14 Angular Typescript (31)

https://ide.c9.io/laczor/angular2

Session_31_Typescript

393 Using types

```
let myString: string;


myString = 'This is a string';


// Try to assign a number to a string => Error

//myString = 4;


// TypeScript can also infer types


let anotherString = 'This is a string without :string'; // => Type 'string' was inferred from t

he assigned value


// This will still resolve in a compilation error

// anotherString = 4;


// TypeScript may only infer values when those values are assigned at the declaration

// This does not work:


let yetAnotherString;


yetAnotherString = 'This is a string';


// TypeScript does not know the type, therefore we don't get an error ... but no we're also ign

oring TypeScripts strength: Typing

yetAnotherString = 5;
```

```
// Other basic types

let aString: string;
let aNumber: number;
let aBoolean: boolean;
let anArray: Array<string>; // This is a generic type => May only hold 'strings' in this case
let anything: any; // Any can be used if we don't know the actual type => Use it rarely!
// We also got void (=> nothing) and enums (a set of numeric values)
```

394 Classes

```
// Classes allow us to create 'blueprints' for objects
// In Angular 2 we use classes a lot. For example to create Components, Services, Directives, P
ipes, ...

// How to create a class

class Car {,
    engineName: string;
    gears: number;
    private speed: number;

    constructor(speed: number) {
        this.speed = speed || 0;
    }

    accelerate(): void {
        this.speed++;
    }

    throttle():void {
```

```
        this.speed--;
    }

    getSpeed():void {
        console.log(this.speed);
    }

    static numberOfWheels(): number {
        return 4;
    }
}

// Instantiate (create) an object from a class

let car = new Car(5);
car.accelerate();
car.getSpeed();

console.log(Car.numberOfWheels());
```

395 Interfaces

Basically they are contracts, type definitons, how the stuff should look like, what properties, functions should it have.

```
// Interfaces allow us to create contracts other classes/ objects have to implement
// We can use them to define custom types without creating classes
// Interfaces ARE NOT compiled to JavaScript! It's just for checking/ validation done by our TypeScript compiler

// Example interface

interface User {
```

```
    username: string;
    password: string;
    confirmPassword?: string; // Optional property => Does not have to be implemented
}

let user:User;

// This value does not satisfy the interface => Compilation error
// user = { anything: 'anything', anynumber: 5};

// This value does satisfy the interface
user = {username: 'max', password: 'supersecret'};

// Interfaces can also contain functions (without the function body - as it only is a blueprin
t/ requirement)

interface CanDrive {
    accelerate(speed:number): void;
}

let car:CanDrive = {
    accelerate: function (speed:number) {
        // ...
    }
};
```

396 Generics

(Which can hold several types)

```
// Generics are types which can hold/ use several types
// We're only touching the very basics here - you can go MUCH more into detail

// Consider the Array object
```

```
let numberArray: Array<number>; // This array will only accept numbers


// Try to initialize it with strings


// numberArray = ['test']; // => Error
numberArray = [1,2,3];
```

397 Modules

(Helps us to load the different javascripts part in our project)

```
// TypeScript is modular, we can divide our code up over several files
// In Angular 2 we then use  "import {} from ''" to access the code in these files
// We export a class, interface, variable, ... by adding 'export' keyword in front of it


export class ExportedClass {
    // This class is exported
}
```