Table of Contents

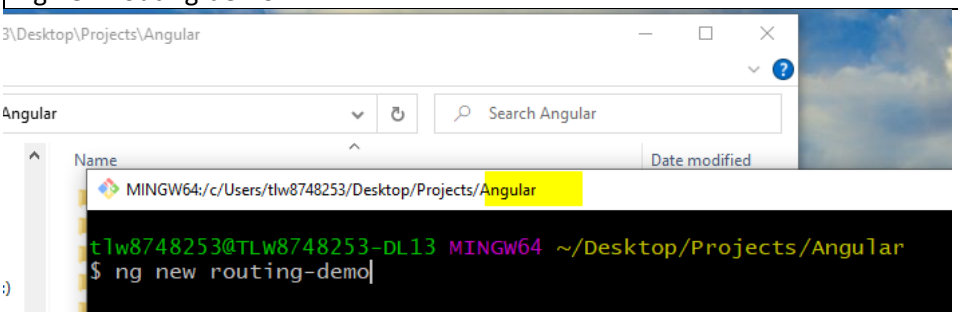
Aug 27th Lecture and Project routing-demo

Create new project routing-demo

Open Git Bash window in your Angular project folder

Type the following to create the project

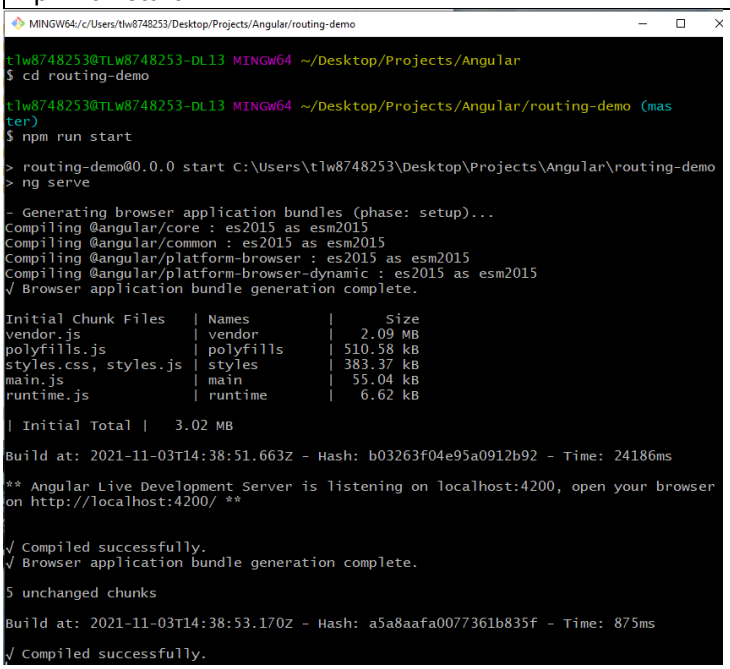
```
ng new routing-demo
```



Start the application

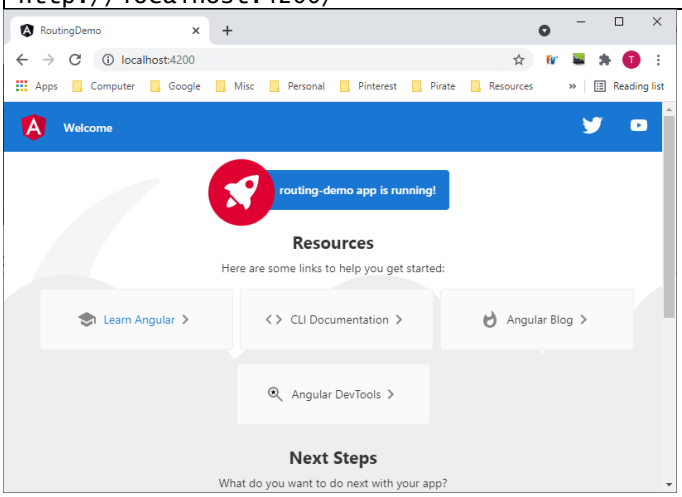
In the Git Bash window in the routing-demo folder type

```
cd routing-demo
npm run start
```

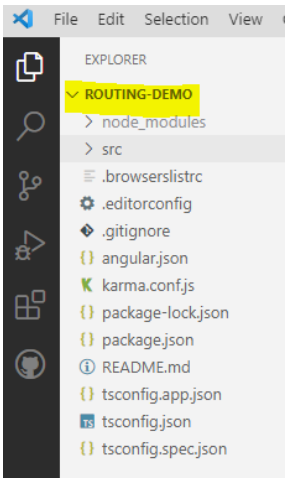


Open the application in a browser

```
http://localhost:4200/
```

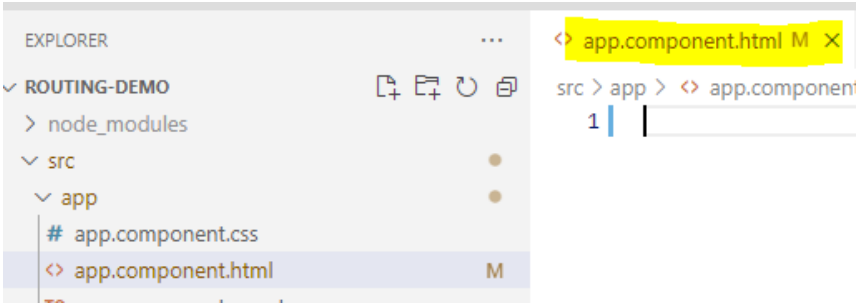


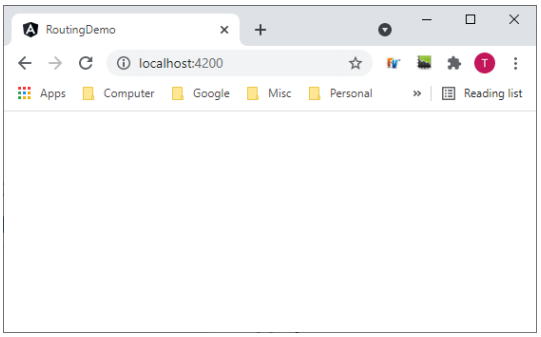
Open the project in VS Code



Modify the default project created

Remove all the html in app.component.html





Adding routing to the Angular project

Tutorial on adding routing

https://angular.io/tutorial/toh-pt5

Add the `AppRoutingModule`

In Angular, the best practice is to load and configure the router in a separate, top-level module that is dedicated to routing and imported by the root `AppModule`.

By convention, the module class name is `AppRoutingModule` and it belongs in the `app-routing.module.ts` in the `src/app` folder.

Use the CLI to generate it.

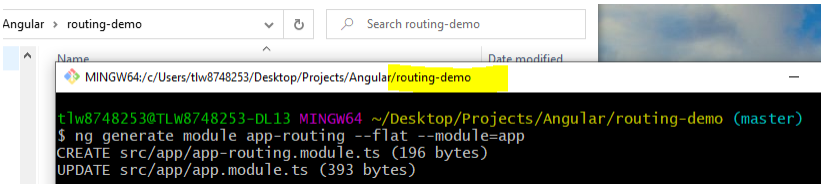
```
ng generate module app-routing --flat --module=app
```

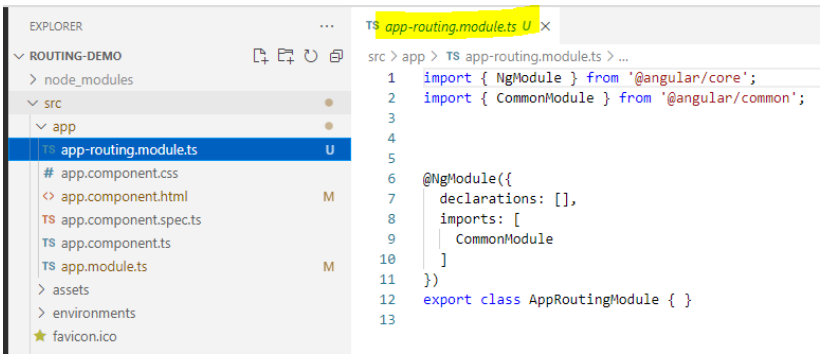
> `--flat` puts the file in `src/app` instead of its own folder.
> `--module=app` tells the CLI to register it in the `imports` array of the `AppModule`.

Create the routing module

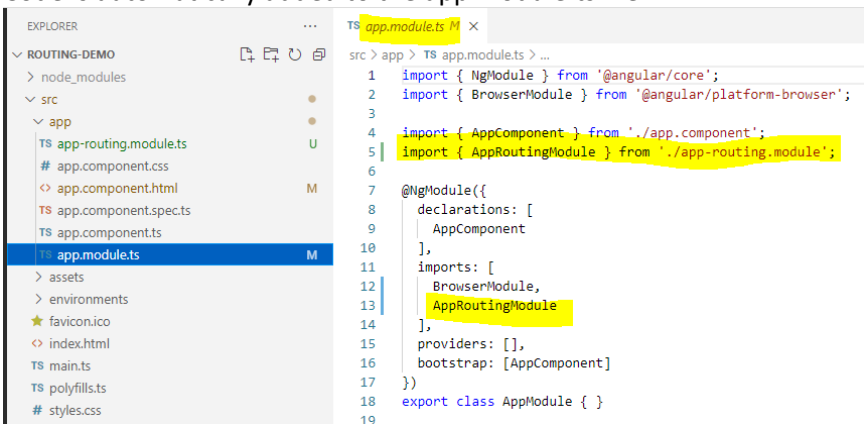
In a new Git Bash window in the routing-demo folder type

```
ng generate module app-routing --flat --module=app
```





Code is automatically added to the app.module.ts file



Update the app-routing.module.ts file

For some reason the default code generated is not correct. It needs to change according to the tutorial.

The generated file looks like this:

src/app/app-routing.module.ts (generated)

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

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

Replace it with the following:

src/app/app-routing.module.ts (updated)

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { HeroesComponent } from './heroes/heroes.component';

const routes: Routes = [
  { path: 'heroes', component: HeroesComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Update the app-routing-module.ts code as describe in the tutorial.

Without the HeroesComponent code.

Replace all the code with:

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';

const routes: Routes = [];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

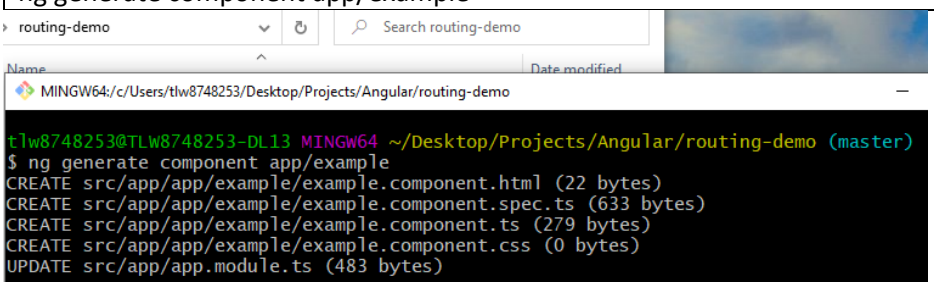
The following line of code from the above is important for defining routing and will be updated shortly

```
const routes: Routes = [];
```

Generate a new components for routing example

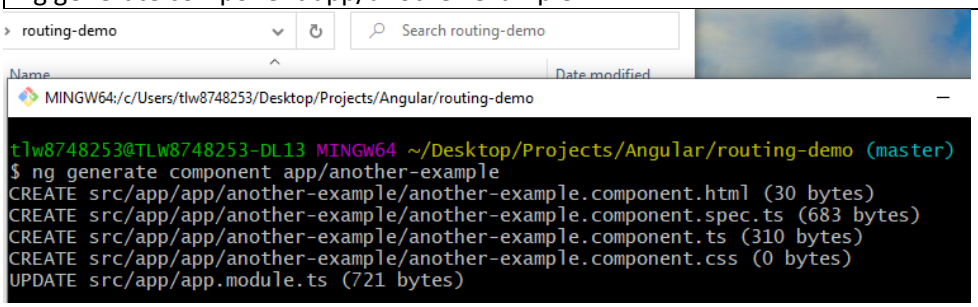
In the Git Bash project window type for an example component

```
ng generate component app/example
```

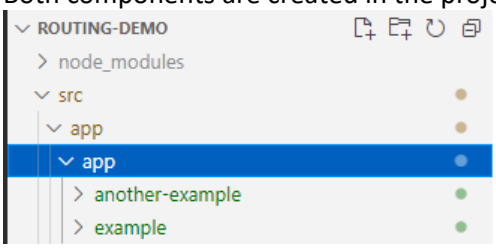


In the Git Bash project window type for anotherexample component

```
ng generate component app/another-example
```



Both components are created in the project



Add routing components to the projects

Add routing components to app-routing.module.ts

Add:

```
…
import { AnotherExampleComponent } from './app/another-example/another-example.component';
import { ExampleComponent } from './app/example/example.component';
…
  { path: 'example', component: ExampleComponent },
  { path: 'anotherexample', component: AnotherExampleComponent }
```

```
TS app-routing.module.ts U ×
src > app > TS app-routing.module.ts > AppRoutingModule
1  import { NgModule } from '@angular/core';
2  import { RouterModule, Routes } from '@angular/router';
3  import { AnotherExampleComponent } from './app/another-example/another-example.component';
4  import { ExampleComponent } from './app/example/example.component';
5
6  const routes: Routes = [
7    { path: 'example', component: ExampleComponent },
8    { path: 'anotherexample', component: AnotherExampleComponent }
9  ];
10
11 @NgModule({
12   imports: [RouterModule.forRoot(routes)],
13   exports: [RouterModule]
14 })
15 export class AppRoutingModule { }
```

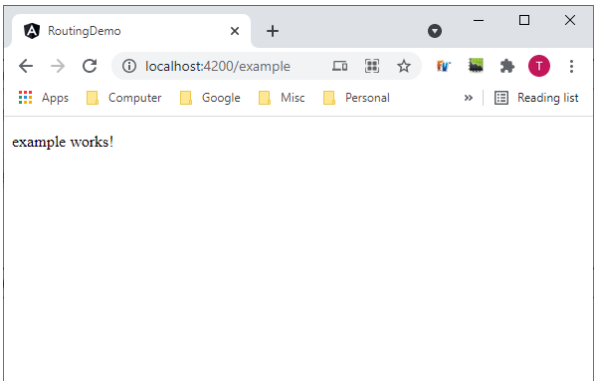
Add routing tag to app.component.html

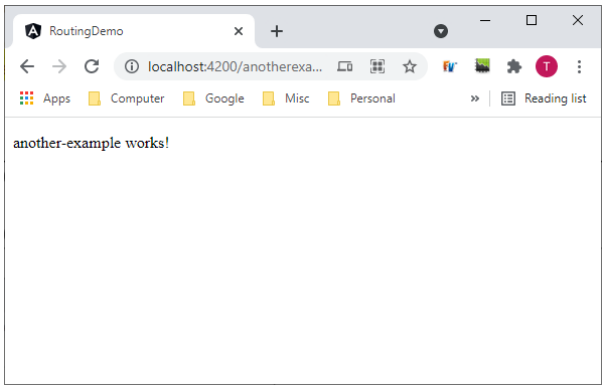
```
<router-outlet></router-outlet>
```

```
app.component.html M ×
src > app > app.component.html > router-outlet
1  <router-outlet></router-outlet>
```

Verify the update in the html by using the following URL

```
http://localhost:4200/example
then:
http://localhost:4200/anotherexample
```





Add a Navigation Bar to the project

In the Git Bash window type the following to create a navigation component

```
ng generate component nav
```

```
MINGW64:/c/Users/tlw8748253/Desktop/Projects/Angular/routing-demo
tlw8748253@TLW8748253-DL13 MINGW64 ~/Desktop/Projects/Angular/routing-demo (master)
$ ng generate component nav
CREATE src/app/nav/nav.component.html (18 bytes)
CREATE src/app/nav/nav.component.spec.ts (605 bytes)
CREATE src/app/nav/nav.component.ts (263 bytes)
CREATE src/app/nav/nav.component.css (0 bytes)
UPDATE src/app/app.module.ts (673 bytes)
```

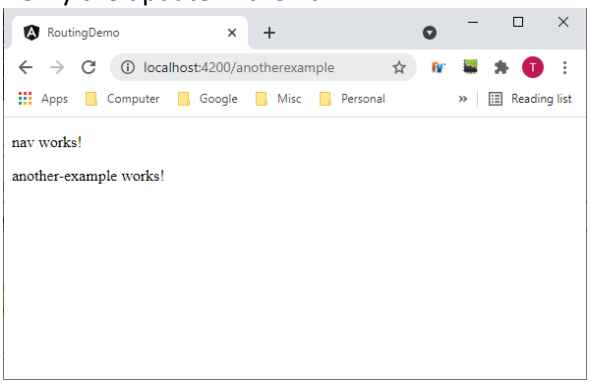
Add the component to the html

Add nav tag to app.component.html

```
<app-nav></app-nav>
```

```
app.component.html M
src > app > app.component.html > ...
1  <app-nav></app-nav>
2
3  <router-outlet></router-outlet>
```

Verify the update in the html



Use Bulma for the CSS global styling

Install Bulma for this project

In the Git Bash window type

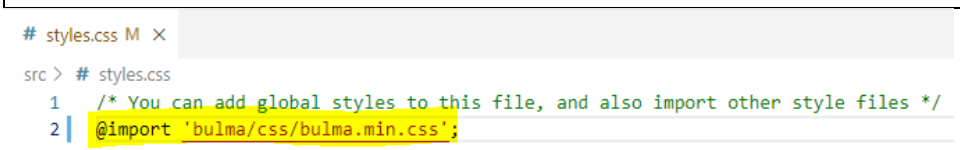
```
npm install bulma
```

```
tlw8748253@TLW8748253-DL13 MINGW64 ~/Desktop/Projects/Angular/routing-demo (master)
$ npm install bulma
```

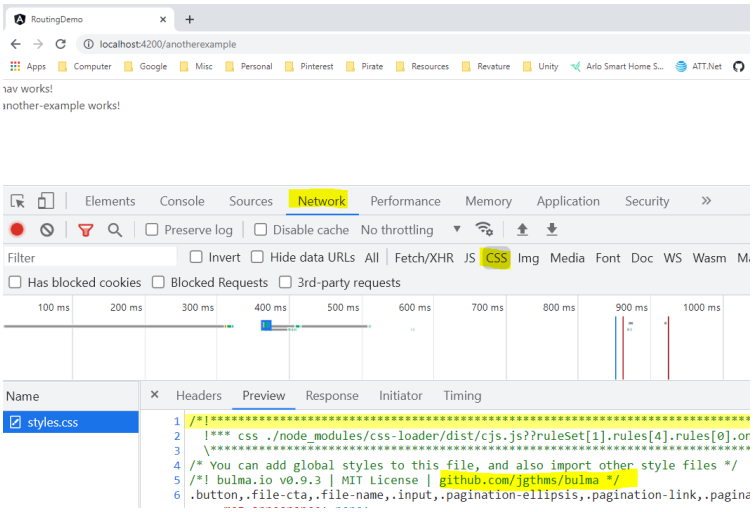
Add Bulma to the global CSS file

Add the following line to styles.css

```
@import 'bulma/css/bulma.min.css';
```



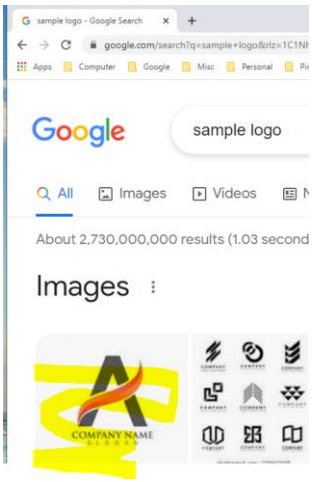
You can verify that Bulma is included by inspecting the webpage, selecting Network then CSS.



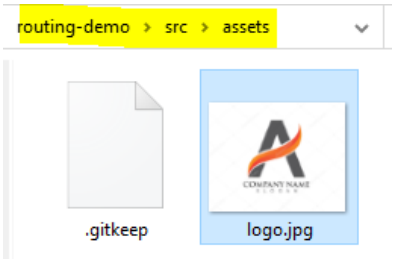
Continue on file updates for this project adding a logo and navigation bar

Copy an image file from an external source

To follow along with the lecture you can do a Google search on “sample logo” and save the sample company name logo.



Save this file to the project routing-demo\src\assets folder with the name logo.jpg.



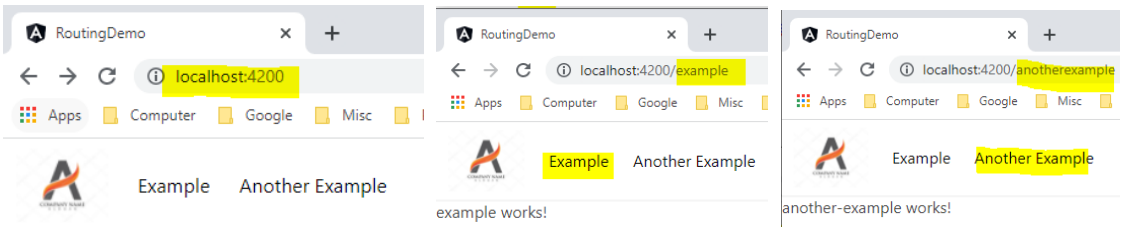
Update the navigation component html

The initial code update demonstrates a behavior that we do not want and will be replaced. It shows the example and anotherexample pages loading as separate pages and has a side effect of restarting the application each time a page is loaded. Instead we want a seamless transition between pages and the appearance of a one page application.

Replace the default html in the nav.component.html with code that include href

```
<nav class="navbar has-shadow is-white">
    <div class="navbar-brand">
      <a href="" class="navbar-item">
        <img src="assets/logo.jpg" alt="Site logo" style="max-height: 60px;">
      </a>
    </div>
    <div class="navbar-menu">
      <div class="navbar-start">
```

```
        <a class="navbar-item" href="/example">Example</a>
        <a class="navbar-item" href="/anotherexample">Another Example</a>
      </div>
    </div>
  </nav>
```



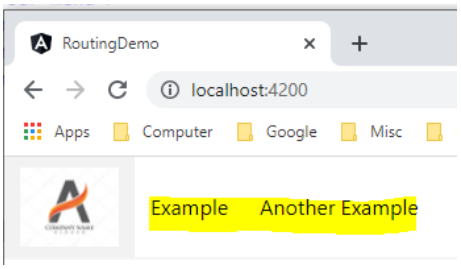
This behavior is caused by using href. Replace href with routerLink.

Update the html in the nav.component.html with code that include routerLink

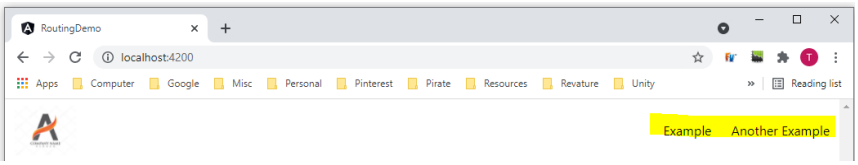
```
<nav class="navbar has-shadow is-white">
    <div class="navbar-brand">
      <a routerLink="" class="navbar-item">
        <img src="assets/logo.jpg" alt="Site logo" style="max-height: 60px;">
      </a>
    </div>
    <div class="navbar-menu">
      <div class="navbar-start">
        <a class="navbar-item" routerLink="/example">Example</a>
        <a class="navbar-item" routerLink="/anotherexample">Another Example</a>
      </div>
    </div>
  </nav>
```

The changes will provide a seamless transition between pages and the appearance of a one page application without reloading the application each time.

Also notice the `<div class="navbar-start">` line. This left justifies the navigation bar.



The code is change to `<div class="navbar-end">` which right justifies the navigation bar.



Create additional project components to demonstrate routing

Create an error component

In the Git Bash window type the following to generate an error component

```
ng generate component error
```

```
MINGW64:/c/Users/tlw8748253/Desktop/Projects/Angular/routing-demo
tlw8748253@TLW8748253-DL13 MINGW64 ~/Desktop/Projects/Angular/routing-demo (master)
$ ng generate component error
CREATE src/app/error/error.component.html (20 bytes)
CREATE src/app/error/error.component.spec.ts (619 bytes)
CREATE src/app/error/error.component.ts (271 bytes)
CREATE src/app/error/error.component.css (0 bytes)
UPDATE src/app/app.module.ts (751 bytes)
```

Add error component to routing module

Add code to app-routing.module.ts for error component

```
…
import { ErrorComponent } from './error/error.component';
…
…,
{ path: '**', component: ErrorComponent }
```

```
src > app > app-routing.module.ts > routes
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { AnotherExampleComponent } from './app/another-example/another-example.component';
import { ExampleComponent } from './app/example/example.component';
import { ErrorComponent } from './error/error.component';

const routes: Routes = [
  { path: 'example', component: ExampleComponent },
  { path: 'anotherexample', component: AnotherExampleComponent },
  { path: '**', component: ErrorComponent }
];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Add an error message to the error html

Replace the default code in error.component.html with

```
<p>You are on a link that doesn't exist</p>
```

```
error.component.html U ×
src > app > error > error.component.html > ...
<p>You are on a link that doesn't exist</p>
```

Create a home component

In the Git Bash window type the following to generate a home component

```
ng generate component home
```

```
MINGW64:/c/Users/tlw8748253/Desktop/Projects/Angular/routing-demo

tlw8748253@TLW8748253-DL13 MINGW64 ~/Desktop/Projects/Angular/routing-demo (master)
$ ng generate component home
CREATE src/app/home/home.component.html (19 bytes)
CREATE src/app/home/home.component.spec.ts (612 bytes)
CREATE src/app/home/home.component.ts (267 bytes)
CREATE src/app/home/home.component.css (0 bytes)
UPDATE src/app/app.module.ts (825 bytes)
```

Add home component to routing module

Add code to app-routing.module.ts for home component

```
…
import { HomeComponent } from './home/home.component';
…
{ path: '', component: HomeComponent },
```

```
TS app-routing.module.ts U ×
src > app > TS app-routing.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { RouterModule, Routes } from '@angular/router';
3  import { AnotherExampleComponent } from './app/another-example/another-example.component';
4  import { ExampleComponent } from './app/example/example.component';
5  import { ErrorComponent } from './error/error.component';
6  import { HomeComponent } from './home/home.component';
7
8  const routes: Routes = [
9    { path: '', component: HomeComponent },
10   { path: 'example', component: ExampleComponent },
11   { path: 'anotherexample', component: AnotherExampleComponent },
12   { path: '**', component: ErrorComponent }
13 ];
14
15 @NgModule({
16   imports: [RouterModule.forRoot(routes)],
17   exports: [RouterModule]
18 })
19 export class AppRoutingModule { }
```

The final Webpage Results

The updates produces the results in the webpage.