Angular

* Front-end framework.
* It utilizes TS not JS.
* It was designed by Google.
* It is a SPA (Single Page Application)
  + There is only one page on the front end and HTML is dynamically updated and removed.
  + It is usually more efficient and easier to build in frameworks.
* Angular IS NOT AngularJS
  + AngularJS is the original
    - Written in JS not TS.
    - Different scopes.
    - Very different from Angular 2.0+.
  + Angular 2.0 and up are called Angular.
* Angular is based on components.
  + Small chunks of HTML/CSS/TS that we use to build a larger application.
  + You can duplicate components.
  + Create custom components.
  + Makes the application modular and easier for a team to work on.

Angular Architecture

* Module
  + Every application has a root module.
    - Main registry for components and other modules that comprise your angular application.
      * FormsModule
      * HttpClientModule
      * MaterialModule
      * RoutingModule
    - app.module.ts
* Components
  + Visual/interactable aspect of an angular application.
  + They are what a person sees and interacts with.
  + Logical chunk of HTML/CSS/TS
  + They are reusable.
  + They can be nested
  + Every application has app-root as the main component.
    - Bootstrap component
      * The first component pulled into the index.html
  + Three parts of any component
    - Decorator (meta-information)
      * @Component()
        + Contain meta information like the selector and links to other files.
    - HTML template (Visual)
      * Defined directly in the component or reference a file elsewhere.
    - TS class (logic)
      * Variables and methods that work on that template
  + Whenever a component is rendered for the first time the onInit() function is called.
* Services
  + Singletons
    - When you create a service Angular will only ever make instance of that service.
  + Services via Dependency Injection are put into your components.
    - Constructor(private somethingService:SomethingService)
  + Are used to shared data between components.
  + To reuse commonly used functions in your application without having to retype those function.
    - DRY Do not repeat yourself.
* Pipes
  + Transform values.
  + Must have an input and must give out some sort of output
  + Very helpful for formatting information in the HTML
    - {{x | currency :’USD’}}

Biding inAngular

* One way binding
  + Interpolation
    - TS => html
      * name:string = “adam” => {{name}}
  + Event binding
    - HTML => TS
      * <button (click)=”hello()”> => hello(){}
  + Property Binding
* Two-way binding
  + [(ngModel)]
    - HTML ⬄ TS
    - Simultaneously talk to each other

Routing

* App-routing.module.ts
  + A variable called routes where you can add Route objects
    - {path:”home”,component:HomeComponent}
  + Router-outlet tag in the app-component where the selected component will get placed into.

Directives

* Custom HTML specific to Angular
* Make our HTML dynamic.
* Three types of directives
  + Component directive
    - Custom HTML tag
  + Structural Directive
    - \*ngFor
    - \*ngIf
    - Manipulate the DOM by adding or removing elements.
  + Attribute Directives
    - Custom attributes
    - [(ngModel)]

Angular CLI

* CLI (Command Line Interface) commands to know.
  + Ng new exampleapp (makes a new project)
  + Ng generate
    - Component (create a new component)
      * Html template
      * TS class file
      * CSS file
      * Spec.ts (for testing)
    - Service (create a new service)
    - Pipe (create a new pipe)
  + Ng serve
    - Run you application on a local development server
      * Hot loads your application so that any changes to the code are immediately detected and the app rebuilt
  + Ng build
    - Transpiles and links your code together into a deliverable set of files
    - Dist folder stores these files
  + Ng add
    - Add a new dependency to an angular application

HTTP requests in Angular

* Import HttpClientModule in your app.module.ts
* Dependency inject into the constructor the HttpClient class
  + Usually only services would inject this.
* http.get<type>(“url”)
  + default returns an observable
  + .toPromise() to turn them into promises.
  + Observables are more powerful than promises.
  + With a promise you are waiting on 1 thing to come back
  + You subscribe to an observable and perform a function every time you get an update
    - Very similar to the concept of github webhooks
    - Watch for a change and perform an action a change is detected
  + 