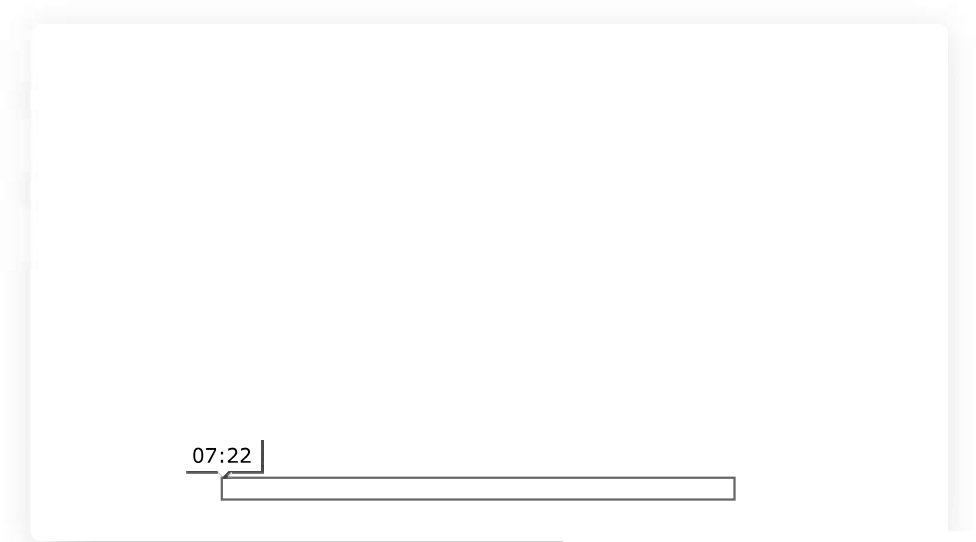


Build Your First Angular Website

Creating an Angula Footer

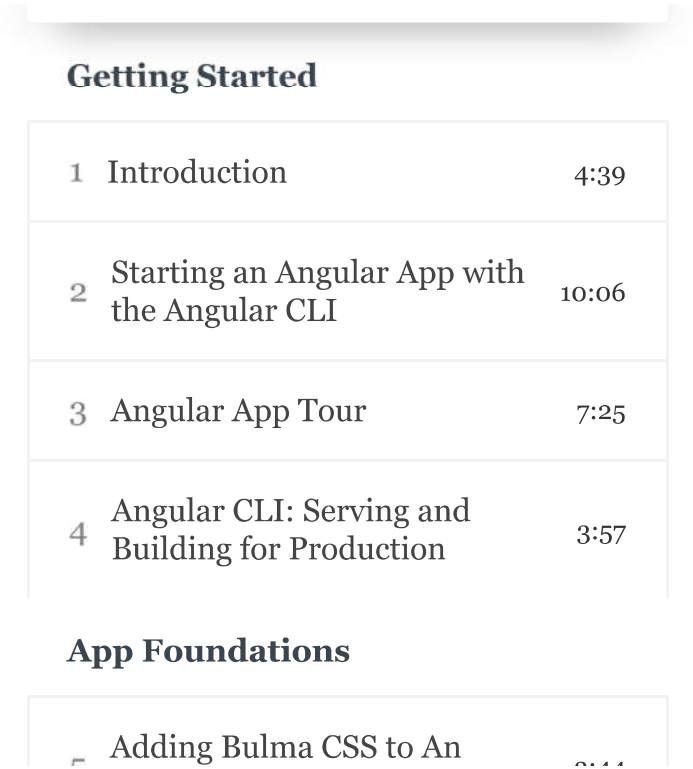
Chris Sevilleja (@chrisoncode) (https://twitter.com

7 Comments Bookmark

E(HTTPS://GITHUB.COM/SEVILAYHA/ANGULAR-FIRST-SITE

1O(HTTPS://SEVILAYHA.GITHUB.IO/ANGULAR-FIRST-SITE-

The header and footer are the two components that we usually start with when building out a new website. Let's go ahead and create these new parts of our site.



Adding a Heac Footer the Wr

Let's talk about how you wou header. We would probably `app.component.ts` since that h application template.

We could add the entire header and footer there. We're going to use some pre-built Bulma components copied over from the Bulma Navbar docs (https://bulma.io/documentation/components/navbar/) and Bulma Footer docs (https://bulma.io/documentation/layout/footer/).

```ts
// src/app/app.component.ts

@Component({
  selector: 'app-root
  template: `
    <!-- header -->
    <nav class="navba

      <!-- logo -->
      <div class="nav
```

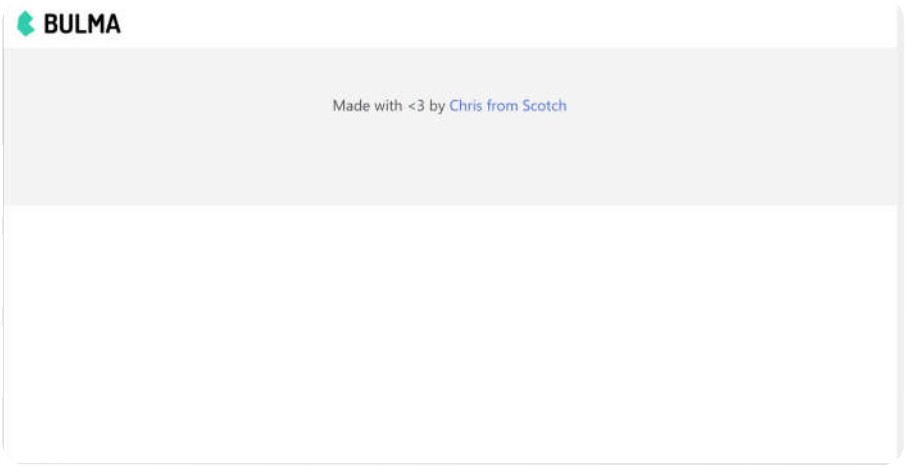
```
      <a class="navbar-item">
        <img src="https://bulma.io/ima
      </a>
    </div>
  </nav>

  <!-- routes will be rendered here --
  <router-outlet></router-outlet>

  <!-- footer -->
  <footer class="footer">
  <div class="container">
  <div class="conte
    <p>
      Made with <3
    </p>
  </div>
  </div>
  </footer>
`,
styles: []
```

18 Conclusion 1:34

```
})
export class AppComponent {}
```

While this will work, our site code is already starting to look confusing. Lots of stuff is going on in this main `AppComponent` and it's not too clear what's happening.

Here's what our app looks like now:



Creating Head Footer the Rig (Components)

The JavaScript world and the web have been moving towards components in recent years. Having components that are brought together to build our sites/apps makes them easy to manage and easy to read the code.

Let's build out a **header component** and a **footer component** and see how much cleaner our code will get.

Use the CLI to Create Header and Footer

We'll use CLI to create **two** inside of a `src/app/components`

BASH

```
ng generate component components/header

# with the shorthand for generate
ng g component components/footer
```

Now we have two clean components that we can use!


```
// src/app/components
import { Component, O
```

```
@Component({
  selector: 'app-header',
  template: `
    <p>
      header works!
    </p>
  `,
  styles: []
})
export class HeaderComponent implements
  constructor() {}
  ngOnInit() {}
}
```

TS

```ts
// src/app/components/footer.component.t
import { Component, OnInit } from '@angu

@Component({
  selector: 'app-footer',
  template: `
    <p>
      footer works!
    </p>
  `,
  styles: []
})
export class FooterCo
  constructor() {}
  ngOnInit() {}
}
```

We have a `selector`, `template`, and `styles` inside of the `@Component()` decorator that tells Angular how to create these components.

We are also given a `class` that isn't really used for these two components since they are presentational and only display the template. We don't have any data in these two components yet.

The header may get some au
later date. We'll be handling
authentication in another co
of this course is to build a sir
and get familiar with how yo
build.

Components Added to the AppModule

The main `AppModule` is our applications main place we register all the parts of this site. The Angular CLI already registered these two new components there so we don't have to!

TS

```
// src/app/app.module

...

@NgModule({
  declarations: [
    AppComponent,
    HeaderComponent,
```

```
    FooterComponent
  ],

  ...
```

`declarations` is where we put any components that we have created. We'll see in a later lesson more about how we can use these `@NgModule()` s to organize our application.

Using Our Nev and Footer Co

The first thing we can do is t components. If you look in e

you'll see the `selector` that we can use to insert these components into our app. We currently have `app-header` and `app-footer`. Let's add these to the `AppComponent` template now:

TS

```
// src/app/app.component.ts
...
@Component({
  selector: 'app-root'
  template: `
    <!-- header -->
    <app-header></app

    <!-- routes will
    <router-outlet></

    <!-- footer -->
```

```
    <app-footer></app-footer>
  `,
  styles: []
})
...
```

Notice how much cleaner our overall template has gotten. Really easy to read! This is the part of the benefits of using components to compose our application. Next is to ac
templates to both our heade
components:

TS

```
// src/app/components/header.component.t
...
@Component({
  selector: 'app-header',
  template: `
    <nav class="navbar">

      <!-- logo -->
      <div class="navbar-brand">
        <a class="nav
          <img src="h
        </a>
      </div>
    </nav>
  `,
  styles: []
})
...
```

TS

```
// src/app/components/footer.component.t
...
@Component({
  selector: 'app-footer',
  template: `
    <footer class="fo
    <div class="conta
    <div class="conte
      <p>
        Made with <3
      </p>
    </div>
    </div>
    </footer>
```

```
    `,
    styles: []
})
...
```

Our app looks the same as it did earlier, but now our organization is much better.

Like this article? Follow @chrisoncode on Twitter (https://twitter.com/chrisoncode)

NEXT LESSON:

ADDING AN IMAGE/LOC