**Angular**

**Creating Angular Application By Angular Cli**

**Install Angular Cli**

To install Angular Cli, user the NodeJs and npm

*npm install -g angular-cli*

To start an application use

*ng new <application name>*

To run application use

*ng serve*

To generate code use

*ng generate <scafford> <name>*

Scafford types: component, directive, service, route, module, pipe, interface, enum.

To build application in deployment environment

*ng build*

*ng build --prod*

To run the Angular tests

*ng test*

**Components**

Angular application just means a tree structure of components, when the parent component renders, it renders all children.

Component communicate with other by data binding and service:

Parent component communicates with its children through @Input binding to go down the tree

Child component communicates with its parent through @Output binding to go up the tree

All children component can communicate with others through custom service or their parent as around way

**Life Circle Hooks**

**Hooks for the component**

**contructor**

Use when create component by call new on the class

**ngOnChanges**

Invoked every time there is change in one of th input properties of the component

**ngOnInit**

Invoked when the component has been initialized. This hook is only called once after the first ngOnChange

**ngDoCheck**

Invoked when change detector of given component is invoked, it allow us to implement our change detection algorithm for the given component

**ngOnDestroy**

Invoked before Angular destroy component. Use this hook to unsubscribe observables and detach event handlers to avoid memory leaks.

**Hooks for the component children**

**ngAfterContentInit**

Invoked after Angular perform any content projection into the component view

**ngAfterContentChecked**

Invoked each time content of the given components has been checked by change detection mechanism of Angular

**ngAfterViewInit**

Invoked when the component’s view has been fully initialized.

**ngAfterViewChecked**

Invoked each time the view of the given component has been checked by the change detection mechanism of Angular

**Stream and Reactive Programming**

Stream is the sequence of values over time.

For example,

The x,y position of mouse as it around the screen in a HTML5 game.

The data return from a real time websockets connection.

The chat window opened by this user in a browser.

Reactive programming is the idea that can define the application as a series of different streams with operation that connect the different streams together and which are automatically called when new values are pushed onto those streams.

Observable, is a new primitive type which acts a blueprint for how to create streams, subscribe to them, react to new values and how combine streams together to build new ones.

Set the time for each push of value onto the stream, use interval:

*let obs = Rx.Observable.interval(1000);*

Make the stream hot to everyone to hear the stream, use subscribe:

*obs.subscribe(value => console.log(“Subscriber: ” + value));*

pass in the subscribe a call back to react each time the value push onto stream

Take the number of values on the new second stream, use take:

*let obs = Rx.Observable.interval(1000).take(3);*

*obs.subscribe(value => console.log(“Subscriber: ” + value))*;

**Http API**

Angular supports Http Module for get communicated to server by GET, POST, PUT, DELETE, so on … method.

Using the Observable (by default) or the Promise after returning the result from server, can react with the stream to make process for request (when request success or failed)

**Single Page Application**

SPA, the application will show the view to browser and client through one page, all further changes in URL will only be made by client and some request for changes will be sent to server, and the result will be show on that view, the entire page has no need to reload.

**SPA advantages:**

Faster because only making request to server when having changes.

Less bandwidth, instead of getting a big html page and render it, just call a smaller API which return just enough data to render the change.

Independent between the client development and the server development.

**Routing**

To make SPA, need something called Client side routing, which make the change in URL but still hold enough state’s information to keep all results on page existing.

Angular supports Router Module with Router, Router Outlet, Activated Route, Router Link Active, Router Navigate, … for making a SPA.