Angular and Web Development

Part 1

SWEN-261
Introduction to Software Engineering

Department of Software Engineering
Rochester Institute of Technology

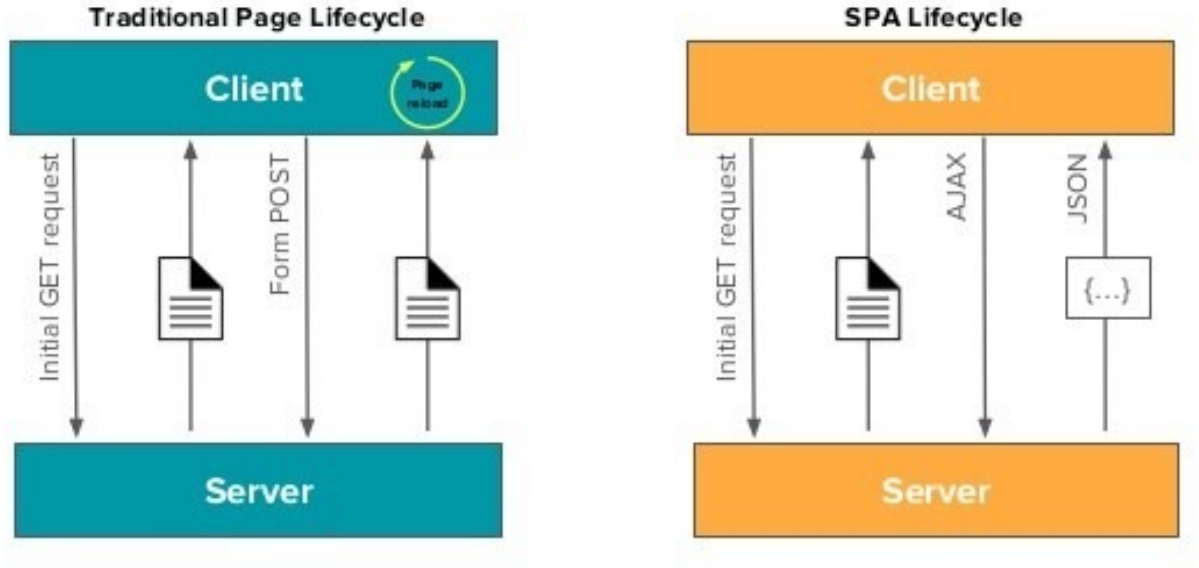
Software Engineering
Rochester Institute of Technology

What is Angular?

- Angular is a platform and framework for building single-page applications using HTML and TypeScript
- TypeScript is a primary language for Angular application development
  - It is a super set of JavaScript and is strongly typed, object oriented and compiled language
- Angular uses HTML as a template language and its syntax can be extended to build application's components quickly
- As a platform, Angular includes:
  - A component-based framework for building scalable web applications
  - A collection of well-integrated libraries that cover a wide variety of features, including routing, forms management, client-server communication, and more
  - A suite of developer tools to help you develop, build, test, and update your code

Single-Page Applications

- In a Single Page Application (SPA), all of your application's functions exist in a single HTML page (index.html)
- As users access your application's features, the browser needs to render only the parts that matter to the user, instead of loading a new page
- This pattern can significantly improve your application's user experience



Hypertext Markup Language (HTML)

- HTML is the language for describing the structure of Web pages. It gives authors the means to:
  - Publish online documents with headings, text, tables, lists, photos, etc.
  - Retrieve online information via hypertext links, at the click of a button.
  - Design forms for conducting transactions with remote services, for use in searching for information, making reservations, ordering products, etc.
  - Include spread-sheets, video clips, sound clips, and other applications directly in their documents

Hypertext Markup Language (HTML)

- Every HTML page uses these three tags:
  - **<html>** tag is the root element that defines the whole HTML document.
  - **<head>** tag holds meta information such as the page's title and charset.
  - **<body>** tag encloses all the content that appears on the page.

```
<html>
  <head>
    <!-- META INFORMATION -->
  </head>
  <body>
    <!-- PAGE CONTENT -->
  </body>
</html>
```

Hypertext Markup Language (HTML)

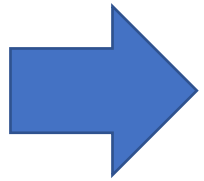
- Example

```
<html>
<body>

<h2>HTML Buttons</h2>

<p>HTML buttons are defined with the button tag:</p>

<button>Click me</button>

</body>
</html>
```

Heading Tag

Paragraph Tag

Button Tag



HTML Buttons

HTML buttons are defined with the button tag:

Click me

- HTML includes many element types including:
  - Content: `h1-h5`, `p`, `img`, `a`, `div/span`, many more…
  - Lists: `ul`, `ol`, `li`
  - Forms: `form`, `input`, `button`, `select/option` …
  - Tables: `table`, `thead`, `tbody`, `tr`, `th/td` …

Angular CLI

- The Angular CLI is a command-line interface tool that you use to initialize, develop, scaffold, and maintain Angular applications directly from a command shell
- It handles the build flow and then starts a server listening on localhost so you can see the results, with a live reload feature
- Install the CLI using the npm package manager:

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

- For more info on Angular CLI, see: https://angular.io/cli

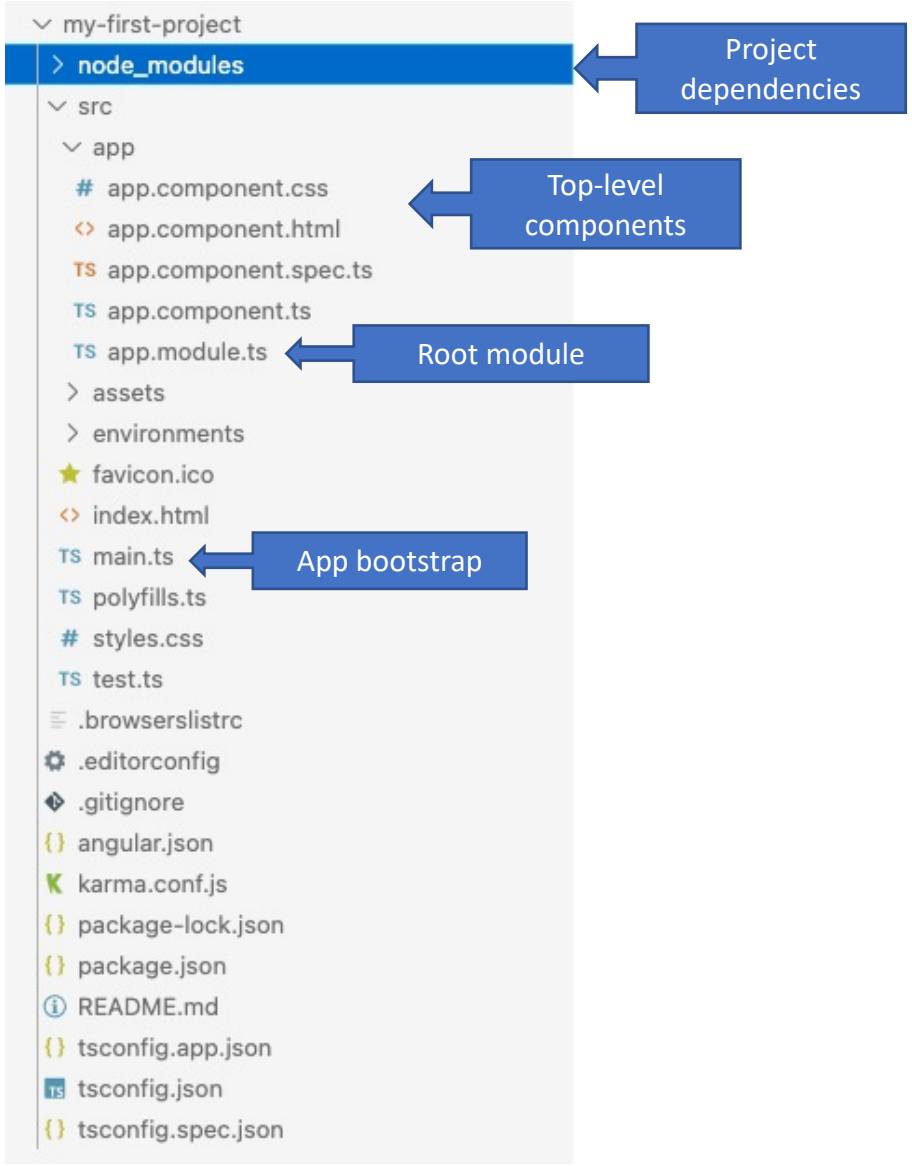
Angular – Code Structure

```
ng new my-first-project
```

This command will create a new directory with the boilerplate project and install all required dependencies – thousands of files in the node_modules directory of your project

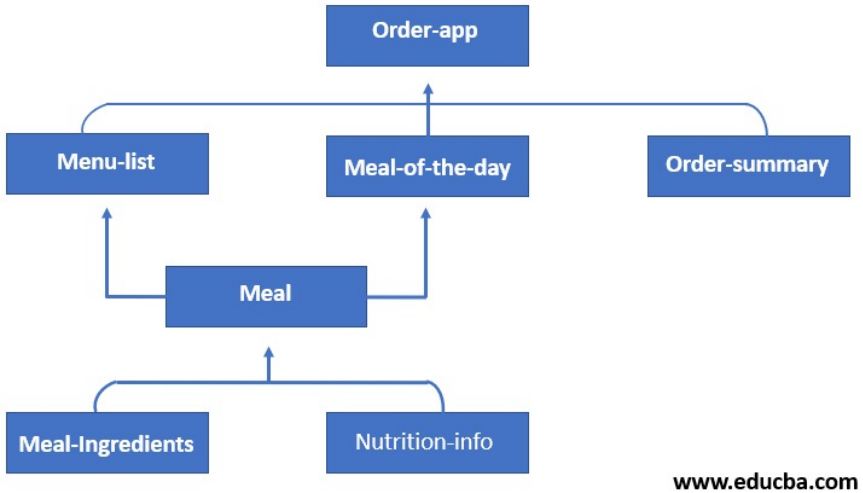
Key files:

- `app.module.ts` - root module
- `app.component.ts` – root component



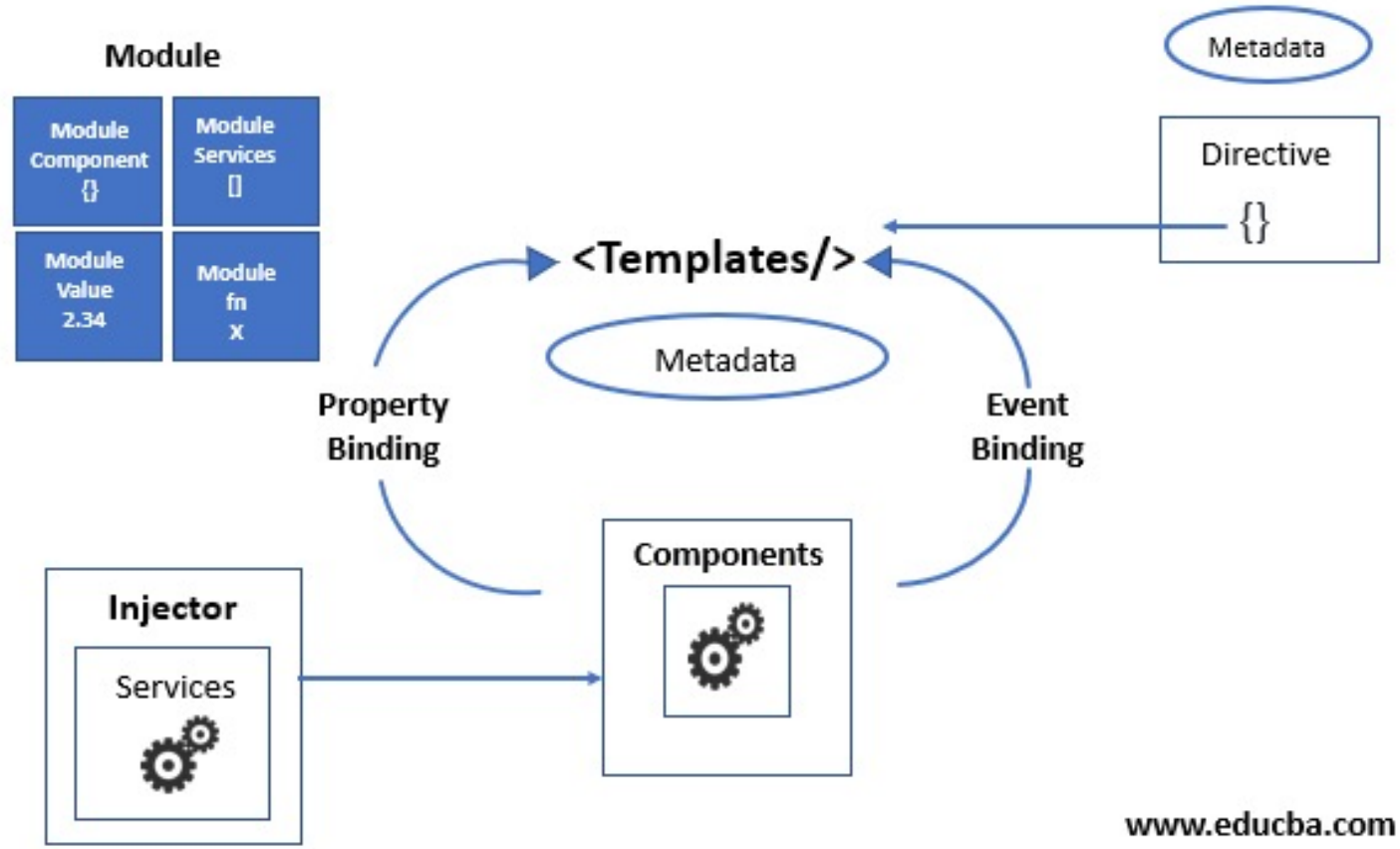
Angular – How does it work

- Angular follows **component-based** architecture, where a large application is broken (decoupled) into functional and logical components
- These components are reusable hence can be used in any other part of the application
- These components are independent hence can be tested independently; this architecture makes Angular code highly testable



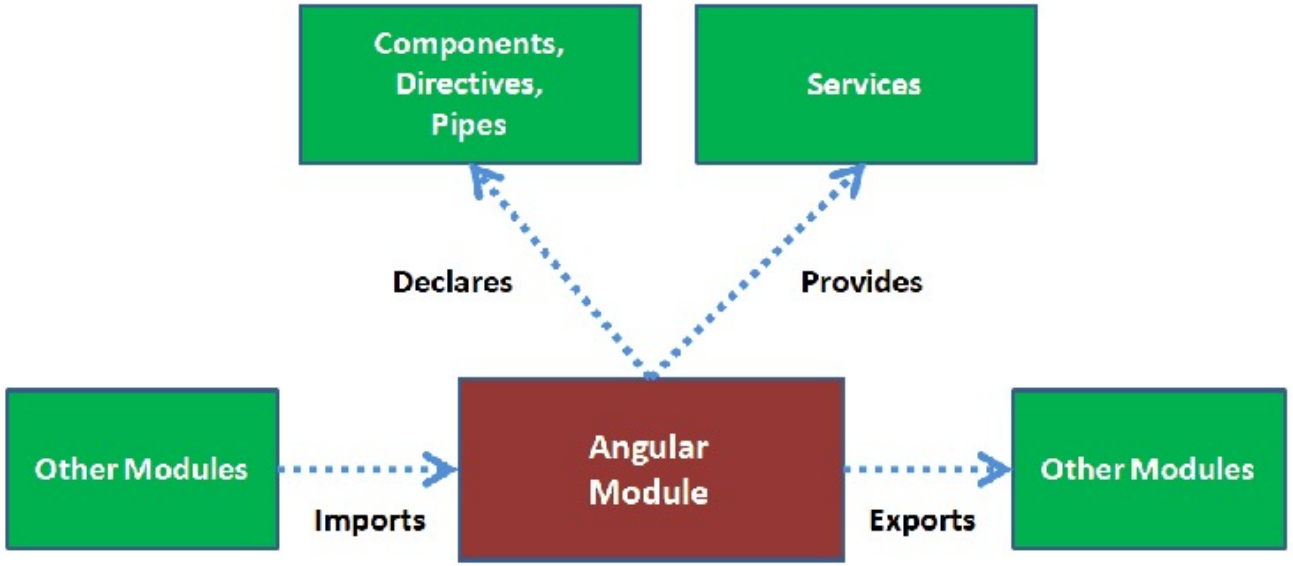
Angular Building Blocks

- The main building blocks of Angular are:
  - Modules
  - Components
  - Templates
  - Metadata
  - Services
  - Data binding
  - Directives
  - Dependency Injection



Angular Building Blocks - Module

- The Angular **module** help us to organize the application parts into cohesive blocks of functionality
- The Angular module must implement a specific feature
  - The Components, Directives, Services which implement such a feature, will become part of that Module



Angular Building Blocks - Module

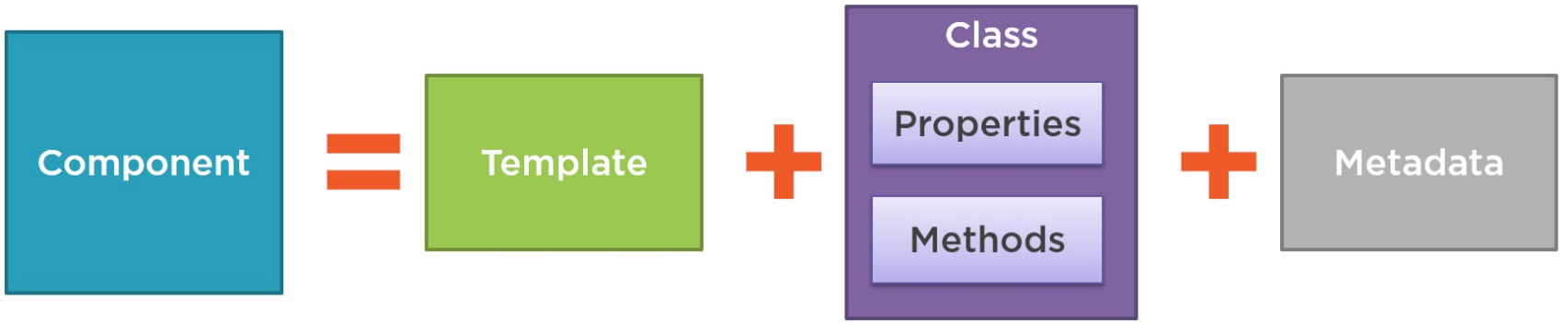
- Many modules combine together to build an angular application
- An application always has at least a **root module** that enables bootstrapping, and typically has many more feature modules
- From a syntax perspective, an Angular module is a class annotated with the @NgModule() decorator
  - A **decorator** function is prefixed with a @ followed by a class, method or property
  - It provides configuration metadata that determines how the component, class or a function should be processed, instantiated and used at runtime

```
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 { }
```

Use **export** so module is visible other modules

Angular Building Blocks - Components

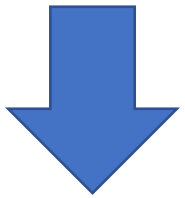
- The **component** is the basic building block of Angular
- Each component defines a class that contains application data and logic, and is associated with an HTML template that defines a view to be displayed in a target environment



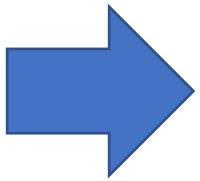
Angular Example – Create New Component

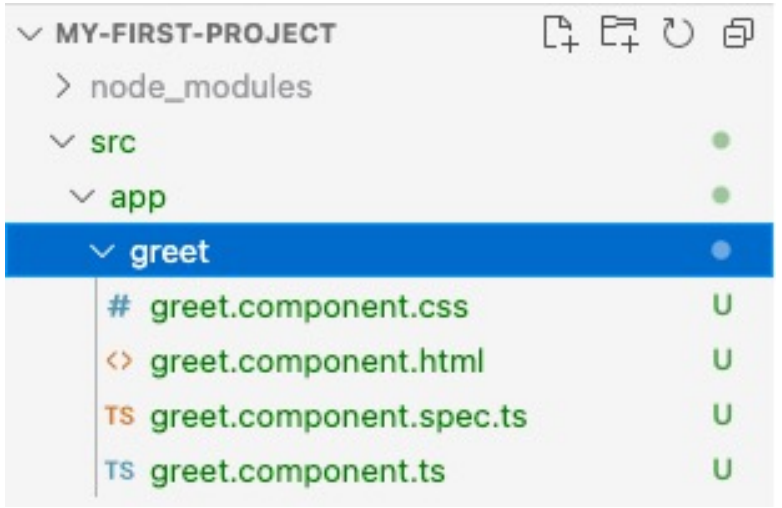
- Use the Angular CLI to generate a component and the associated files

```
ng g component greet
```



```
PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

mikez@Mikes-MacBook-Pro-2 my-first-project % ng g component greet

CREATE src/app/greet/greet.component.css (0 bytes)
CREATE src/app/greet/greet.component.html (20 bytes)
CREATE src/app/greet/greet.component.spec.ts (619 bytes)
CREATE src/app/greet/greet.component.ts (271 bytes)
UPDATE src/app/app.module.ts (392 bytes)
```





Angular Example – Component Details

- The GreetComponent is generated with the template, metadata and the component class

greet.component.ts

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

@Component({
  selector: 'app-greet',
  templateUrl: './greet.component.html',
  styleUrls: ['./greet.component.css']
})
export class GreetComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

}
```

Component decorator

HTML Template File name and Location

Cascading Style Sheet (CSS) File Name and Location

Template & Metadata

Component Class

A **selector** instructs Angular to instantiate this component wherever it finds the corresponding tag (<app-greet>) in template HTML.

We will reference this later the HTML of the root component (app.component.html)

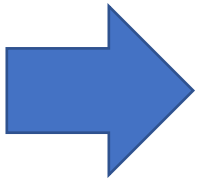
Cascading Style Sheets (CSS)

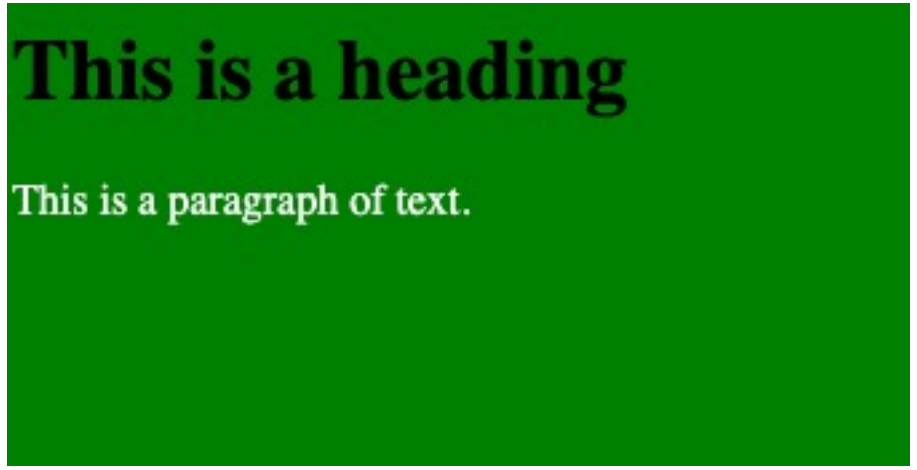
- CSS is the language we use to style a Web page
- CSS is a design language that makes a website look more appealing than just plain or uninspiring pieces of text
- Whereas HTML largely determines textual content, CSS determines visual structure, layout, and aesthetics
- Think “look and feel” when you think CSS

Cascading Style Sheets (CSS)

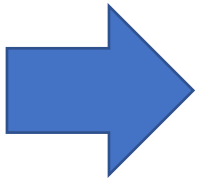
- Example with inline CSS in HTML

```
<html>
<head>
    <title>Example of CSS Embedded Style Sheet</title>
    <style>
        body { background-color: Green; }

        p { color: #fff; }
    </style>
</head>

  <body>
    <h1>This is a heading</h1>
    <p>This is a paragraph of text.</p>
</body>

</html>
```





- The CSS rules can be stored separate from the HTML in a .CSS file



```
body
 { background-color: YellowGreen; }

p
 { color: #fff; }
```

Angular Example – Add code to component

greet.component.ts

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

@Component({
  selector: 'app-greet',
  templateUrl: './greet.component.html',
  styleUrls: ['./greet.component.css']
})
export class GreetComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
  }

  name: string = "Steve";

  greet(): void {
      alert("Hello " + this.name);
  };

}
```

greet.component.html

```
<div>
    Enter Your Name: <input type="text" value={{name}} /> <br/>
    <button (click)="greet()">Greet Me!</button>
</div>
```

Interpolation, denoted with {{}}, allows us to include expressions as part of any string literal, which we use in our HTML

We have added the “name” property and the “greet” method in the component class

Angular Example – Load component

- We want to load our new GreetComponent so the **root** module (app.module.ts) must know about it so we import it and add the GreetComponent in the declarations array in app.module.ts

app.module.ts

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

import { AppComponent } from './app.component';
import { GreetComponent } from './greet/greet.component';

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

Import GreetComponent

Include GreetComponent in declarations

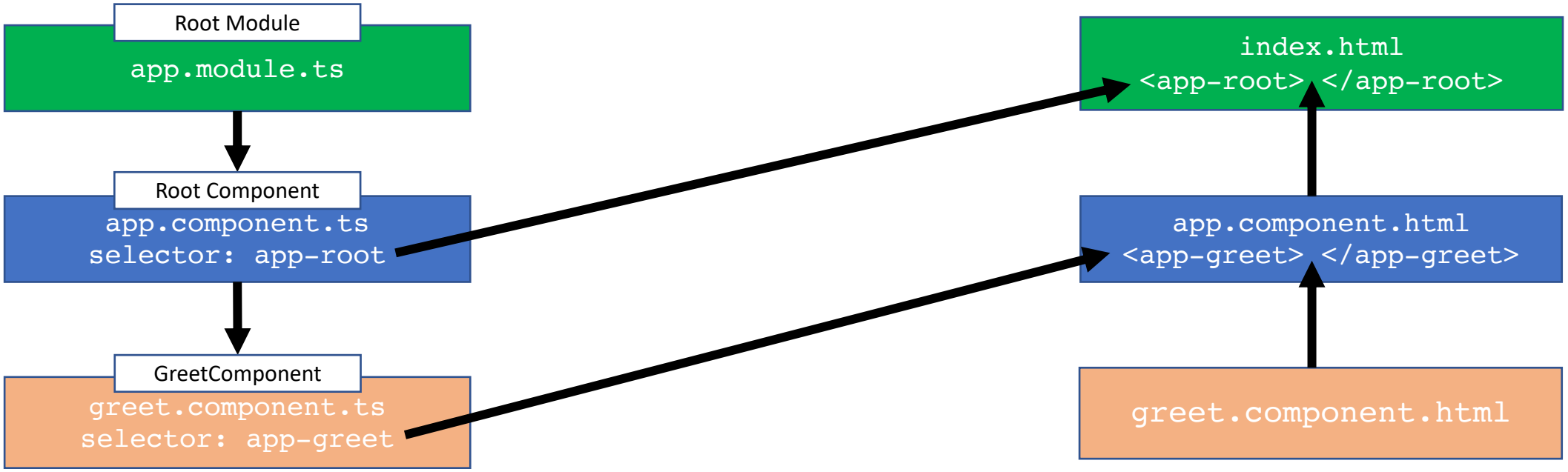
app.component.html

```
<div>
  <app-greet></app-greet>
</div>
```

- After adding a component declaration, use component tag <app-greet></app-greet> in the HTML file of the root component (app.component.html)
- This will allow us to pull in the GreetComponent during the startup (bootstrapping)

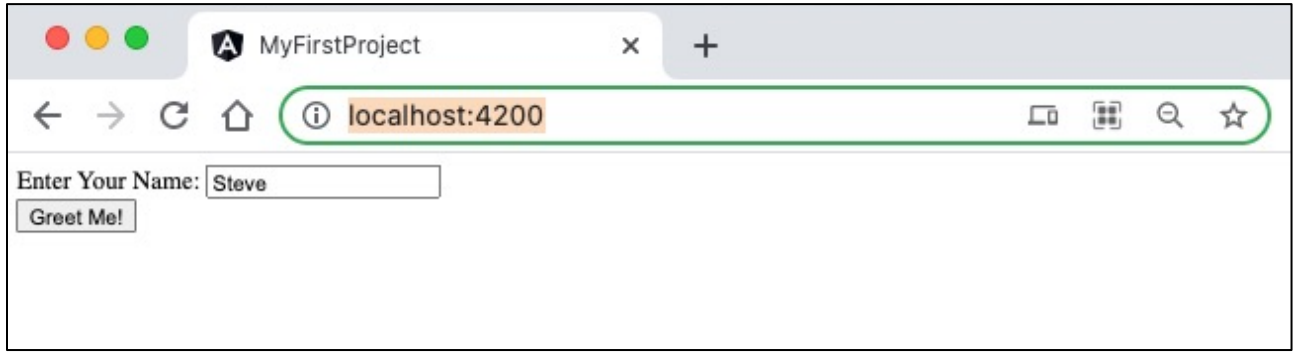
Angular Example – Bootstrapping

- When the Angular CLI builds an Angular app, it first parses **index.html** and starts identifying HTML tag elements inside the body tag
- An Angular application is always rendered inside the body tag and comprises a tree of components
- When the Angular CLI finds a tag that is not a known HTML element, such as <app-root>, it starts searching through the components of the application tree



Angular Example – Let's run it

```
ng serve
```



- With "ng serve" running, you can update your code and it will automatically compile and update when you save your files

Angular Activity – Tour of Heroes – Part 1

- Do activity “Tour of Heroes – Part 1”
- This Tour of Heroes tutorial shows you how to set up your local development environment and develop an application using the Angular CLI tool, and introduces the fundamentals of Angular
- Do steps 1-3 only
- We will cover the remaining steps in the next class