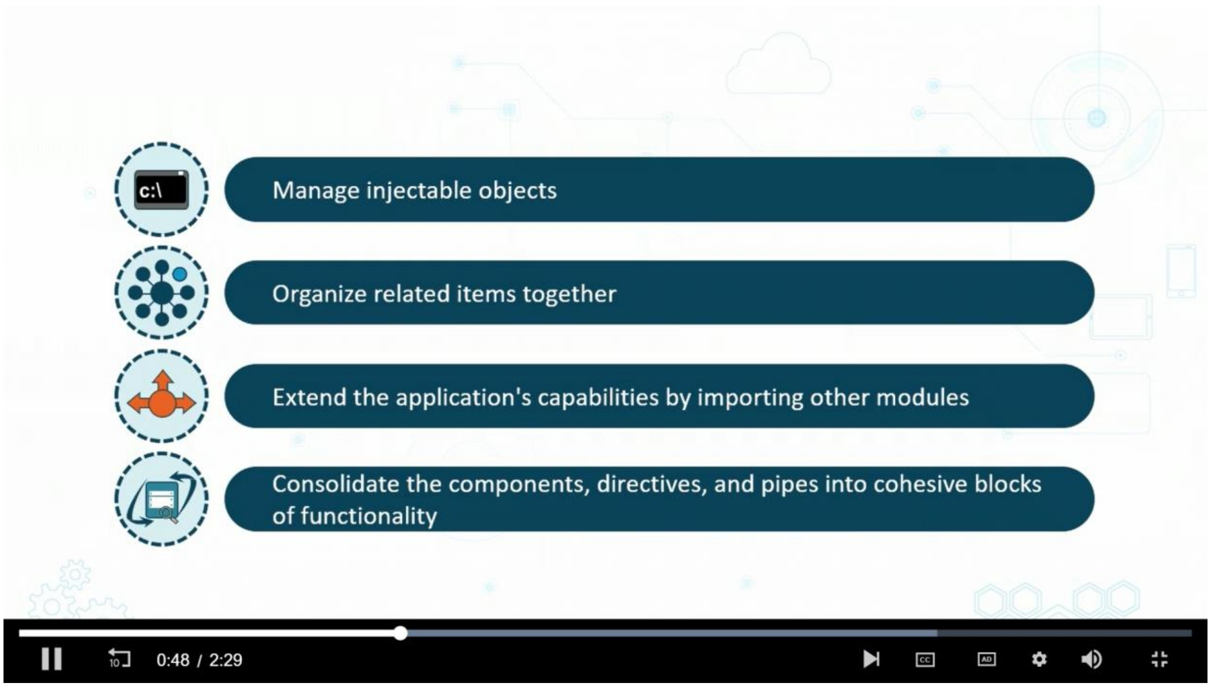
Working With Angular Modules

Ngmodules

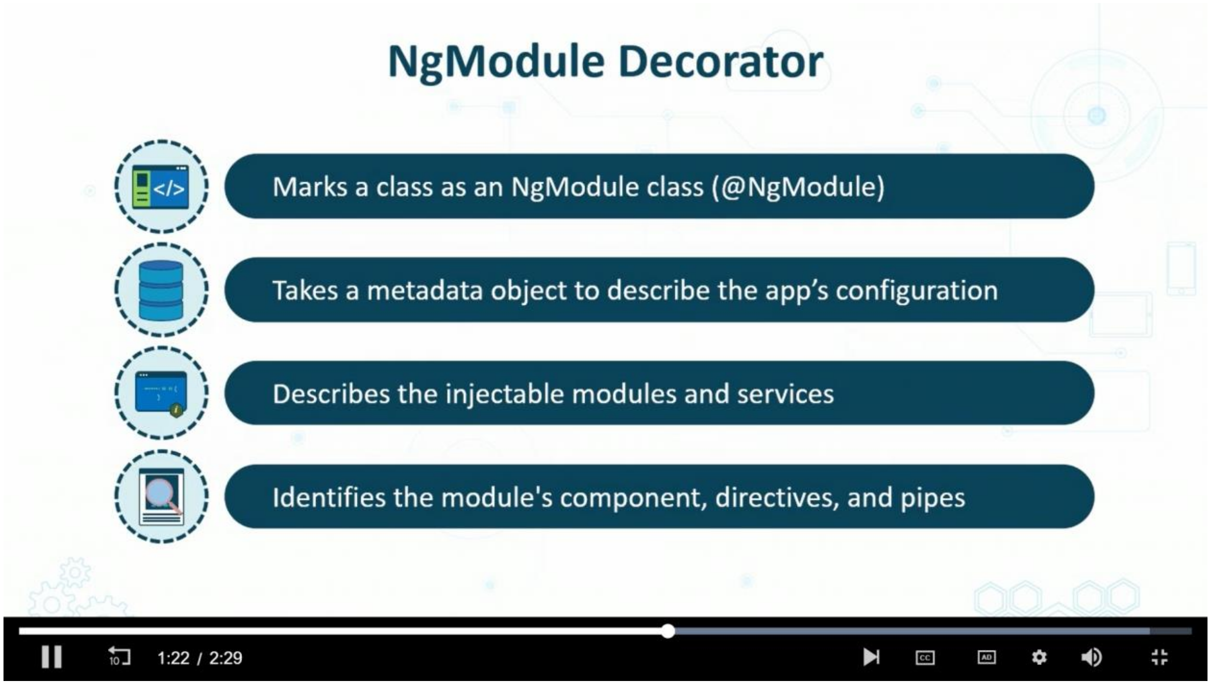


Manage injectable objects

Organize related items together

Extend the application's capabilities by importing other modules

Consolidate the components, directives, and pipes into cohesive blocks of functionality



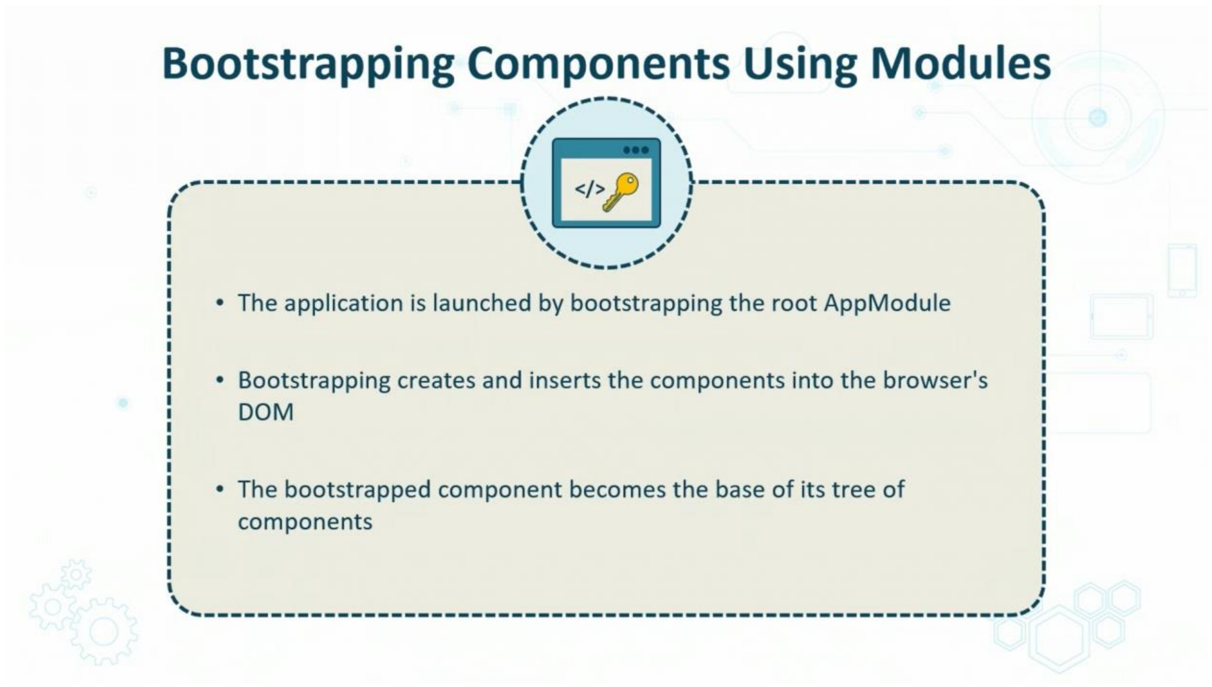
NgModule Decorator

Marks a class as an NgModule class (@NgModule)

Takes a metadata object to describe the app’s configuration

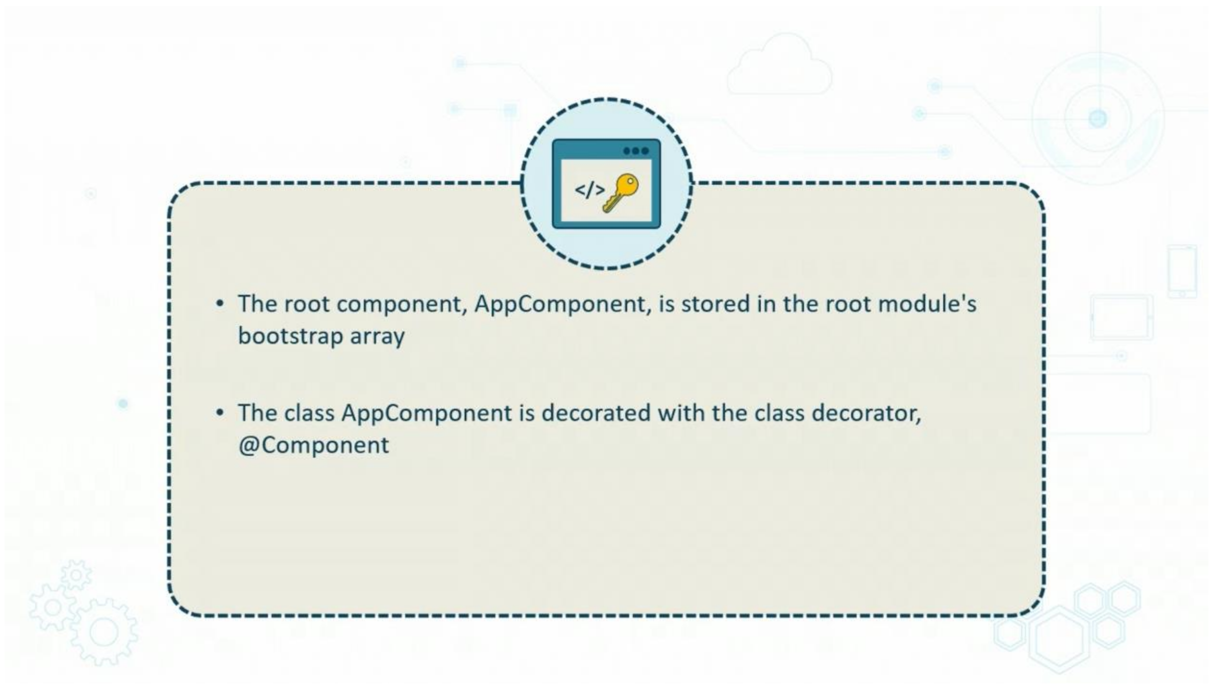
Describes the injectable modules and services

Identifies the module's component, directives, and pipes



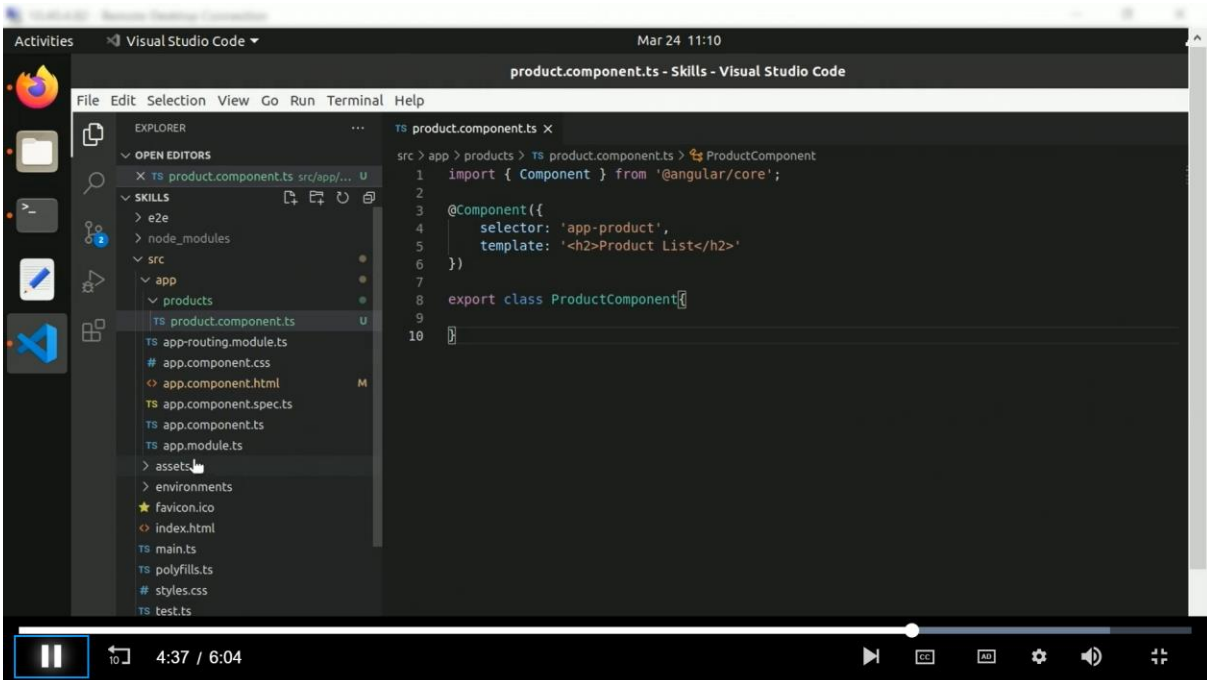
Bootstrapping Components Using Modules

- The application is launched by bootstrapping the root AppModule
- Bootstrapping creates and inserts the components into the browser's DOM
- The bootstrapped component becomes the base of its tree of components

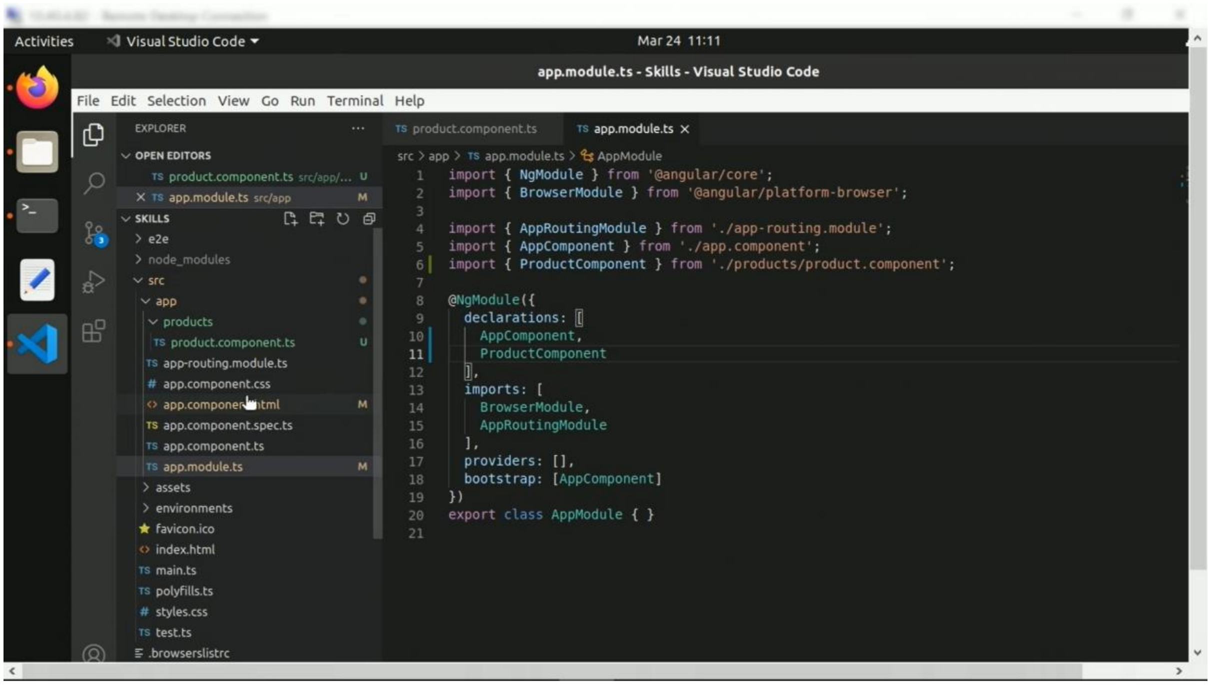


- The root component, AppComponent, is stored in the root module's bootstrap array
- The class AppComponent is decorated with the class decorator, @Component

Created this Product.component.ts by manual

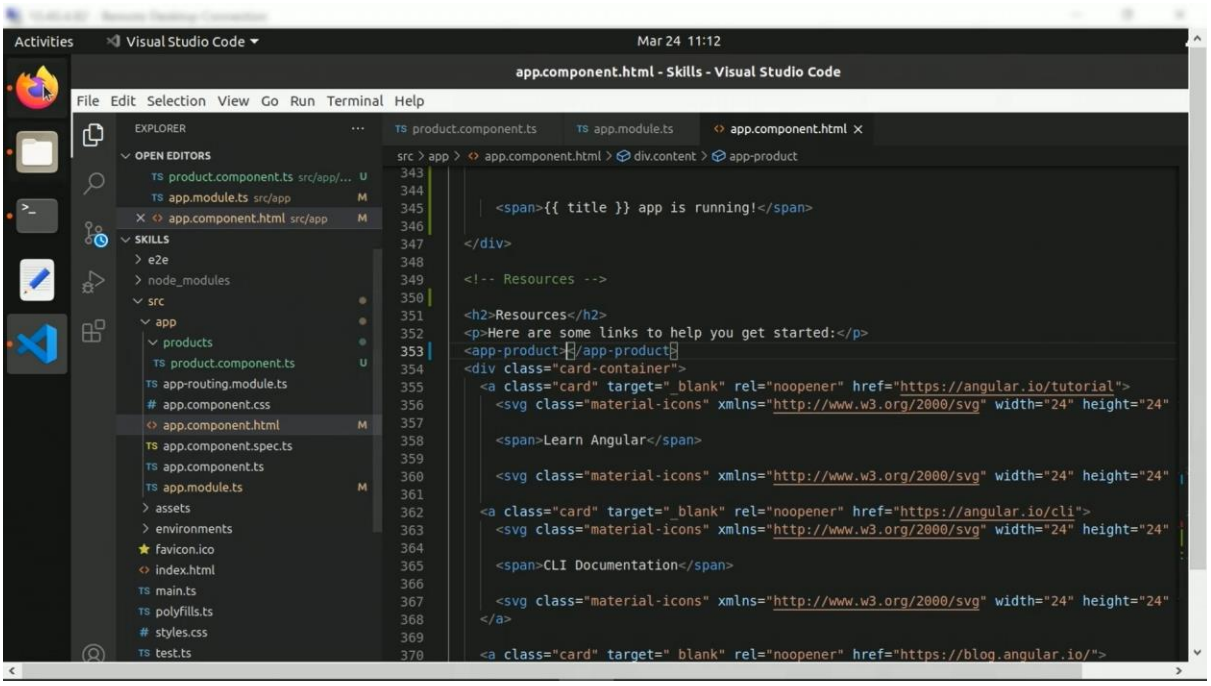


```typescript
import { Component } from '@angular/core';

@Component({
    selector: 'app-product',
    template: '<h2>Product List</h2>'
})

export class ProductComponent{

}
```

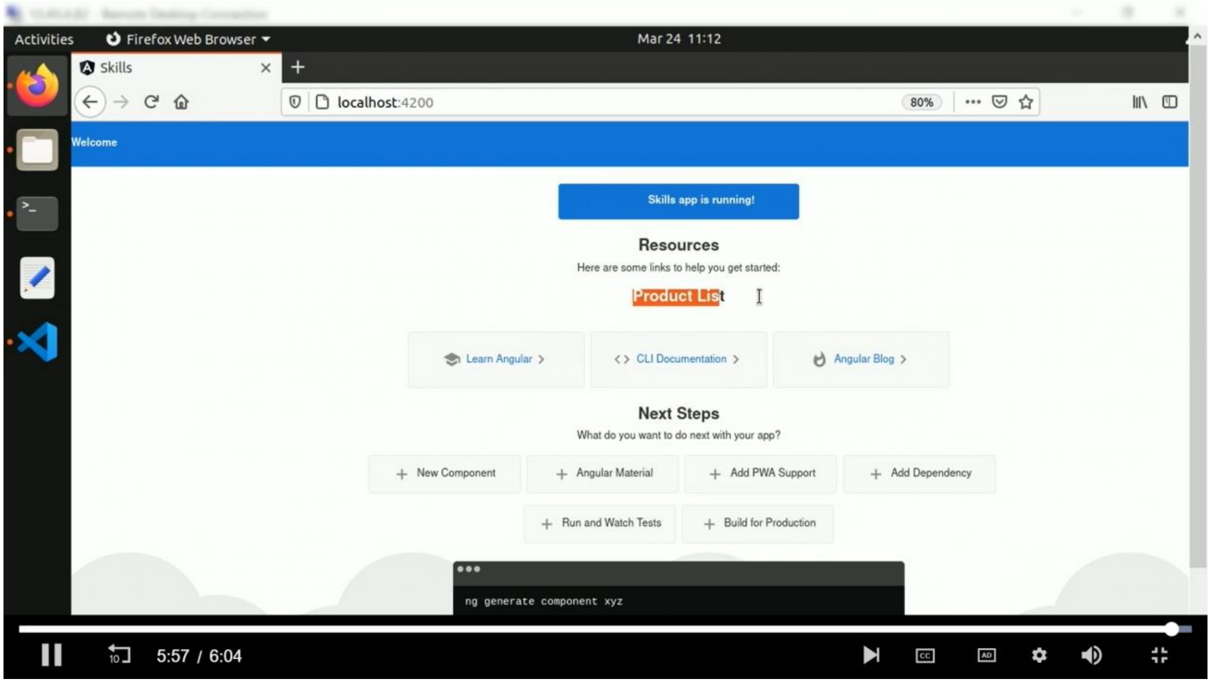


Activities
Visual Studio Code
Mar 24 11:11
app.module.ts - Skills - Visual Studio Code
File Edit Selection View Go Run Terminal Help
EXPLORER
OPEN EDITORS
SKILLS
product.component.ts
app.module.ts
src > app > app.module.ts > AppModule

```typescript
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { ProductComponent } from './products/product.component';

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



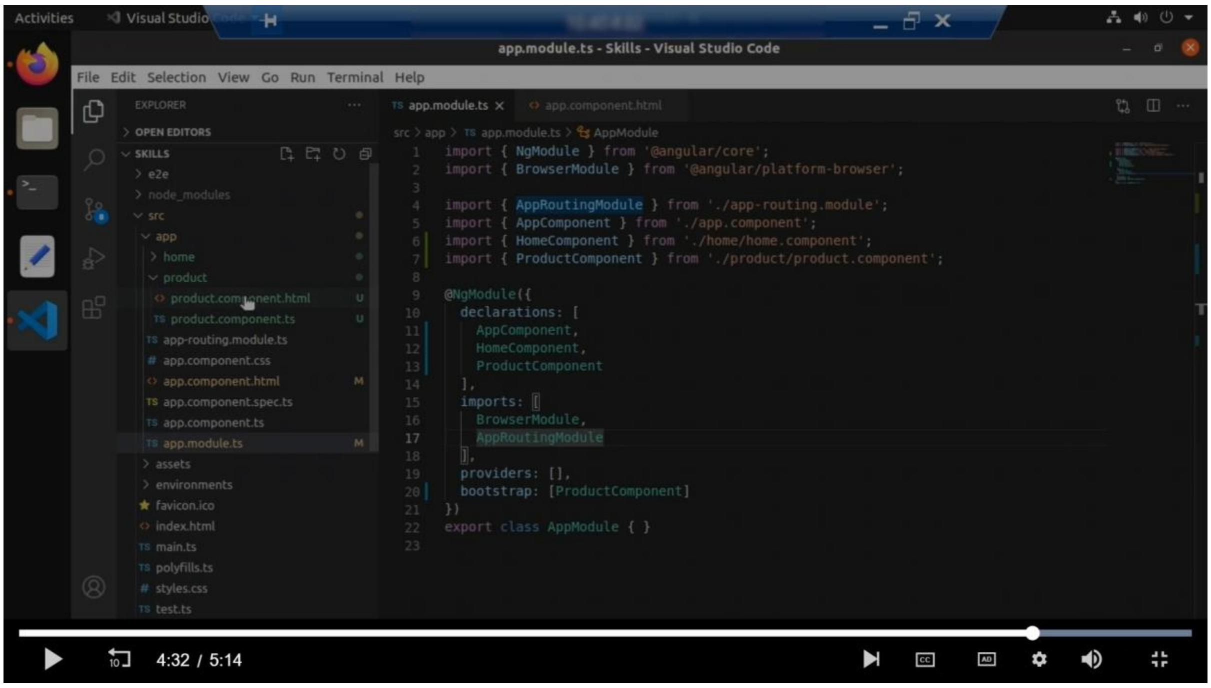
```html
      <span>{{ title }} app is running!</span>

  </div>

  <!-- Resources -->

  <h2>Resources</h2>
  <p>Here are some links to help you get started:</p>
  <app-product></app-product>
  <div class="card-container">
    <a class="card" target="_blank" rel="noopener" href="https://angular.io/tutorial">
      <svg class="material-icons" xmlns="http://www.w3.org/2000/svg" width="24" height="24"

      <span>Learn Angular</span>

      <svg class="material-icons" xmlns="http://www.w3.org/2000/svg" width="24" height="24"

    <a class="card" target="_blank" rel="noopener" href="https://angular.io/cli">
      <svg class="material-icons" xmlns="http://www.w3.org/2000/svg" width="24" height="24"

      <span>CLI Documentation</span>

      <svg class="material-icons" xmlns="http://www.w3.org/2000/svg" width="24" height="24"
    </a>

    <a class="card" target="_blank" rel="noopener" href="https://blog.angular.io/">
```



Bootstrapping components using Modules

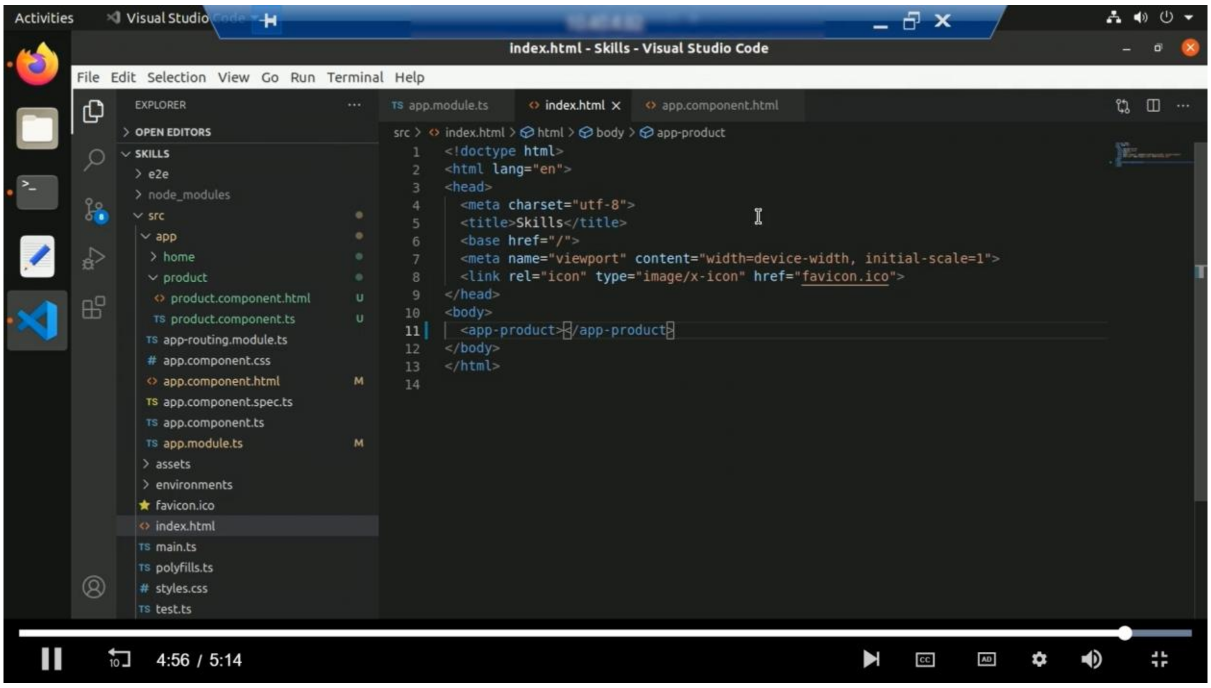
Boostrap is the Entry Point of anular default page

App.module.ts file

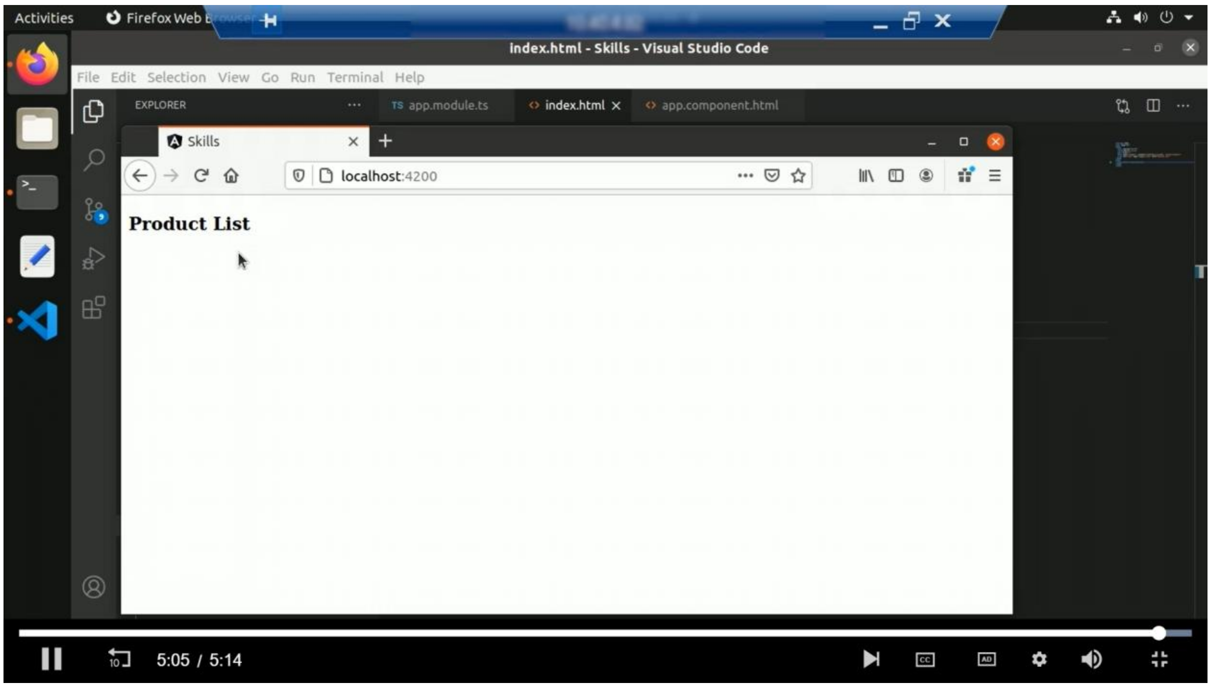

app.module.ts - Skills - Visual Studio Code
File Edit Selection View Go Run Terminal Help
4:32 / 5:14

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';
import { HomeComponent } from './home/home.component';
import { ProductComponent } from './product/product.component';

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

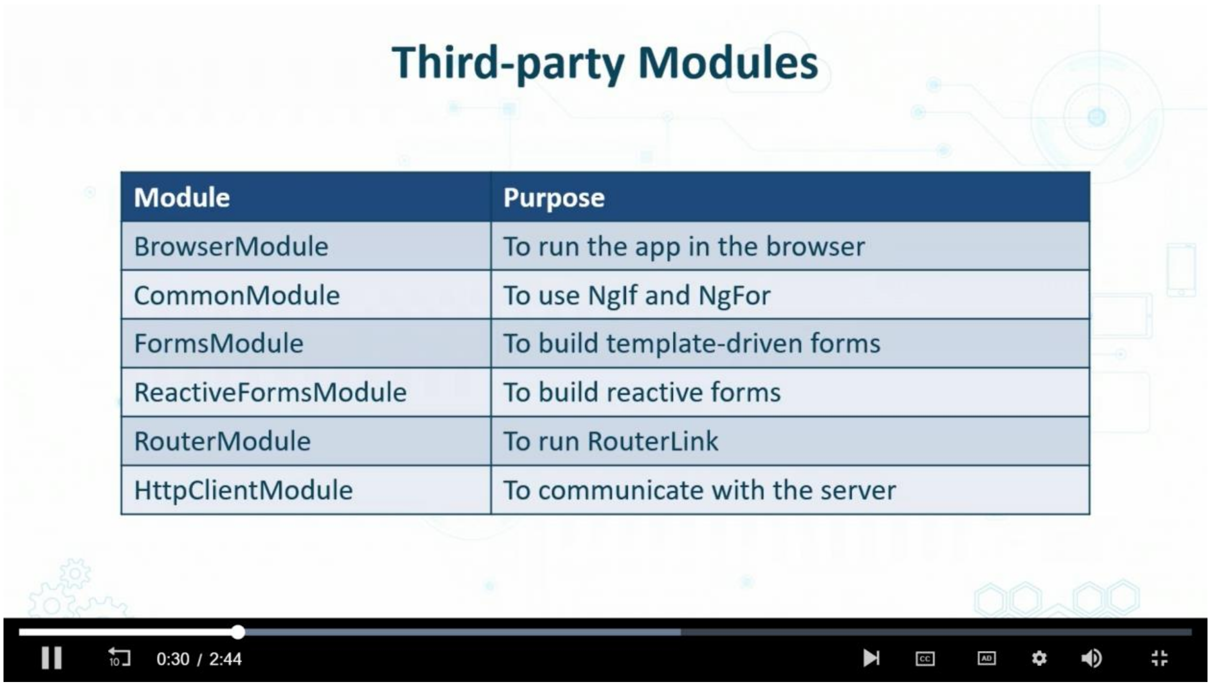
Change in index.html page


index.html - Skills - Visual Studio Code
File Edit Selection View Go Run Terminal Help
4:56 / 5:14

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Skills</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-product></app-product>
</body>
</html>
```

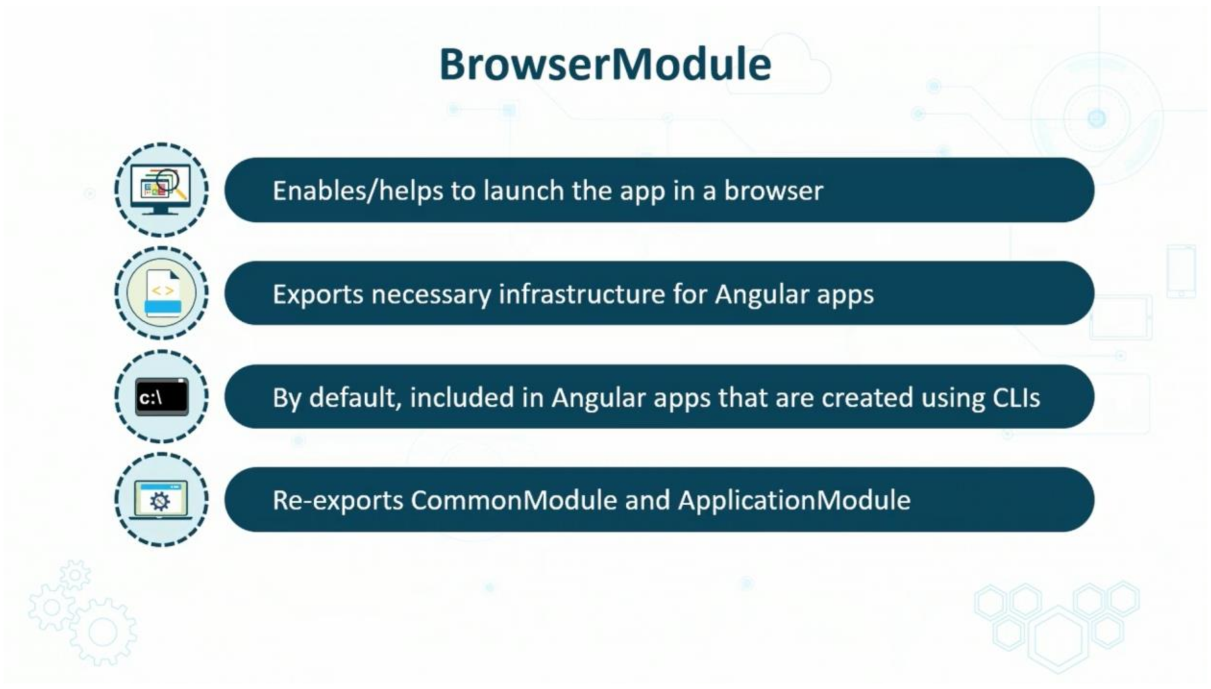


Product List



Third-party Modules

Module	Purpose
BrowserModule	To run the app in the browser
CommonModule	To use NgIf and NgFor
FormsModule	To build template-driven forms
ReactiveFormsModule	To build reactive forms
RouterModule	To run RouterLink
HttpClientModule	To communicate with the server



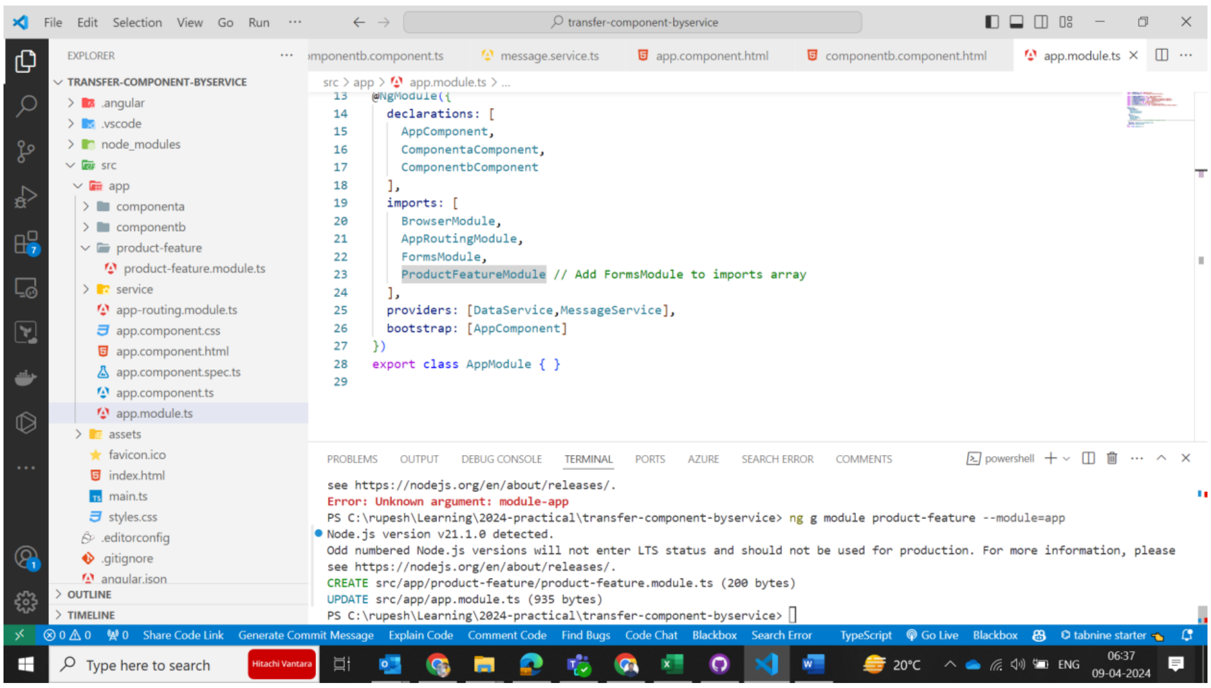
BrowserModule

- Enables/helps to launch the app in a browser
- Exports necessary infrastructure for Angular apps
- By default, included in Angular apps that are created using CLIs
- Re-exports CommonModule and ApplicationModule

Feature module

Helps organise your code and other modules components

Ng g module product-feature –module=app



```
@NgModule({
  declarations: [
    AppComponent,
    ComponentaComponent,
    ComponentbComponent
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ProductFeatureModule // Add FormsModule to imports array
  ],
  providers: [DataService,MessageService],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

```
see https://nodejs.org/en/about/releases/.
Error: Unknown argument: module-app
PS C:\rupesh\Learning\2024-practical\transfer-component-byservice> ng g module product-feature --module=app
Node.js version v21.1.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please
see https://nodejs.org/en/about/releases/.
CREATE src/app/product-feature/product-feature.module.ts (200 bytes)
UPDATE src/app/app.module.ts (935 bytes)
PS C:\rupesh\Learning\2024-practical\transfer-component-byservice>
```

Product Module:

```
// product.module.ts
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ProductListComponent } from './product-list/product-list.component';
import { ProductDetailComponent } from './product-detail/product-detail.component';
import { ProductService } from './product.service';

@NgModule({
  declarations: [
    ProductListComponent,
    ProductDetailComponent
  ],
  imports: [
    CommonModule
  ],
  providers: [ProductService],
  exports: [
    ProductListComponent,
    ProductDetailComponent
  ]
})
export class ProductModule { }
```

Category Module:

- The Category Module would contain components, services, and other features related to managing product categories.

```
// category.module.ts
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { CategoryListComponent } from './category-list/category-list.component';
import { CategoryDetailComponent } from './category-detail/category-
detail.component';
import { CategoryService } from './category.service';

@NgModule({
  declarations: [
    CategoryListComponent,
    CategoryDetailComponent
  ],
  imports: [
    CommonModule
  ],
  providers: [CategoryService],
  exports: [
    CategoryListComponent,
    CategoryDetailComponent
  ]
})
export class CategoryModule { }
```

Common Module:

- The Common Module would contain reusable components, directives, pipes, and services that are shared across multiple modules.

```
// common.module.ts
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { HeaderComponent } from './header/header.component';
import { FooterComponent } from './footer/footer.component';

@NgModule({
  declarations: [
    HeaderComponent,
    FooterComponent
  ],
  imports: [
    CommonModule
  ],
  exports: [
    HeaderComponent,
    FooterComponent
  ]
})
export class CommonModule { }
```

App Module:

- The App Module would be the root module of the application, and it would import and configure the Product Module, Category Module, and Common Module.

```
// app.module.ts
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';
import { AppComponent } from './app.component';
import { ProductModule } from './product/product.module';
import { CategoryModule } from './category/category.module';
import { CommonModule } from './common/common.module';

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

App Component Template: In your root component's template (usually `app.component.html`), you can include the components from the different modules using their selector tags.

```
<!-- app.component.html -->
<app-header></app-header>
<router-outlet></router-outlet> <!-- This is for routing -->
<app-footer></app-footer>
```

1. **Routing Configuration (if applicable):** If your application uses routing, you'll configure the routes in the `AppRoutingModule` or directly in the `AppModule`. Each route can load a component from any module.

   typescriptCopy code

   ```
   import NgModule from '@angular/core' import RouterModule Routes from '@angular/router' import HomeComponent from './home/home.component' import ProductListComponent from './product/product-list/product-list.component' import CategoryListComponent from './category/category-list/category-list.component' const routes Routes path '' component HomeComponent path 'products' component ProductListComponent path 'categories' component CategoryListComponent imports RouterModule forRoot exports RouterModule export class AppRoutingModule
   ```

   Ensure that your `AppModule` imports `AppRoutingModule` to enable routing.
2. **Lazy Loading (optional):** For larger applications, you may want to consider lazy-loading feature modules to improve initial loading times. Lazy-loaded modules are loaded asynchronously only when the user navigates to their routes.

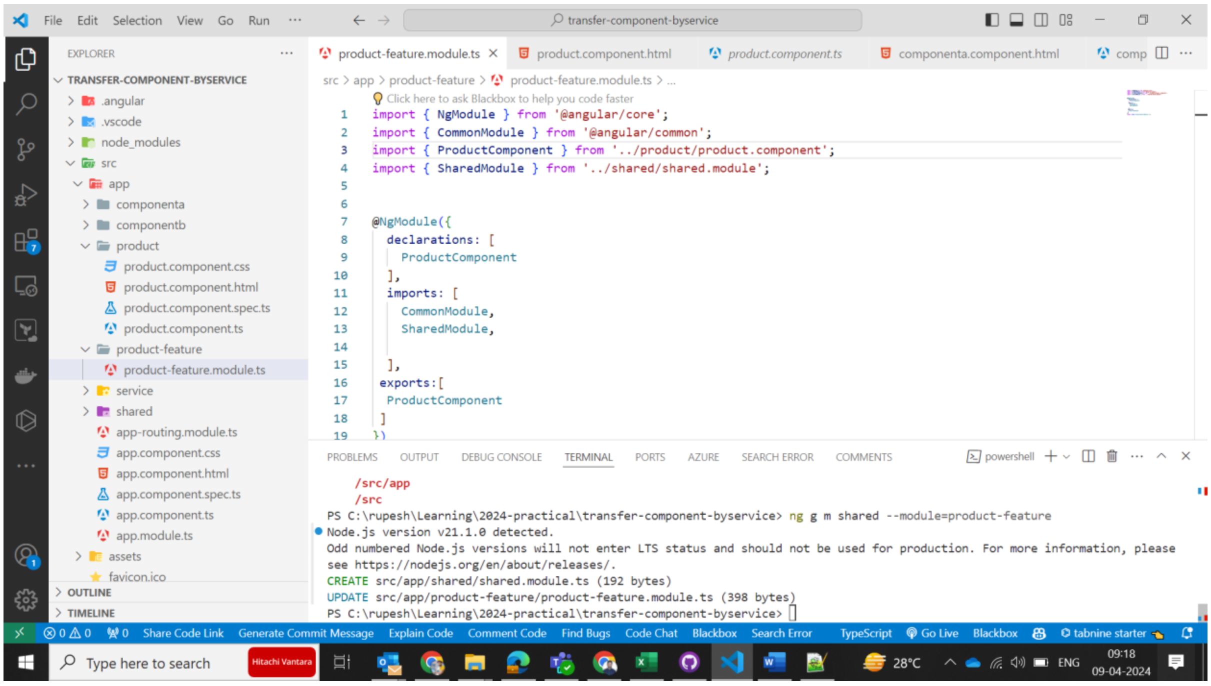
   typescriptCopy code

   ```
   const routes Routes path '' component HomeComponent path 'products' loadChildren import './product/product.module' then path 'categories' loadChildren import './category/category.module' then
   ```

ng g m shared --module=products-feature

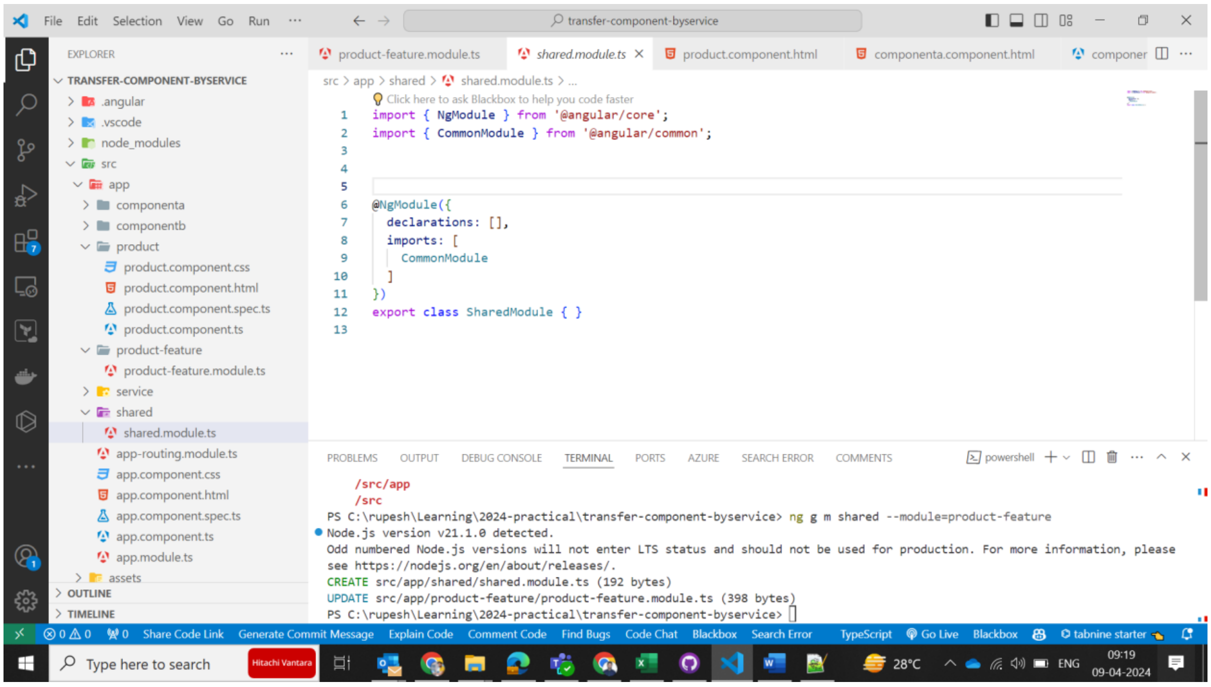
created and updated

shared is used for shared resource


transfer-component-byservice
product-feature.module.ts
product.component.html
product.component.ts
componenta.component.html
src > app > product-feature > product-feature.module.ts > ...
Click here to ask Blackbox to help you code faster
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
SEARCH ERROR
COMMENTS
powershell

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';
import { ProductComponent } from '../product/product.component';
import { SharedModule } from '../shared/shared.module';


@NgModule({
  declarations: [
    ProductComponent
  ],
  imports: [
    CommonModule,
    SharedModule,

  ],
 exports:[
  ProductComponent
 ]
})
```

```
    /src/app
    /src
PS C:\rupesh\Learning\2024-practical\transfer-component-byservice> ng g m shared --module=product-feature
Node.js version v21.1.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please
see https://nodejs.org/en/about/releases/.
CREATE src/app/shared/shared.module.ts (192 bytes)
UPDATE src/app/product-feature/product-feature.module.ts (398 bytes)
PS C:\rupesh\Learning\2024-practical\transfer-component-byservice>
```


transfer-component-byservice
product-feature.module.ts
shared.module.ts
product.component.html
componenta.component.html
src > app > shared > shared.module.ts > ...
Click here to ask Blackbox to help you code faster
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
AZURE
SEARCH ERROR
COMMENTS
powershell

```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';



@NgModule({
  declarations: [],
  imports: [
    CommonModule
  ]
})
export class SharedModule { }

```

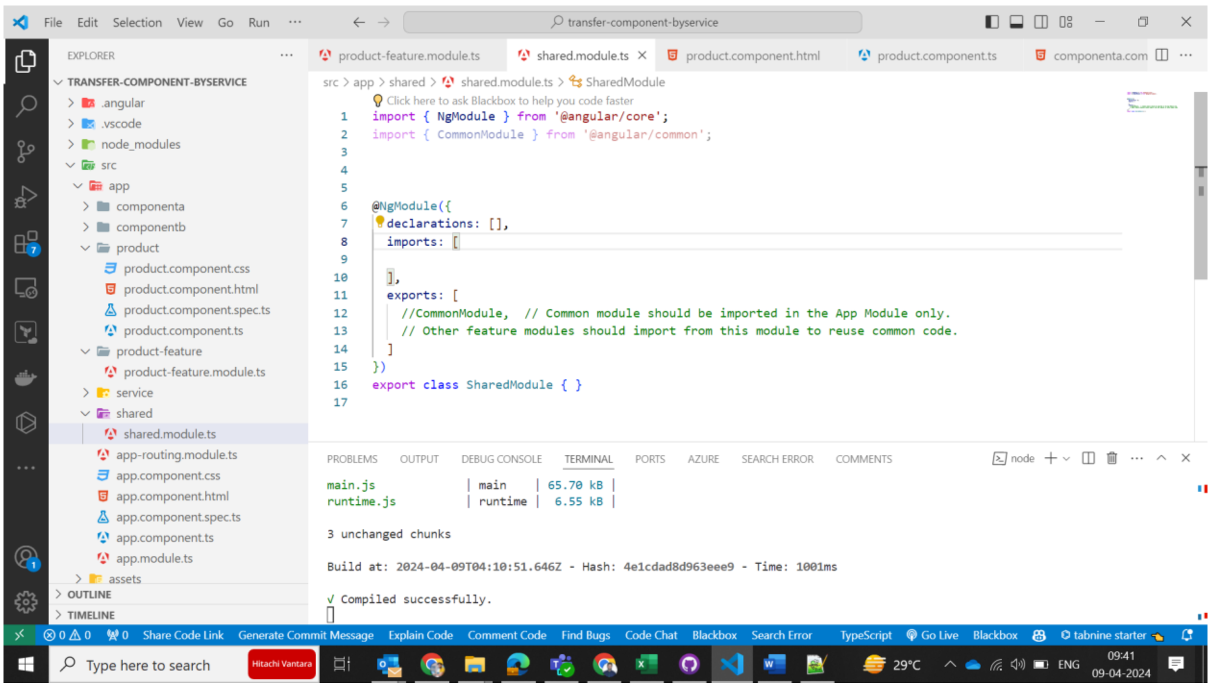
```
    /src/app
    /src
PS C:\rupesh\Learning\2024-practical\transfer-component-byservice> ng g m shared --module=product-feature
Node.js version v21.1.0 detected.
Odd numbered Node.js versions will not enter LTS status and should not be used for production. For more information, please
see https://nodejs.org/en/about/releases/.
CREATE src/app/shared/shared.module.ts (192 bytes)
UPDATE src/app/product-feature/product-feature.module.ts (398 bytes)
PS C:\rupesh\Learning\2024-practical\transfer-component-byservice>
```

Delete CommonModule bcs now available from shared already

Exports used to export for use this for another component so this is available for othrs also

```
  exports: [
    CommonModule,  // Common module should be imported in the App Module only.
    // Other feature modules should import from this module to reuse common code.
  ]
```

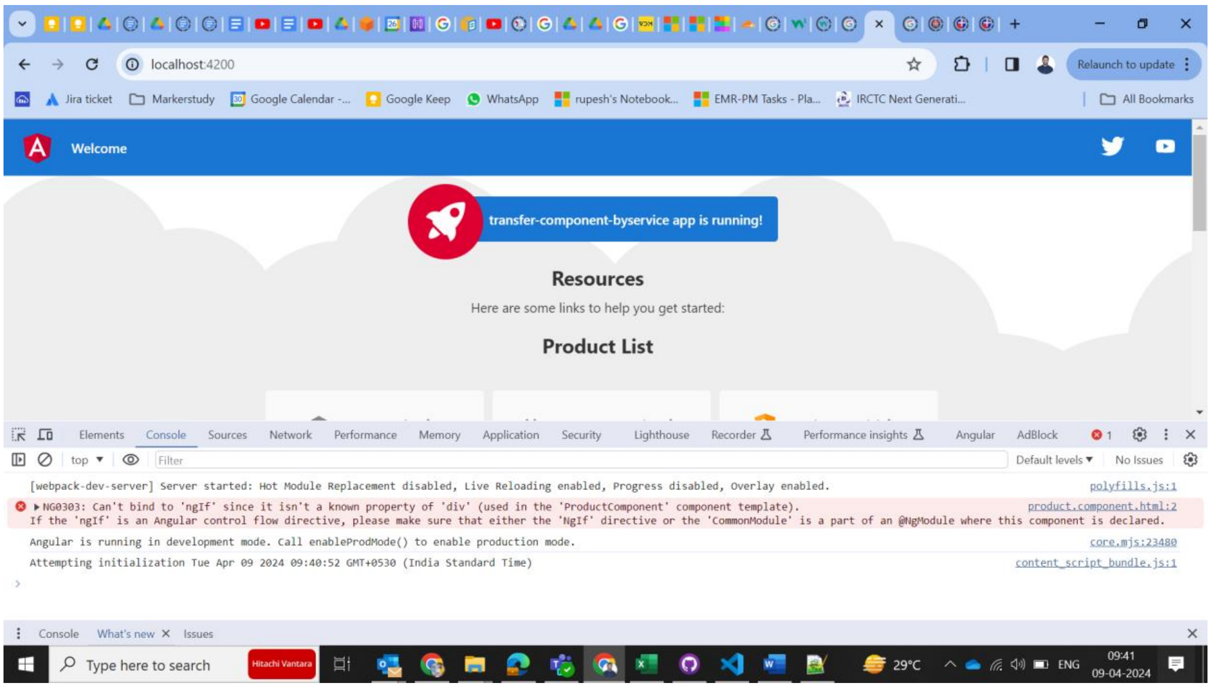
I commented from shared module



```
import { NgModule } from '@angular/core';
import { CommonModule } from '@angular/common';



@NgModule({
  declarations: [],
  imports: [

  ],
  exports: [
    //CommonModule,  // Common module should be imported in the App Module only.
    // Other feature modules should import from this module to reuse common code.
  ]
})
export class SharedModule { }
```

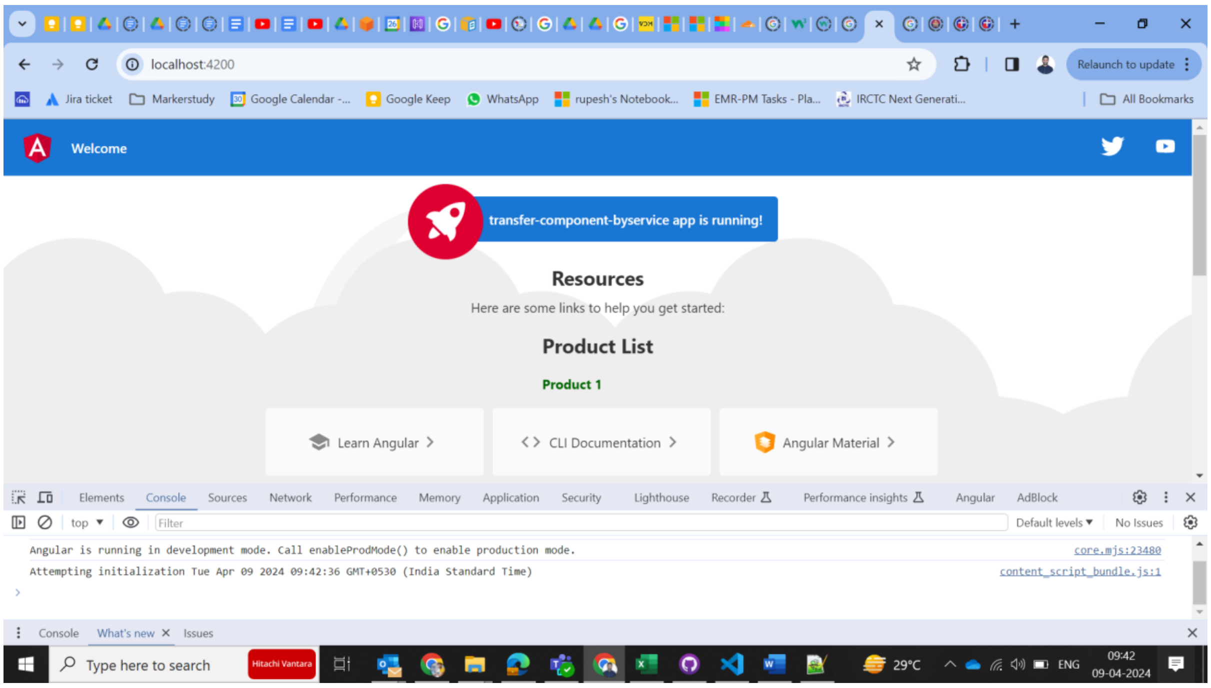
```
main.js                | main      | 65.70 kB |
runtime.js             | runtime   |  6.55 kB |

3 unchanged chunks

Build at: 2024-04-09T04:10:51.646Z - Hash: 4e1cdad8d963eee9 - Time: 1001ms

√ Compiled successfully.
```

See error


localhost:4200
Welcome
transfer-component-byservice app is running!
Resources
Here are some links to help you get started:
Product List

```
[webpack-dev-server] Server started: Hot Module Replacement disabled, Live Reloading enabled, Progress disabled, Overlay enabled.   polyfills.js:1
▶NG0303: Can't bind to 'ngIf' since it isn't a known property of 'div' (used in the 'ProductComponent' component template).   product.component.html:2
If the 'ngIf' is an Angular control flow directive, please make sure that either the 'NgIf' directive or the 'CommonModule' is a part of an @NgModule where this component is declared.
Angular is running in development mode. Call enableProdMode() to enable production mode.   core.mjs:23480
Attempting initialization Tue Apr 09 2024 09:40:52 GMT+0530 (India Standard Time)   content_script_bundle.js:1
```

Its working



```
Angular is running in development mode. Call enableProdMode() to enable production mode.   core.mjs:23480
Attempting initialization Tue Apr 09 2024 09:42:36 GMT+0530 (India Standard Time)   content_script_bundle.js:1
```

Y can import from product and shared module

Working with Export/import Array

1. Create Array List randomgen.js

```
genRandomNumbers = function ()
{
    let rNum = []
    for (i = 0; i < 10; i++)
    {
        let rnd = Math.floor(Math.random() * 999 + 1);  //generates a random
number between 0 and 49
        rNum.push(rnd)
    }
    return rNum;
}
```

2. Angular.json update

```
 "scripts": [
              "src/assets/js/randomgen.js"
            ]
```

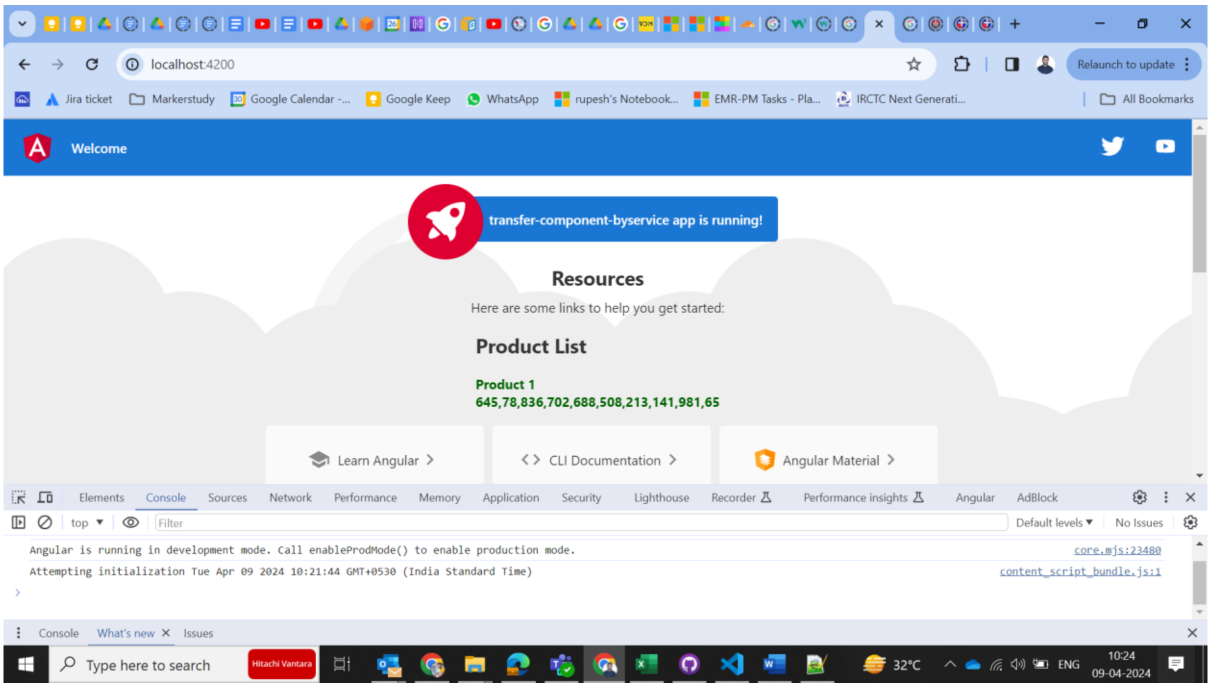
3. Product.com.ts

```
declare const genRandomNumbers:any;



export class ProductComponent {
  showDiv = true;
  rNum=<[]>genRandomNumbers();
}
```

4. Product.html

```
<div>
                    {{rNum}}
               </div>
```

Output



transfer-component-byservice app is running!

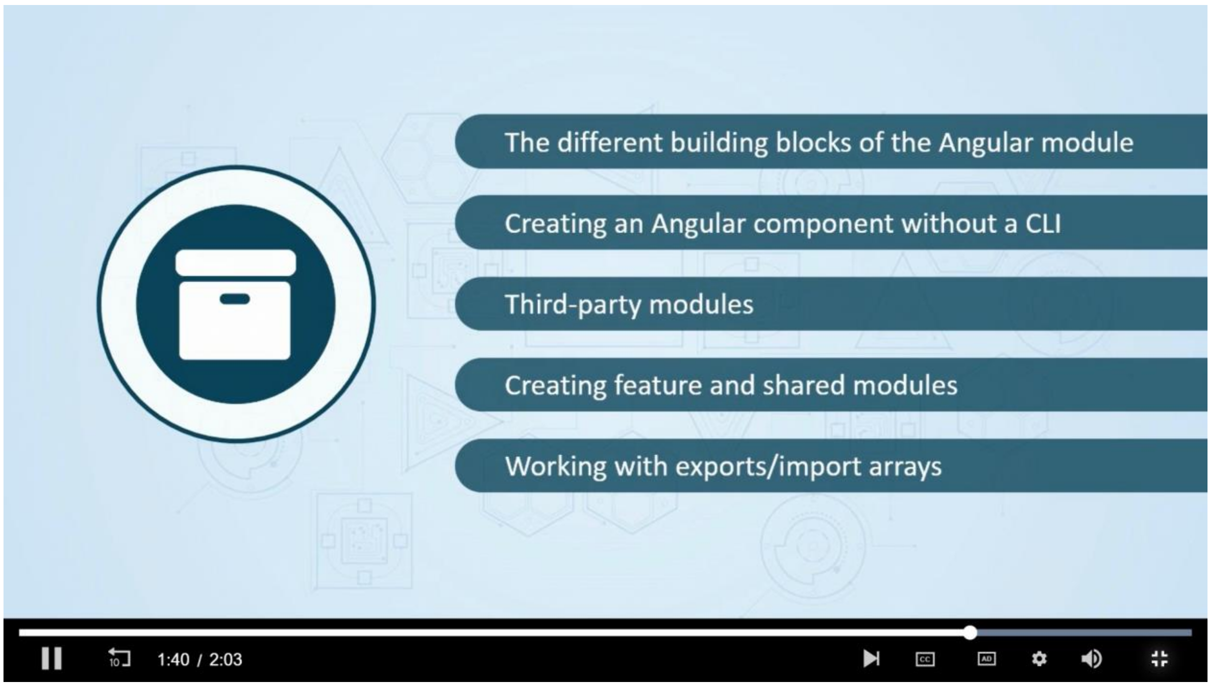
Resources

Here are some links to help you get started:

Product List

Product 1
645,78,836,702,688,508,213,141,981,65

```
Angular is running in development mode. Call enableProdMode() to enable production mode.        core.mjs:23480
Attempting initialization Tue Apr 09 2024 10:21:44 GMT+0530 (India Standard Time)    content_script_bundle.js:1
```

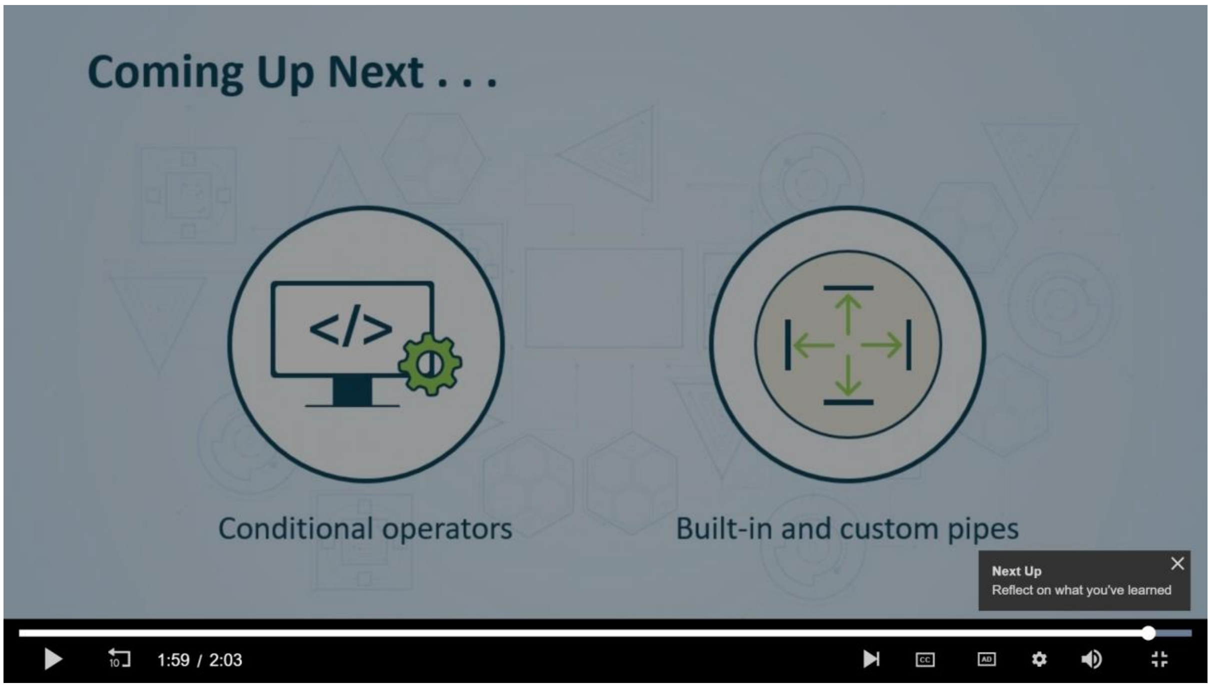

The different building blocks of the Angular module

Creating an Angular component without a CLI

Third-party modules

Creating feature and shared modules

Working with exports/import arrays


Coming Up Next . . .

Conditional operators

Built-in and custom pipes