Introduction to Angular

1. Setting the scene
2. Creating an application
3. Understanding the application

Pearson

Section 1: Setting the Scene

- What is Angular?
- Angular versions

What is Angular?

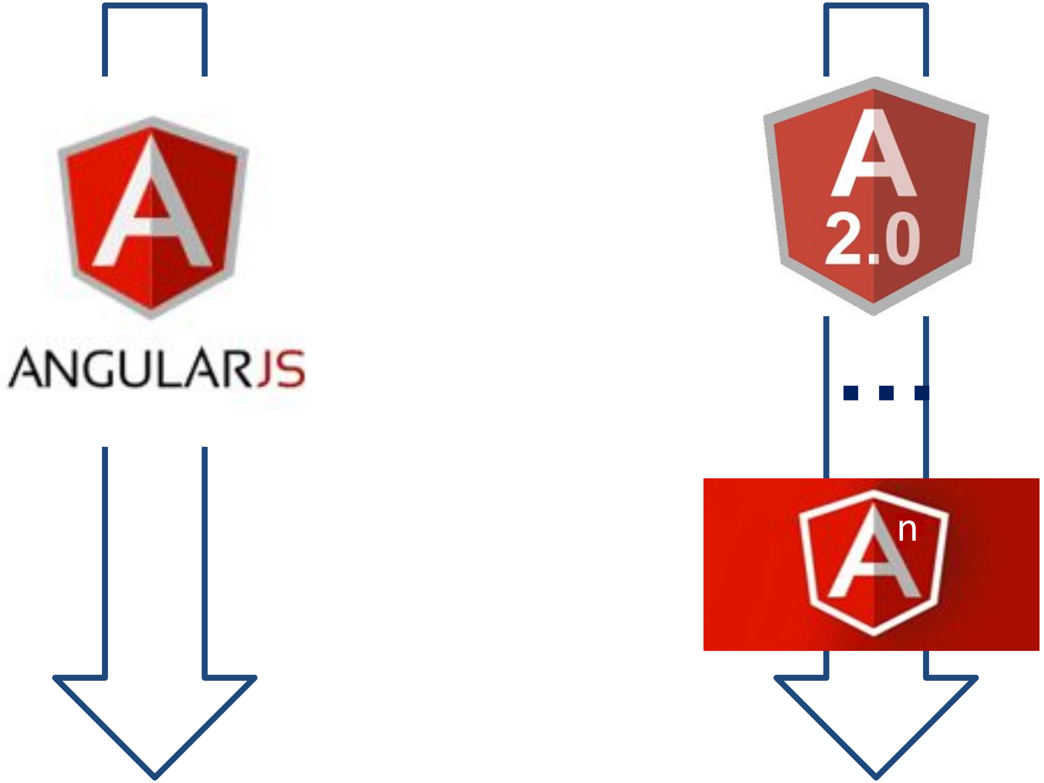
- Angular is a client-side framework from Google
  - Makes it easier to develop large-scale rich Web apps

- How does Angular help?
  - Client-side data binding
  - Routing
  - Dependency injection and services
  - REST integration

- For full details, see https://angular.io/docs

Pearson

Angular Versions



Section 2: Creating an Application

- Overview of Angular CLI
- Getting Angular CLI
- Angular CLI capabilities
- Creating an application
- Reviewing the application
- Serving the application
- Viewing the application

Pearson

Overview of Angular CLI

- Angular CLI is a command-line interface tool that you can use to create a new Angular application
  - Scaffolds a complete template application
  - Generates appropriate config files
  - Reinforces best practices for file names, folders, etc.

- Angular CLI also has commands to generate new artifacts in a standardised way
  - Components, services, directives, etc.

Getting Angular CLI

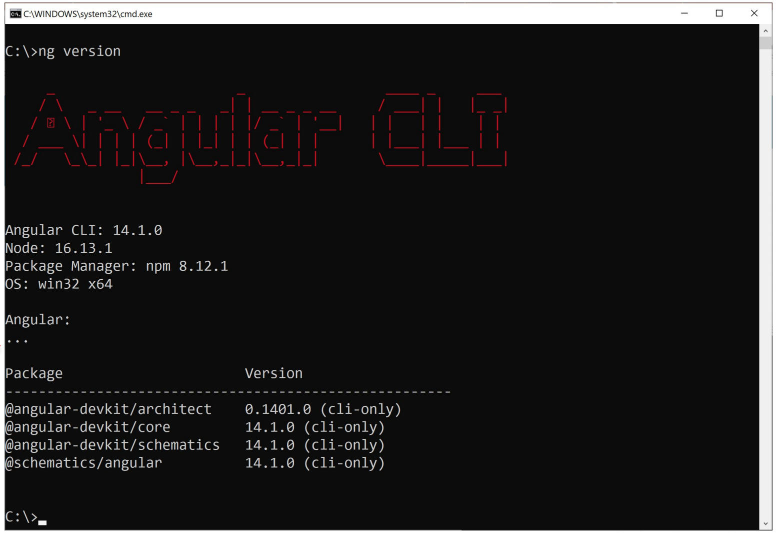
- Angular CLI is a Node.js application
  - You must install Node.js first (version 14 or above)

- You can install Angular CLI on your machine as follows:

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

- You can verify the version of Angular CLI installed:

```
ng version
```



```
C:\>ng version

Angular CLI

Angular CLI: 14.1.0
Node: 16.13.1
Package Manager: npm 8.12.1
OS: win32 x64

Angular:
...

Package                      Version
---------------------------------------------------------
@angular-devkit/architect    0.1401.0 (cli-only)
@angular-devkit/core         14.1.0 (cli-only)
@angular-devkit/schematics   14.1.0 (cli-only)
@schematics/angular          14.1.0 (cli-only)


C:\>
```

Pearson

Angular CLI Capabilities

- Once you'll installed Angular CLI, it enables you to:
  - Create a new application
  - Add artifacts to the application
  - Serve an application, i.e. host in a server
  - Build an application into bundled files
- Angular CLI is currently version 14 (as of June 2022)
  - The Angular version is incremented every 6 months

Creating an Application

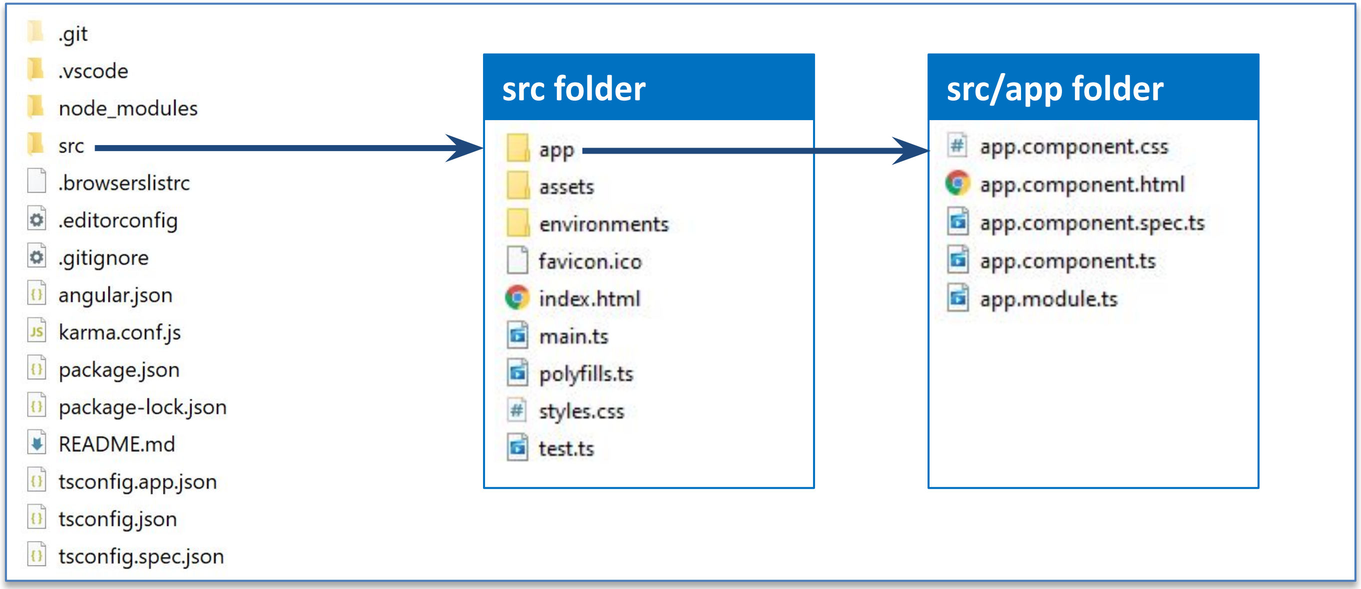
- To create an Angular application using Angular CLI...
  - Run the following command from the command line:

```
ng new DemoApp
```

- You'll be asked a few questions:
  - Would you like to add Angular routing? (choose **No**)
  - Which stylesheet format would you like? (choose **CSS**)

Reviewing the Application

- Angular CLI creates a fully functional Angular app with minimalistic functionality:



```
.git
.vscode
node_modules
src
.browserslistrc
.editorconfig
.gitignore
angular.json
karma.conf.js
package.json
package-lock.json
README.md
tsconfig.app.json
tsconfig.json
tsconfig.spec.json
```

src folder

```
app
assets
environments
favicon.ico
index.html
main.ts
polyfills.ts
styles.css
test.ts
```

src/app folder

```
app.component.css
app.component.html
app.component.spec.ts
app.component.ts
app.module.ts
```

Serving the Application

- To serve the application:
  - Go to the application folder
  - Run the following command:

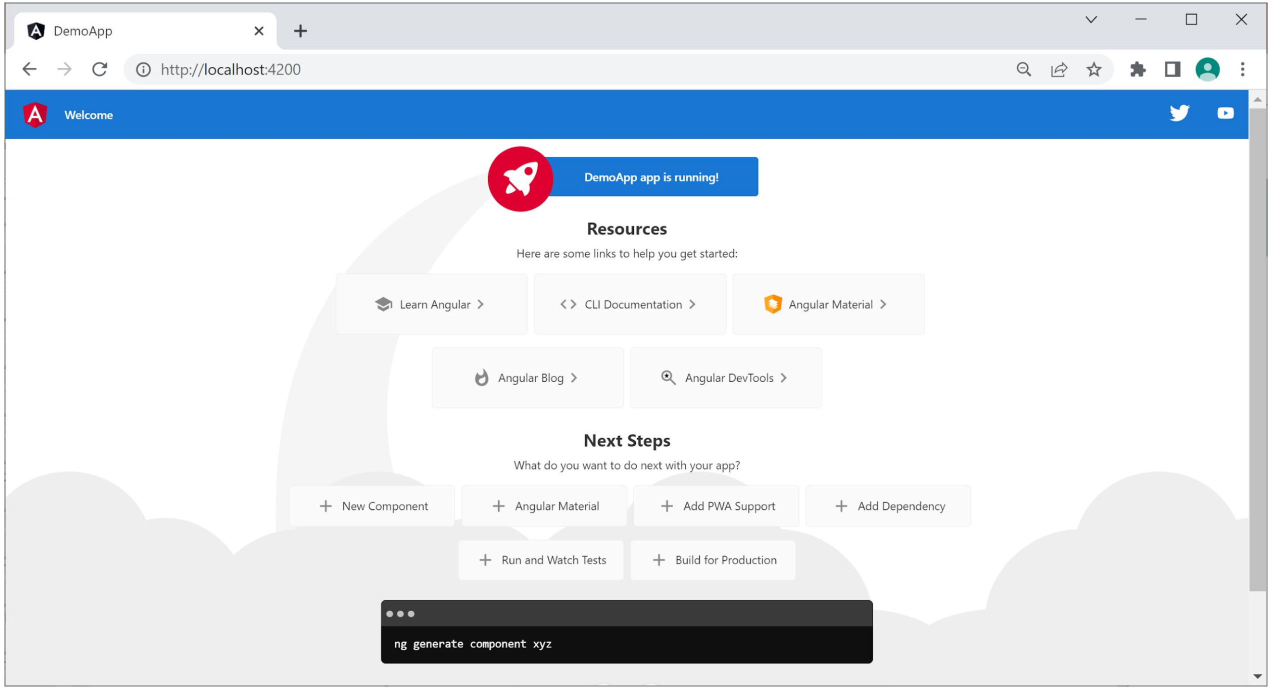
```
ng serve
```

- This command builds and serves the app in memory
  - Default host is `localhost`, default port is `4200`
  - Use `--host` and `--port` for a different host and port

Pearson

Viewing the Application

- To view the app, browse to http://localhost:4200
  - The app automatically reloads if you change source files



```
ng generate component xyz
```

Pearson

Aside: Updating an Existing App

- Angular comes out with a new version every 6 months

- If you have an existing Angular app that you want to update to Angular 14:
  - Uninstall your (old) version of Angular CLI
  - Then install the latest version of Angular CLI
  - Then run the following command:

```
ng update @angular/core @angular/cli --allow-dirty
```

Section 3: Understanding the Application

- Overview
- Key configuration files
- Application home page
- Main source file
- Module source file
- Component source files
- Additional techniques

Overview

- Angular applications have a lot of moving parts
  - There are a lot of config files and source files...

- We'll take a quick journey through some of the key details in this section (details follow later in the course)
  - Config files
  - Application home page
  - Source code files (in TypeScript)

Pearson

Key Configuration Files

- `package.json`
  - Specifies Node.js dependencies (e.g. Angular libraries)
  - Specifies development tools (e.g. TypeScript transpiler)
  - Specifies command-line scripts (e.g. `ng serve`)

- `tsconfig.json`
  - Configures the TypeScript transpiler

- `angular.json`
  - Specifies application home page, `src/index.html`
  - Specifies main source file, `src/main.ts`

Application Home Page

- Here's the application home page:

```
<!doctype html>
<html lang="en">
<head> … </head>
<body>
  <app-root></app-root>
</body>
</html>
```

src/index.html

- When the application is launched:
  - The `<app-root>` element is populated with content generated by our Angular application code

Main Source File

- Here's the main source file in our application:

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

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

src/main.ts

  - Launches `AppModule`, the "module" for our app
  - See next slide for the definition of `AppModule`

Module Source File

- Here's the code for `AppModule`:

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';

@NgModule({
  declarations: [AppComponent],
  imports: [BrowserModule],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

src/app/app.module.ts

  - `AppModule` contains our application components etc.
  - It also designates the "bootstrap" (top-level) component

Component Source Files

- In Angular, a component is a class that renders HTML
  - Instance variables - Data to display
  - `templateUrl` - HTML page to render the data
  - `selector` - Where to render the HTML
  - `styleUrls` - Optional CSS style sheet(s)

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

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

src/app/app.component.ts

Additional Techniques

- HTML template:
  - You can use the `template` property, for inline HTML
  - Use backticks (for ES6 interpolated strings)
  - Use `{{xxx}}` for Angular data binding expressions

```
template: `<h1>The title is {{title}}</h1>`
```

- CSS styles
  - You can use the `styles` property, for inline CSS styles

```
styles: ['h1 { font-family: Verdana; color: red }']
```

Summary

- Setting the scene
- Creating an application
- Understanding the application