Day2

Tuesday, August 30, 2022 8:58 AM

Single Page Applications

Instead of navigating across multiple html pages
the user stays on one page and the content is switched out

Webapp that readers a single page

All the HTML, JS, and CSS will be loaded at once during startup

During navigation the browser never refreshes because you stay on the same page

We will use Angular to help us develop SPA's

Advantages
- Fast and Responsive
- Caching Capability
- Overall pleasant user experience across multiple platforms

Disadvantages
- Doesn't perform as well with SEO
- Less secure vs Cross Site Scripting
- The initial page load may be slower

Angular and its history

Angular is a typescript based, open source framework
used to develop dynamic web applications

Angular has gone through many changes since creation by Google in 2010

It started out as AngularJS

In 2016 a completely rewritten version was released
- Angular 2

Now refferred to as Angular 2+

Angular 4+ is now the standard
- aprox every 6 months a new version released

AngularJS vs Angular 4+ Comparison

JS uses MVC Architecture vs 4+ uses Component based UI

JS used Javascript to create applications vs 4+ utilizes typescript

JS binding attributes was more difficult vs 4+ uses [] or () notation

JS did not support mobile vs 4+ is mobile friendly

The main building blocks of Angular
- modules
- components
- templates
- metadata

- components
- templates
- metadata
- databinding
- directives
- services
- dependency Injection

Angular CLI

Installed with npm install -g @angular/cli

The CLI is used to create projects, run projects, and add components to projects

To create a project ng new app-name

To start the app in development mode ng serve --open

To build a production version ng build

Webpack

Is a powerful static module bundler for Javascript applications that packages all modules in the application into a bundle serves it to the browser

Its going to take our angular app take all the html and merge it into a single file

It'll also do that with the JS and the CSS

This helps speed up the loading of our webapp

As webpack bundles our application it starts a list of modules and builds a dependency graph that includes every module our app needs to more effiency create those bundles

Angular Modules

In angular a module is a group of components that are related in some way

modules can be combined with modules to make up an entire application

There are two types of modules:
- root module
- feature modules

We have one single root module
- it does all the bootstrapping
- configuration and keeps of components and other modules

We have 0 or more feature modules
- custom groups of components

To create a feature module ng generate module name --module app.module

@NgModule Decorator

Used to mark a class as a module

In the AppModule we use @NgModule to setup the root of the application

NgModule will take in metadata and describe how to compile components templates and how to inject services at runtime

The Angular CLI creates AppModule for us, and includes the following metadata:
- declarations: a list of components, directives, guards
- imports: list of other modules being used in the app
- providers: service providers for the app
- bootstrap: contains the root component

The steps of running our application (bootstrapping)

1. the main.ts as the entry point of the app
2. Pass the app.module to the main.ts as an argument
3. Analyze the app.component and then knows there is an app-root selector defined
4. Handle/inject the app-root component into the index.html
5. index.html is displayed on the page

Components and @Component Decorator

Components are the basic building blocks of webapp

We want to breakup of UI and logic into individual peices with specific purpose

The angular app has one main root component AppComponent

To mark a class as a component we use @Component

This decorator comes from @angular/core, and is used in the ts file of your component files

The decorator has these properties/metadata
- selector: tag used when using the component in html
- templateUrl: links an html page to the component
- styleUrls: links css to the component

To generate a component we use ng generate component name
- sets up all files and asrociations

Angular Component Life Cycle

The lifecycle of a component:
- Creation
- Render
- Create/Rerender Children
- Check for data changes
- Destroy and remove from Dom

The events are called LifeCycle hooks and we have 8 different functions to hook into the life cycle

constructor: the constructor of the class, called first
- inject dependencies

constructor: the constructor of the class, called first
- inject dependencies

ngOnChanges: called whenever the input properties of the component change

ngOnInit: Called once after the component initialized, set input properties

ngDoCheck: called during all change detection runs, immediately after ngOnChanges

ngAfterContentInit: invoked once after angular performs any content projection into any views

ngAfterContentChecked: invoked every time angular checks for content projection

ngAfterViewInit: after angular initializes views and its child views

ngAfterViewChecked: every time angular checks for content projections

ngOnDestroy: called once before removing component

Data Binding

How we provide communication between the dom and components

We have two ways of data binding

1-way data binding

Allows us to manipulate views through models

String interpolation: take data from the component.ts and inject it into html with {{ data }}

Property: bind values to the attributes of an html element uses []
. Use this to send data from a parent component to child
- in the child component.ts we have to use the decorator @Input()

Event binding: register event listeners with ()