](#) recipe which also uses the Angular CLI and goes into detail on debugging the generated project's unit tests.

### [MEAN Starter](#)

If you'd like to see a full MEAN (MongoDB, Express, Angular, Node.js) stack example, look at [MEAN.JS](#). They have documentation and an application generator for a sample MEAN project. You'll need to install and start [MongoDB](#), but you'll quickly have a MEAN application running. VS Code also has great [MongoDB support](#) through the [Azure Databases](#) extension.

### [React](#)

[React](#) is a library for building user interfaces and it is more minimal than angular. If you'd like to see an example of React working with VS Code, check out the [Using React in VS Code](#) tutorial. It will walk you through creating an React application and configuring the launch.json file for the JavaScript debugger.

### [Angular Extensions](#)

In addition to the features VS Code provides out of the box, you can install VS Code extensions for greater functionality.

(This article is sourced from: <https://code.visualstudio.com/docs/nodejs/angular-tutorial> and the content of the article is here in case of link breakage.)