TypeScript

* TypeScript is a super set of JS
  + Any valid JS is also valid TS
* TypeScript has features that JS does not have
  + Static Typing (TYPES) Literally called TYPEScript
  + Interfaces
  + Access Modifiers
  + Has classes before JS (More OOP features than JS)
    - TypeScript enables more OOP design than pure JS
* Transpiled language
  + Must be turned into JS to be executed
  + By default it transpiles into es3 (2009)

Angular

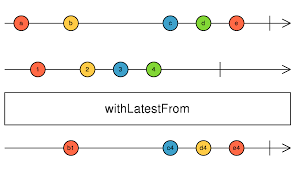
* Angular is a front-end framework to make Single Page Applications
  + SPA – There is only html page that is ever shown to the user
    - HTML is dynamically swapped in and out of this page to give the illusion of multiple pages
    - Faster loading than multiple pages
    - Better user experience
  + Bootstrapping
    - The process of one component loading in another component
    - AppComponent is the initial BootStrap Component
* Directives (Custom HTML)
  + Structural Directives (manipulate the dom by adding or hiding elements)
    - \*ngIf \*ngFor
  + Component Directives
    - Custom HTML tags
  + Attribute Directives (an html attribute that does not exist in regular HTML)
    - [(ngModel)]
* Angular Architecture
  + Module
    - Every Angular application has a root module
      * This root module is where all of your components and services must register with Angular
        + Providers: services
        + Imports: other modules

Think outside libraries like httpClient

* + - * + Declarations: components
  + Components
    - Logical constructs of HTML
    - Every component exists of three parts
      * HTML
        + View (what the user sees)
      * TS Class
        + Logic (TS code)
      * Decorator @Component()
        + Meta information about your component

Where to find the view

What is the custom html tag for this component

* + Services
    - Utility (service) classes
    - Singletons
      * Every service gets created only once
      * A single instance of the service is used throughout the application
    - Do not contain any view
    - Services are @Injectable
      * They are injected into components via the component’s constructor
    - They are convenient to share information between components
  + Pipes
    - A way to transform our information before rendering it to the html page
    - Very convenient for when we want to do some formatting to our web page
* Data Binding
  + How we can pass information between the html and TS
  + One-way data binding
    - Interpolation TS=> HTML
      * <h1>{{variable}}</h1>
    - Event Binding HTML=>TS
      * <button (click)=”function()”>click me</button>
    - Attribute Binding TS =>HTML
      * < img [src] = “variable”></img>
  + Two-way
    - [(ngModel)]
    - HTML 🡸🡺 TS communicate simultaneously
    - Very useful for form in Angular
* HTTP calls in Angular
  + Import HttpClientModule and register it in the main module
  + Dependency inject the HttpClient class into any service that needs to make http calls
    - this.http.get<Book>(“url”).toPromise();
  + HttpClient returns an Observable
    - A variable whose value could change and require us to update it
  + 
    - Imagine if your endpoints would periodically give you back responses
* Routing
  + Angular is a SPA where going to a new page is just an illusion to the user
  + There is a tag <router-outlet></router-outlet> that will have a component loaded into it when some searches for a path in the url
  + There is a routing module which has an array routes
    - Here you can define routes

Continuous Integration

* You are never coding just for yourself in modern applications
* Modern applications can be very large and are worked on by teams of developers
* Small changes that you commit and push every few hours makes debugging and integrating code much easier
  + Imagine if your team only talked to each other one a week to try and merge your code together
* All developers share a remote repository
* When a developer has to create a new feature he branches off the main branch
* The developer writes that feature
  + Tests it locally (if applicable)
  + Pushes it to the remote branch
  + Jenkins can read these branches and run tests to see if you code works and integrates with the rest of the code base
* Pull requests
  + Another team member will analyze your code to make sure it is of sufficient quality before merging it into the main branch

Jenkins

* Jenkins is a CI tool
* Read your github repository
  + Build your application
  + Test your application
  + Deploy your application (we did not do this)
* Jenkins terms
  + Job: Project
  + Build: every time Jenkins tries to build an application
  + Webhook: URL endpoint to the Jenkins that Github will send information to whenever Github detects a change

Static Code Analysis

* Sonar Cloud or any other code linter reads your source code
  + Does not compile anything or run any tests
  + It just looks for bad coding practices and makes you aware of them

Continuous Integration vs Continuous Delivery vs Continuous Deployment

* Continuous Integration
  + Build and run tests
* Continuous Delivery
  + Creates a Deliverable
  + A final .jar file
* Continuous Deployment
  + The deliverable is run on server so that end users can begin using the application immediately

Selenium

* Automates Web browsers
* Core Interface of Selenium is the WebDriver
  + The actual implementation driver like ChromeDriver() depends on the browser being automated
* Ways to get elements in Selenium
  + Id
  + Name
  + Xpath
  + CSS
  + LinkText
  + className
* Types of waits in Selenium
  + ImplicitWait
    - Set once
    - Default amount of time Selenium will wait for an element to appear before throwing an exception
  + ExplicitWait
    - Set for a specific element
    - How long you are willing to wait for this Specific element to appear before throwing an exception
* POM
  + Page Object Model
  + Class representation of a web page
  + The instance variables of the class correspond to main elements in the webpage
  + @FindBy(id = “”) to populate the fields of a pom
  + POM allow us to write our code more succinctly
    - Homepage.searchbar.sendKeys(“sadfdsf”);
    - Driver.findElementBy(By.name(“searhbar”).sendKeys(“sesdf’);

Cucumber

* BDD framework
* Feature files in Gherkin Syntax
  + The scenarios in the feature file have acceptance criteria steps
  + These steps can be linked to Step Implementations in another package