Angular JS vs Angular > 2 - the first version of Angular without TypeScript vs Angular

**Typescript**

Typescript is a superset of JS, uses the .ts file extension:  
  
let isDone: boolean = false;  
let hex: number = 0xf00d;

let binary: number = 0b1010;

let octal: number = 0o744;

let message: string = ‘’

let messages: string[] = [‘Test 1’,’Test 2’]

let messages: [‘Test 1’, number] = [Test 1’, 2.3]

<https://www.typescriptlang.org/docs/handbook/intro.html>

tsc index.ts –target es6 - convert Typescript to JS based on a ECMA standard, транспилация

!!! you can dynamically change the type of a variable, but you can assign the value of a variable from different type to the variable, for which you want its type to be changed !!!

**Namespace in Angular:**

namespace Vehicle {

export enum Car {

Peugeout = 'Message 1',

Citroen = 1

}

export enum Truck {

Ford = 'Ford',

RangeRover = 2

}

}

let carMessages: [number, Vehicle.Car] = [1, Vehicle.Car.Citroen]

**constructor property promotion is available in Typescript as well:**   
  
class Suv {

public model: string;

constructor(model: string) {

this.model = model;

}

}

= >  
  
class Suv {

constructor(public model: string) {

}

}

**method overriding**

class Suv {

constructor(protected model: string) {

}

drive():string {

return `I am driving this ${this.model}`;

}

}

export class SmallerSuv extends Suv {

constructor(model: string, private size: number) {

super(model);

}

override drive(): string {

return super.drive() + ' called from child!'

}

park(): void {

console.log(`I am parking this ${this.model} in size ${this.size}`)

}

}

**we can set a custom type**  
  
// creating a custom type

function printLabel(labelledObj: {label: string}){

console.log(labelledObj.label);

}

**Generics** - primary used for data structures

function identity<T>(arg: T): T {

return arg;

}

let output = identity<string>('myString');

**!isNan(+arg)** - check if arg is number

!!! interface are possible in Typescript with the interface keyword and we can create objects of set interface as well as classes, however, we can not check if an object is instance of interface and interfaces are converted to basic JavaScript classes !!!

!!! native Javascript feature is # to define private or static properties and it is different from the Typescript:   
  
<https://stackoverflow.com/questions/59641564/what-are-the-differences-between-the-private-keyword-and-private-fields-in-types>

The private keyword in TypeScript is a compile time annotation. It tells the compiler that a property should only be accessible inside that class  
  
vs  
  
Private fields ensure that properties are kept private at runtime: !!!

**Install Angular:**   
  
npm install -g @angular/cli

ng new some-app

cd some-app  
//starting the server:  
ng serve   
ng build  
  
**Start Angular project:**

[15:18:32] donetianpetkov@Donetian-Petkov-NEWs-MacBook-Pro [ ~/WebstormProjects/angular\_first/soft-uni-day-one ] $ ng serve

✔ Browser application bundle generation complete.

Initial Chunk Files | Names | Raw Size

vendor.js | vendor | 1.70 MB |

polyfills.js | polyfills | 294.81 kB |

styles.css, styles.js | styles | 173.24 kB |

main.js | main | 47.69 kB |

runtime.js | runtime | 6.53 kB |

| Initial Total | 2.21 MB

Build at: 2022-04-03T12:18:54.282Z - Hash: 19f21a7f6345c537 - Time: 11344ms

\*\* Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ \*\*

✔ Compiled successfully.

**Build:**

[15:01:27] donetianpetkov@Donetian-Petkov-NEWs-MacBook-Pro [ ~/WebstormProjects/angular\_first/soft-uni-day-one ] $ ng build

✔ Browser application bundle generation complete.

✔ Copying assets complete.

✔ Index html generation complete.

Initial Chunk Files | Names | Raw Size | Estimated Transfer Size

main.20a8ef08287cd70f.js | main | 116.45 kB | 34.88 kB

polyfills.bd73d25360343036.js | polyfills | 33.03 kB | 10.64 kB

runtime.20c563868d71cc52.js | runtime | 1.06 kB | 606 bytes

styles.ef46db3751d8e999.css | styles | 0 bytes | -

| Initial Total | 150.54 kB | 46.12 kB

Build at: 2022-04-03T12:24:06.435Z - Hash: 06132fb6ef214ebb - Time: 17633ms

[15:25:47] donetianpetkov@Donetian-Petkov-NEWs-MacBook-Pro [ ~/WebstormProjects/angular\_first/soft-uni-day-one ] $ cd dist/

[15:25:55] donetianpetkov@Donetian-Petkov-NEWs-MacBook-Pro [ ~/WebstormProjects/angular\_first/soft-uni-day-one/dist ] $ ls

soft-uni-day-one

[15:25:55] donetianpetkov@Donetian-Petkov-NEWs-MacBook-Pro [ ~/WebstormProjects/angular\_first/soft-uni-day-one/dist ] $ cd soft-uni-day-one/

[15:25:58] donetianpetkov@Donetian-Petkov-NEWs-MacBook-Pro [ ~/WebstormProjects/angular\_first/soft-uni-day-one/dist/soft-uni-day-one ] $ ls

3rdpartylicenses.txt index.html polyfills.bd73d25360343036.js styles.ef46db3751d8e999.css

favicon.ico main.20a8ef08287cd70f.js runtime.20c563868d71cc52.js

**config file: angular.json** - how the project will be built

<https://stackblitz.com/> - online IDE for JS, Angular, React and etc  
  
<https://plnkr.co/> - online IDE for JS, Angular, React and etc

**tsconfig.json** - how the Typescript and the project will be compiled

**Component in Angular:**

import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'soft-uni-day-one';

}

**Manually Creating Components in Angular:**  
  
1. Create component ts in the src/app folder -> define component inside it:  
  
​​import { Component } from '@angular/core';

@Component({

selector: 'app-test',

template: `<div>Test</div>`,

styles: [`div {color:blue}`]

})

export class TestComponent {

title = 'testing';

}

2. then include the component in the app.module.ts:  
  
import {TestComponent} from "./test.component";

@NgModule({

declarations: [

AppComponent,

TestComponent

3. and finally set the selector in the app.component.html or the template file where you want to use the component:   
  
<app-test></app-test>

**Automatically Creating Components:**

[19:21:59] donetianpetkov@Donetian-Petkov-NEWs-MacBook-Pro [ ~/WebstormProjects/angular\_first/soft-uni-day-one ] $ ng generate component example

CREATE src/app/example/example.component.css (0 bytes)

CREATE src/app/example/example.component.html (22 bytes)

CREATE src/app/example/example.component.spec.ts (633 bytes)

CREATE src/app/example/example.component.ts (279 bytes)

UPDATE src/app/app.module.ts (467 bytes)

**ng g c example** - shorter syntax

!!! the ng g c finds the closest module to the newly created component and adds the component in the Declaration !!!

!!! Every Angular project has a module file (App.Module) where the Angular app is launched:  
  
import { NgModule } from '@angular/core';

import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';

import {TestComponent} from "./test.component";

import { ExampleComponent } from './example/example.component';

@NgModule({

declarations: [

AppComponent,

TestComponent,

ExampleComponent

],

imports: [

BrowserModule

],

providers: [],

bootstrap: [AppComponent]

})

export class AppModule { }

in the bootstrap: we set which components will be launched when the app is executed. We need to also specify in the index.html the selector for these components:  
  
**app.component.ts:**  
  
import { Component } from '@angular/core';

@Component({

selector: 'app-root',

templateUrl: './app.component.html',

styleUrls: ['./app.component.css']

})

export class AppComponent {

title = 'soft-uni-day-one';

}

->   
  
**index.html:**  
  
<body>

<app-root></app-root>

</body>

!!!!

**interpolation is only for showing text in HTML - {{}}**

**if we need to use a value from the class we use square brackets - in other words we bind:**   
  
in the template / html file for the component:

<input [value]=”title”> - title will be the title property in the class  
  
**events in Angular:**  
  
in the template / html file for the component:  
  
<button (click)=”nameOfFunction()”>Click Me</button> - when we click the button the nameOfFunction will be executed  
  
in the TS file for the component:   
  
export class AppComponent {

nameOfFunction(): returnType {

// some functionality  
 }  
}

**Directives in Angular**special commands for working within the HTML template - \*ngFor, \*ngIf

in the component ts file:   
  
users = [

{

name: 'Ivan',

age: 20

},

{

name: 'Naiden',

age: 30

},

{

name: 'Ivana',

age: 50

}

]

in the template / html file:

<ul>

<li \*ngFor="let user of users">

{{user.name}}

</li>

</ul>

in the template / html file:  
  
<div \*ngIf="showText">VISIBLE</div>

<button (click)="toggleText()">{{showText ? 'HIDE TEXT' : 'SHOW TEXT'}}</button>

in the component ts file:   
  
showText = true;  
  
toggleText(): void {

this.showText = !this.showText;

}

**the event in the callback functions is $event**

<button (click)="toggleText($event)">{{showText ? 'HIDE TEXT' : 'SHOW TEXT'}}</button>

->  
  
toggleText(event: MouseEvent): void {

event.preventDefault();

this.showText = !this.showText;

}

**binding / adding CSS classes to elements:**in html:   
  
<div \*ngIf="showText" [class]="classes">VISIBLE</div>

in component:  
  
export class ExampleComponent implements OnInit {

classes=['test', 'test-1'];

in component css:  
  
.test {

color:red;

}

**conditional class - >** [class.special]=”isSpecial”

if isSpecial in the class is true, the element will have CSS class special

**adding styles inline:**   
  
[style.color]=”isSpecial ? ‘red’ : ‘blue’”

**Templates variables**calling HTML elements in other elements - this is how we can take the values of input fields by clicking a button without a form:   
  
<input #inputElement type="text">

<button (click)="changeTitleHandler(inputElement.value)">Change Title</button>

<p>{{title}}</p>

**nullsafe operators**

{{game?.title}}

!!!! not recommended to use two-way data binding - [(ngModel)] !!!  
  
**Lifecycle Hooks**

ngOnInit - after constructor before content  
ngAfterViewInIt  
ngOnDestroy   
ngOnChanges - when Input changes, we can call ngOnChanges(simpleChanges: SimpleChanges) { console.log(simpleChanges} to see the change, need for the reference to be changed  
ngDoCheck

Презентационни компоненти vs Container компоненти - display data vs process data

!!! we can set a property to a class without initializing it with the ! after the prop - user!: IUser   
  
Another alternative: user: IUser | undefined; !!!

**Passing input to child components:**  
  
**<app-user-list-item>:**ts:  
  
@Input() **user**!: IUser;

html:

<span>{{**user**?.name}}</span>

<span>{{**user**?.age}}</span>

->  
  
**<app-user-list>:**

ts:  
  
@Input() **userArray**: {name: string, age: number}[] = [];

html:

<div id="container">

<**app-user-list-item** \*ngFor="let users of **userArray**" [**user**]="**users**"></app-user-list-item>

</div>

->

**<app-root>:**

html:

<**app-user-list** [**userArray**]="**users**"></app-user-list>  
  
ts:  
  
users = [

{

name: 'Ivan',

age: 20

},

{

name: 'Naiden',

age: 30

},

{

name: 'Ivana',

age: 50

}

]

**Passing Output to Parent Components:**  
  
same example:   
  
**<app-user-list>:**  
  
ts:  
  
@Output() addUser = new EventEmitter<IUser>();

addNewUser(userNameInput: HTMLInputElement, userAgeInput: HTMLInputElement): void {

const {value: name} = userNameInput;

const {valueAsNumber: age} = userAgeInput;

this.addUser.emit({name, age}); // triggering the addUser and emitting the name and age to the parent component - addUser is an event and in the parent component we have addNewUserHandler, which handles this event

userAgeInput.value = '';

userNameInput.value = '';

}

html:  
  
<div>

<input #userNameInput type="text">

<input #userAgeInput type="number">

<button (click)="addNewUser(userNameInput, userAgeInput)">Add New User</button>

</div>

**<app-root>:**

ts:  
  
addNewUserHandler(newUser: IUser){

this.users.push(newUser);

}

html:

<app-user-list [userArray]="users" (addUser)="addNewUserHandler($event)"></app-user-list>

**Debugging in Angular** - setting a debugger after the line we want to be executed:  
  
constructor() {

console.log("I am in the constructor")

debugger;

}  
  
Reloading the page it will spawn the debugger on the browser