Angular Bootcamp Notes

How to use a service to communicate between components whenever anything changes in one of the components:

<https://stackoverflow.com/questions/64498367/refresh-one-component-on-click-event-of-another-component-in-angular>

Where apps deployed

<https://vercel.com/jay4351-hotmailcom/cards/settings>

Angular pipes

<https://angular.io/api?type=pipe>

Datepipe:

<https://angular.io/api/common/DatePipe>

<https://www.npmjs.com>

<https://semantic-ui.com/elements/segment.html>

Angular Connect Presentation:

<https://www.animationsftw.in/#/>

Browser comparer:

<https://caniuse.com/?search=web%20animations%20api>

easing custom animation curves

<https://cubic-bezier.com/#.17,.67,.83,.67>

CSS animation examples:

<https://animate.style>

Code used in above examples to see how to implement:

<https://github.com/animate-css/animate.css/tree/main/source>

How to run typescript file from command line:

npx ts-node src/foo.ts

typescript online IDE:

<https://www.typescriptlang.org>

Wikipedia get page:

<https://www.mediawiki.org/wiki/API:Search>

If you google OWASP, you can find examples of code you can use to test whether your application is vulnerable to an attack.

Online IDE Testing for RXJS

<https://out.stegrider.vercel.app>

RXJS documentation:

<https://rxjs.dev>

RXJS code for <https://rxjs.dev>

const {Observable} = Rx;

const {tap, share} = RxOperators;

const observable = new Observable((subscriber) => {

//throw something into our pipeline: number, string, object, whatever

subscriber.next(1);

subscriber.next(2);

subscriber.next(3);

//marks observable as complete

subscriber.complete();

//emit an error. No more vales will come out

//subscriber.error(new Error('gekaglj'));

}).pipe(

tap(value => console.log('From tap:', value)),

share()

);

observable.subscribe(

(value) => console.log('Next value:', value),//next

(err) => console.error('Bad Thing !!!', err.message),//error

() => console.log('Complete') //completion

);

observable.subscribe((value) => {

console.log('From second subscribe', value);

});

/\*

observable.subscribe({

next(value){

console.log('Got a value', value);

},

complete(){

console.log('Observable is complete. Dont expect any more values');

},

error(err){

console.log('Bad thing!!!', err.message);

}

});

\*/

//only here because tool requires is

//observable;

new Observable(() => {});

/\*

const { fromEvent } = Rx;

const { map, pluck } = RxOperators;

const input = document.createElement('input');

const container = document.querySelector('.container');

container.appendChild(input);

const observable = fromEvent(input, 'input')

.pipe(

pluck('target', 'value'),

map(value => parseInt(value)),

map(value => {

if(isNaN(value)){

throw new Error('Enter a number!');

}

return value;

})

)

observable.subscribe({

next(value){

console.log(`Your value is ${value}`);

},

error(err){

console.error('BAD THING HAPPENED!!', err.message)

}

});

//This is specific to this tool, we don't need to write this in the real world

observable;

\*/