**Angular**

* Frontend/Client side JavaScript Framework
* Created & mantained by Google
* Used to build powerfull single page applications (SPAs)

2 ways to install:

* Angular CLI
* Node.js & NPM

Package.json

* app manifest: name, version, licence
* npm scripts
* dependencies & devDependencies

**Why use Angular**

* Rapid development & code generation
* Code organization & productivity
* Dynamic content
* Cross platform
* Unit testing ready

**Components**

* Basic building blocks of the UI. An Angular app is a tree of Angular Components.
* Decorators allow us to mark a class as an Angular component & provide metadata that determines how the component should be processed, instantiated and used at runtime.

**Services**

* Classes that send data and functionality across components.
* Ideal place for AJAX calls.

**Directives**

* **\*ngFor** – loop through collections (*\*ngFor=”let obj of objects, let i = index”*)
* **\*ngIf** – displays http element conditionally (*\*ngIf=”booleanProperty”*)
* **ngClass** – assigns css classes to html element
* **ngModel** – for 2-way data binding

**Routing**

* Needs RouteModule & Routes
* Maps app paths to components
* <router-outlet>

**Data binding**

1. **Interpolation** (only string values) /component -> template/

* Dynamic value in html template {{}}
* *{{propertyName | expression | javascript code}}*

1. **Events** /template -> component/

* *(click) = „onClick($event)”* (click = mouse event)

1. **Property binding** /component -> template/

* Html attribute – initialize DOM property, can not change
* DOM property – current value, can change
* We are binding to DOM property
* *[property] = „propertyName”*
* *bind-property = „propertyName”*
* *property = „{{propertyName}}”* (only for strings)

1. **2 way binding** /component <-> template/

* template reference variables – to easily access DOM elements
* *<input #myInput> -> myInput.value*
* import FormsModule
* *[(ngModel)] = „propertyName”*

**Templating**

**Http Module**

**Forms Module**

**Observables**

**Pipes**

**Events**

**TypeScript**

* Superset of JavaScript with added festures
* Created by Microsoft
* Optional static typing
* Class based object-oriented programming
* Resemples languages like C# and Java