Angular – Week 6 review

TypeScript - Programming language which is a superset of JavaScript and can be compiled down (transcompiled since no lowering) to different versions of JS, depending on tsconfig.json, and adds opt-in, strict, compile time checking.

Class: Like oop classes. Can have constructor (which is named constructor !), properties (fields) and methods, as well as inheritance with extends. Has public, private and protected modifiers, with public as default, and readonly. Access modifiers and/or readonly on a parameter both assigns and declares it as a field in the class. Can declare getters and setters with

get *fieldname*(): *returnType* { *logic* } and set *fieldname*(*varName*: *fieldType*) { *logic* }. Can have static members, and abstract classes and methods, and interfaces.

Interface: Only the shape matters. Don't need to state that you implement the interface, just meet the number and types of properties in the interface (note that order doesn't matter). Can specify optional properties with *name* ?: *type*; Class can inherit from interface with implements.

Type Annotations: After declaring a variable, give the type with : *type*. Not mandatory, but enables compiler to perform type checking. Improves readability and maintainability.

TS Modules: Namespaces to take stuff out of global scope. Export stuff you want visible outside. Others must then import it to see it.

Config: tsconfig.json file indicates the directory is a TS project's root. Specifies compiler options and root files. Compile with tsc command. compilerOptions is optional (will use defaults). Can include and exclude ts (and js) files.

Webpack: TS Module bundler. Combines module files together (might not be into one.) Can perform minification (removes whitespace and shortens names), which minimizes filesize to make transport over n/w better.

Node.js – A JS runtime environment. (Node api or modules are taken as library classes, and the js code is taken with node by the JS runtime engine to be made into machine code that you can run.) Contains the npm.

runtime: npx is npm package runner?

NPM: Node package manager. npm is kinda like sophisticated nuget for node packages, is run on command line to install dependencies.

package.json: For setting 3rd party dependencies. Can specify the version as well. (Also, how you publish your project with a unique name and version number and other info.)

semver: Semantic versioning. ~, tilde, uses patch updates (matches the minor version) and ^, caret, uses minor updates (just matches major version).

node.modules: Included with require('*path*/*name*') function. Can create your own with exports.*name* in your file to make the named property/method available outside the file.

Angular – Framework for single page applications using TS. (Affects the DOM instead of moving to different webpage.)

CLI: Angular cli is installed by npm with npm install -g @angular/cli (-g is for global.) Includes ng new, to create new app, ng generate to create components, routes, services and pipes, and ng serve to run your app locally. Can also test and lint.

configuration: Angular.json. In root of angular workspace. Set workspace and project configuration.

Decorator: Decorators provide metadata that angular needs. Part of angular.

module + decorator: A ng module is a (TS) class with an (Angular, not TS!) decorator, @NgModule.

In @NgModule decorator, Declarations array is for components, directives and pipes, Providers array is for Services, Imports array is for other modules.

component + decorator: Component is a class with a decorator that turns it into a component, @Component. References an html template file to define a view (a physical area on the page with some logic and its own lifetime) and a CSS style file.

ngOnInit: Called by angular shortly after creating a component. (Is a lifecycle hook, which angular uses to show the create + render + destroy of component and checking property change). Good to avoid complex constructor logic.

TS needs import into file to see the dependency and ng has imports array to add needed dependencies to Angular.

Selector defines the html tag element that will be replaced by this component.

Every component must be declared in exactly one NgModule! (module declarations array is for declaring components.)

Every application has at least one Angular module, the root module that you bootstrap to launch the application. Root module has default name appmodule, but can be changed. Set in main.ts .

directives- A directive is anything that can go into template that is not valid html. Apply to host element.

Structural: Starts with asterisk, \*. Responsible for HTML layout (add, remove, change DOM elements). "Change structure" \*ngIf and \*ngFor

Attribute: Change the appearance or behavior of an element, component or other directive. "Used as attribute of elements" NgStyle (logic for setting multiple styles) and NgClass (logic for setting class. Has string name, key:value for multiple classes named the key for each truthy value, or array of the above).

Component: Directive with a template. Most common.

Can create your own directives with @Directive decorator on a class. (Selector having [] makes it attribute selector, etc.)

service/injectable + decorator: Make class a service (view-independent logic that supports dependency injection) with @Injectable.

provider (root, module, component):

dependency injection: Add reusable services to modules by TS importing them into module and adding to its providers array.

data binding – Definitely need FormsModule for two way binding.

interpolation: {{ *name* }}

property: [*name*]="*value*"

event: (*DOMeventType*) = *"componentFunc()"*

two-way: Requires FormsModule. [(ngModule)]="*name.property*"

pipes: Take input, transform it, and output it. Use with pipe operator, | . Can pass parameters with colon, : . Can chain pipes one after the other. Ex: *input* | *pipename* : *parameter* | *pipe2*

Can create your own pipes by importing Pipe and PipeTransform from @angular/core, using @Pipe decorator on class, where you must specify the name, and implementing PipeTransform.

Two different APIs in browsers, XMLHttpRequest interface and fetch() API.

HttpClient: Need HttpClientModule. Then import into HttpClient. Using with a service helps add data checking/error handling and other (post-)processing of the data. Uses XHR.

Observable: Part of RxJS (Reactive Extensions for JavaScript). Asynchronous. Uses JSON by default. Allows any number of subscribers. Can convert observable to promise with .toPromise(), which is better if you only expect one response.

Jasmine unit-testing: Angular default uses Jasmine framework instead of xUnit for testing.

Run default tests with ng test. Has default configs you can edit. Can set up CI.

Unit test methods are 'specs' and are about one object. Uses 'describes' function, composed of 'it' functions, which are composed of 'expects' (assertions). One false expectation fails the spec.

Karma: Test runner.

spy is another name for mock object?

coverage: Use --code-coverage option to generate a code coverage report in created /coverage folder. (Also, an option in cli config, angular.json.) Can use thresholds to enforce coverage.

deployment:

routing: Route is path and component. Set in app with RouterModule.forRoot(routes). RouterOutlet directive ?

guards: Can protect route through navigation guards. (authorization) 5 types: CanActivateChild for children of the route, and CanDeactivate for route deactivation (deepest-down to top) ; CanActivate for if route can be activated/gottento (top down), CanLoad for if feature module is loaded asynchronously, and Resolve for route data retrieval before route activation.

Can implement your own guards, multiple of every level.

Any guard returning false cancels navigation and skips pending guards.

template syntax –

Script element is not allowed (ignored with console warning) in template (to prevent script injection). Some other HTML tags doesn't make sense (<html>, <body>, <abse>).

exprs: A template expression is surrounded by the double curly braces, {{ *expr* }}, or in the quotes of the property binding. Cannot put JS expressions that have effects (such as assignment, increment, or new, cannot chain expressions with ; or , , use operators such as typeof or instanceof, bit operators such as & or |. Also adds new operators, | , ?. , and ! .

statements:

reference variables: Reference to a DOM element in template or an Angular component or directive. Declared with #.