Angular
"formerly Angular 4"
Benefits worth the cost

Eng. Niveen Nasr El-Den
SD & Gaming CoE
iTi


A
2
A
4
A
5
A
?

Angular Team Says..

We believe that writing beautiful apps should be joyful and fun. We're building a platform for the future

Angular

- Angular is a platform that makes it easy to build applications with the web.
- Angular combines declarative templates, dependency injection, end to end tooling, and integrated best practices to solve development challenges.
- Latest version is 5.2.9

https://angular.io/docs

Angular

- Angular empowers developers to build applications that live on the web, mobile, or the desktop
- Angular is sponsored by Google.
- Angular is built by a team of engineers who share a passion for making web development feel effortless


Performance
Scalability
Security
Reusability
Architecture
Javascript
Typescript
AngularJS
Angular 2
Component-based UI
Mobile
Support UI
SEO Friendly
Angular 4

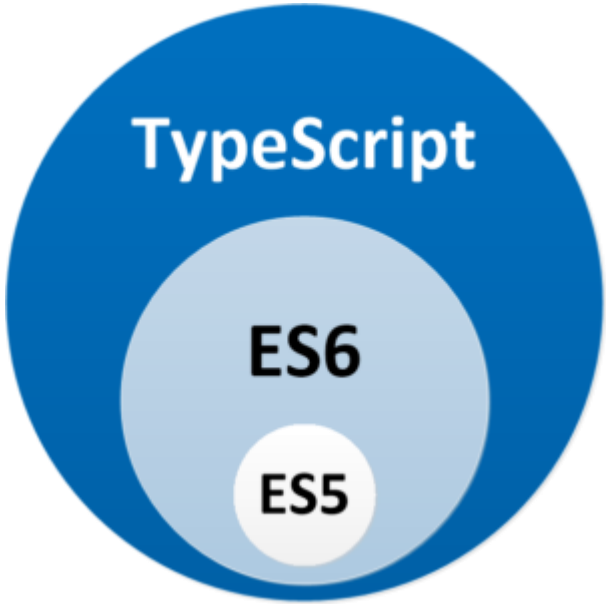
Angular vs AngularJS

- Angular refers to Angular 2+ while AngularJS refers to Angular 1.x
- Angular uses TypeScript
- Faster than previous versions
- No controllers, no scope
- Built on Component Model
- Not backward compatible with AngularJS
- Simpler

Prerequisites & Installation

- Tools
  - node
  - npm
- Text Editor
  - Visual Studio Code
  - Any other preferable one
- Knowledge
  - ES6
  - TypeScript
- Installation
  - npm install -g @angular/cli
  - npm install -g typscript

TypeScript



- TypeScript is a typed superset of JavaScript
- Supported By Microsoft
- It is a compiled language
  - it catches errors before runtime
- TypeScript is a JavaScript transpiler
  - It compiles to Javascript.
- Latest version is 2.7

http://www.typescriptlang.org/

Why TypeScript

- Better Tooling Support because it has static typing (like C#).
  - Intellisense and syntax checking
- Object oriented.
- All JavaScript code is TypeScript code, simply copy and paste
- All JavaScript libraries work with TypeScript
- Runs in any browser or host, on any OS

TypeScript Features

- Support standard JavaScript code with static typing
- Type inference
- Interface
- Generics
- Enums
- Access Modifiers
- etc.

TS= ES6 + Extra Features

TypeScript DataTypes

- Any :any ➔ bypasses type checking
- String : string
- Number : number
- Boolean : boolean
- Array
  - var arr: number[] = [1,2,3]
  - var arr: Array<number>= [1,2,3]
- Enum
  - enum Color {Blue,Red,Green} ;
  - var c : Color = Color.Blue;

Type Annotation

var identifier [[:type] = initVal];

- 4 ways to define a variable
  - var x; //type is any
  - var x = 10; //Type Inferred
  - var x : number;
  - var x : number = 10;

Type Assertion

- Give hint/ensure to compiler the value type for a given variable

- Example:
  let val:any=“abc xyz”
  (<string>val).length

Function

- We can specify returning type
- Use default parameter
- Use …rest parameter
- Syntax

```
function funNM (p1:type=val, p2?:type):returnType{
return ..;
}
```

Interface

- Describe minimum sets of public basic properties and method that a class must follow
- A contract that a class must adhere to;

```
interface human {
job: string;
hasJob?(): boolean;
}
class student extends Person implements human {
constructor(
public job: string,
public name: string,
public stage: number,
public ln: string,
public grade: string
) {
super(name, stage, ln);
}
}
```

Interface

- Interface as custom type

```
interface Point {
readonly x: number;
readonly y: number;
}

var p: Point = { x: 20, y: 30 };
var px = p.x;
```

Interface

- interfaces are also capable of describing function types

```
interface myFunInterface {
        (src: string, str: string): string;
}

let myFunInterfaceUse: myFunInterface;

myFunInterfaceUse = function(a: string, b: string): string {
        return a + " " + b;
};
```

readonly vs const

- The easiest way to remember whether to use readonly or const
  - variables use const
  - properties use readonly.

Type Definition

https://github.com/DefinitelyTyped/DefinitelyTyped

- To use 3rd party library in typescript
- Install type definition file via cmd
  - npm inatall –save @types/jquery
  - @types folder will be created

http://microsoft.github.io/TypeSearch/

- Add reference to declares a dependency on a package.
  - /// <reference types="jquery" />
  - Triple-slash directives are single-line comments containing a single XML tag. The contents of the comment are used as compiler directives.

Decorators

- It is a function that add metadata to the thing it is attached to
- Used to customize our class at design time
- Decorators are an experimental feature that may change in future releases.
  - tsc --target ES5 --experimentalDecorators f.ts
- Multiple decorators can be applied to a declaration

Decorator

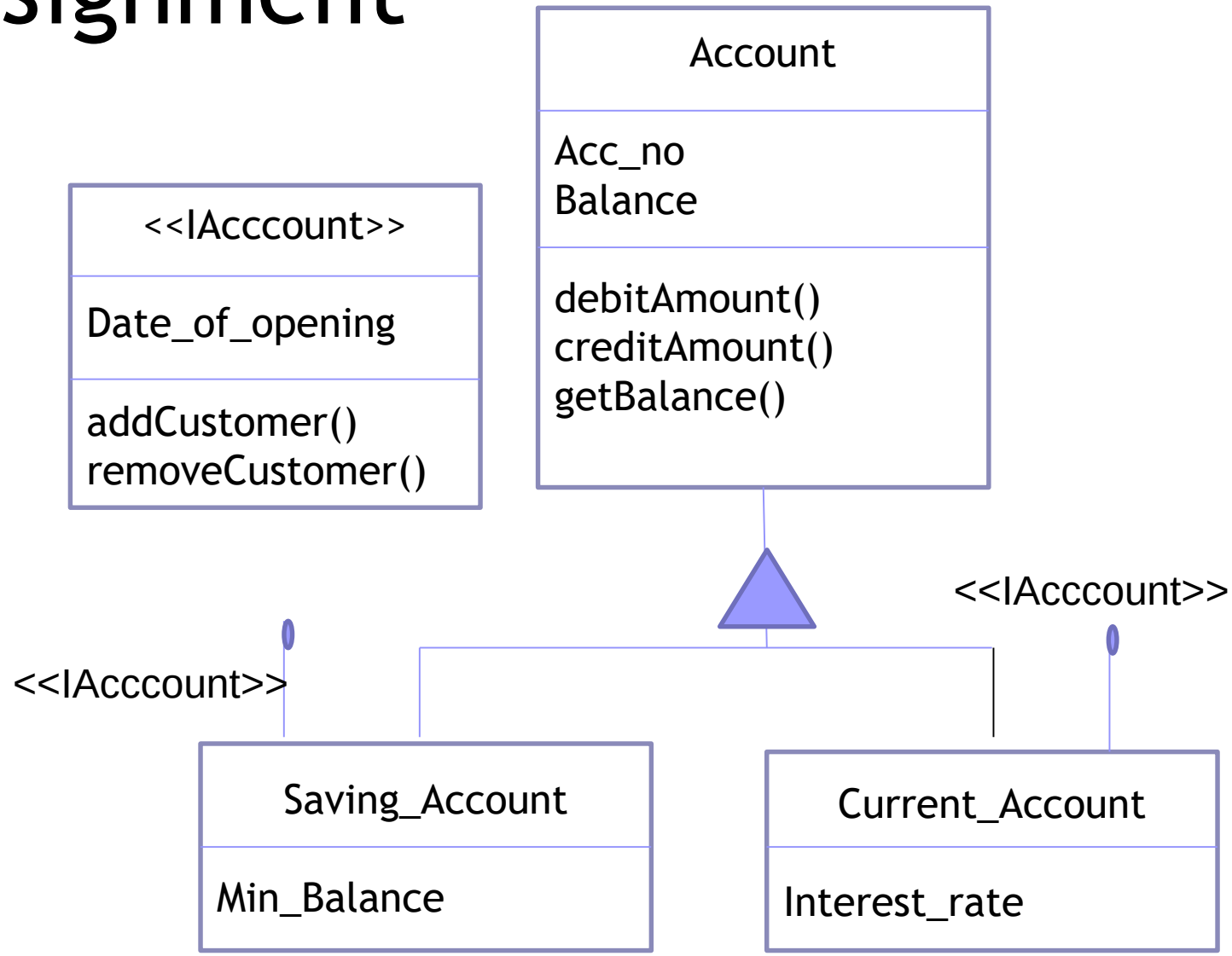
```
var s = new Employee("f", "Niveen", "Nasr", "Eng");
console.log(s); //var s= new Student("f", "Niveen", "NAsr", "Eng")
console.log(s.speak());
```

```
//decorator with argument
function course(topic: any) {
  //topic is the decorator argument
  return function(target: any) {
    //target is the class itself
    Object.defineProperty(target.prototype, "course", {
      value: () => "COURSE DESCRIPTION" + topic.topic + target
    });
  };
}
```

```
@course({ topic: "A2" })
class Employee {
  constructor(
    public gender: string,
    public fname: string,
    public lname: string,
    public job: string
  ) {}
  fullNm() {
    return `${this.fname} ${this.lname} final string`;
  }
  speak() {
    return `I'm ${this.fullNm()} i work as ${this.job}`;
  }
}
```

Assignment

Assignment


Account
Acc_no
Balance
debitAmount()
creditAmount()
getBalance()
<<IAcccount>>
Date_of_opening
addCustomer()
removeCustomer()
<<IAcccount>>
<<IAcccount>>
Saving_Account
Min_Balance
Current_Account
Interest_rate

Assignment

- Implement the give system in previous using TypeScript
- Implement any required class, properties and method
- Note: Account is an abstract class
- Assume your datatype