Angular 4

My very first application

Authors:
Michał Michalczuk, Bartosz Bobin

13 september 2017

goyello

Plan for today

Time-box: 2h 30 min

- Web Client - Web Server: where Angular lives
- Short about Angular – when to use it
- TypeScript: optional static typed JavaScript
- Angular-cli: fast, easy-to-use tool for building and running Angular projects
- Notes list – classic, simple example application

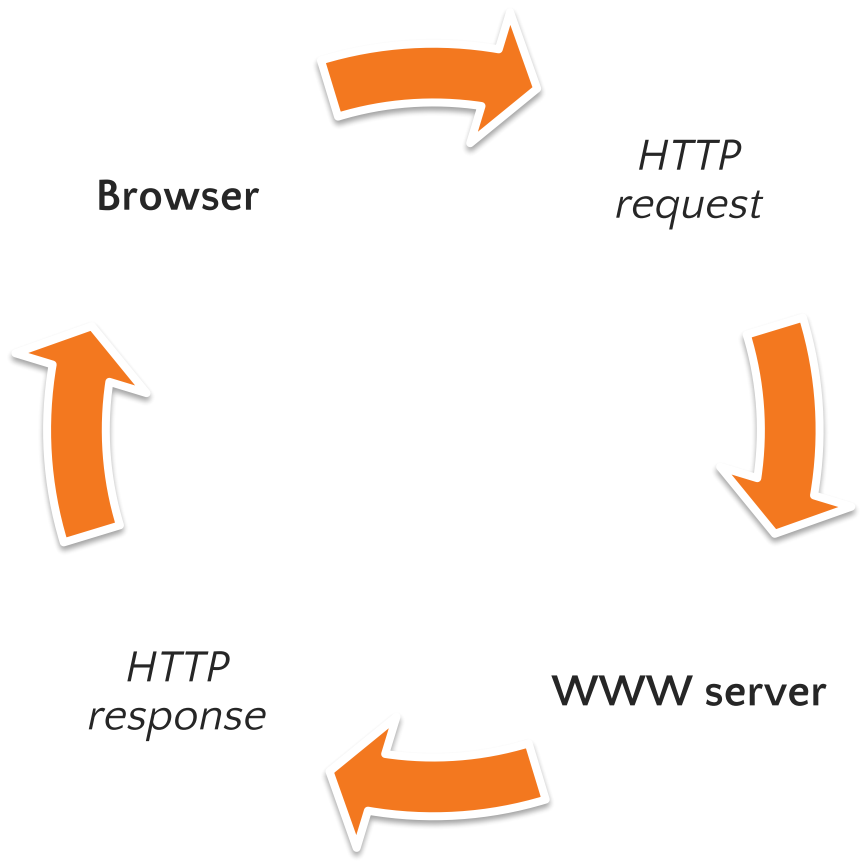
Clone workshops repository:
github.com/michalczukm/gy-angular-workshops

Request – Response

How the internet works

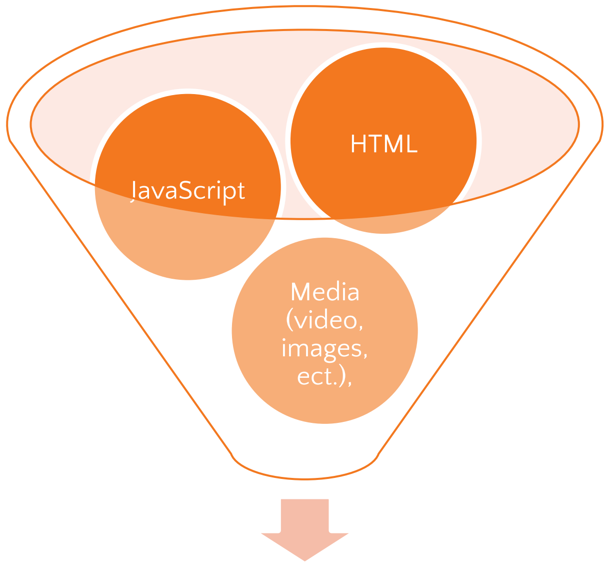
HTTP protocol – what we have to know?

- HTTP allows us to browse web pages
- Describes communication between **browser** *(client side)* and **www server** (server side)
- Works in **Request-Response** flow
- Browser has to send dozen of HTTP requests to render one page



What browser can do?

- Send request to *WWW server*
- Receive and process response
- Display (*render*) *HTML code*
- **Execute *JavaScript*** *code from response*
- Collect data from user (forms)



Web page

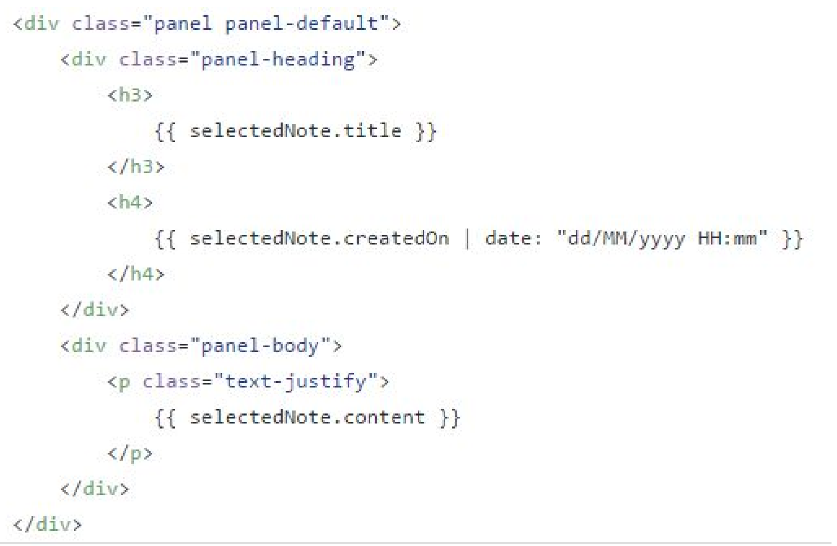
Single Page Application

We are “fixing” HTTP

How does Facebook, Gmail, etc works

- Server sends data (model) and HTML templates separately
- *HTML templates (view)* describes how to present data
- *JavaScript code* interprets template and display result to user
- Browser asks for more data in the background (*AJAX*)

- Example view



```
<div class="panel panel-default">
    <div class="panel-heading">
        <h3>
            {{ selectedNote.title }}
        </h3>
        <h4>
            {{ selectedNote.createdOn | date: "dd/MM/yyyy HH:mm" }}
        </h4>
    </div>
    <div class="panel-body">
        <p class="text-justify">
            {{ selectedNote.content }}
        </p>
    </div>
</div>
```

- Example model

```
{
    title: "Ala ma kota",
    createdOnDate: "2017-04-04T12:00:00.000",
    author: "Goyello",
    description: "just another JSON document
}
```

- SPA framework
- Running under browser control thanks to *JavaScript*
- Known as *Angular 4*
- Created and maintained by *Google*
- Brief: Angular allows us to create <u>dynamic</u> and <u>interactive</u> web applications

https://angular.io

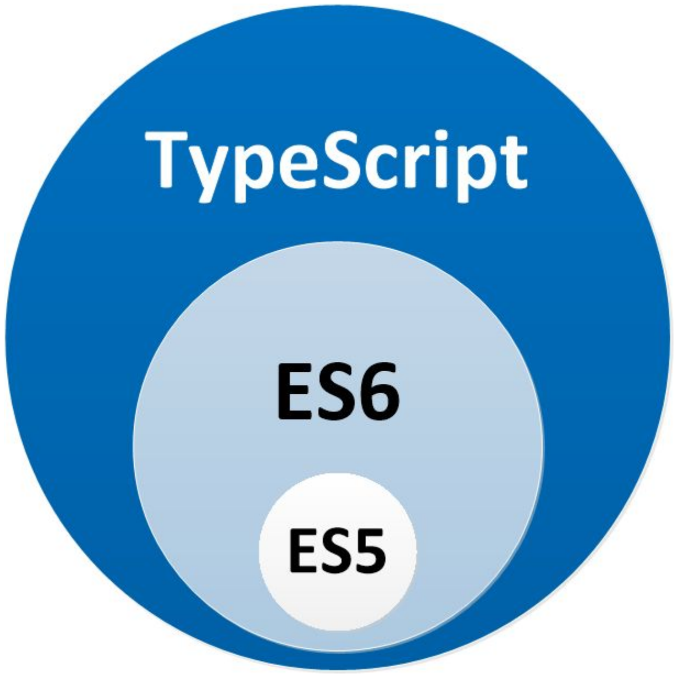
Angular 4 language of choice

TypeScript … in 15 minutes

- Superset of JavaScript
- JavaScript code is legal TypeScript code
- Develop by Microsoft since 2012
- Open project on GitHub
- v **2.5.x**

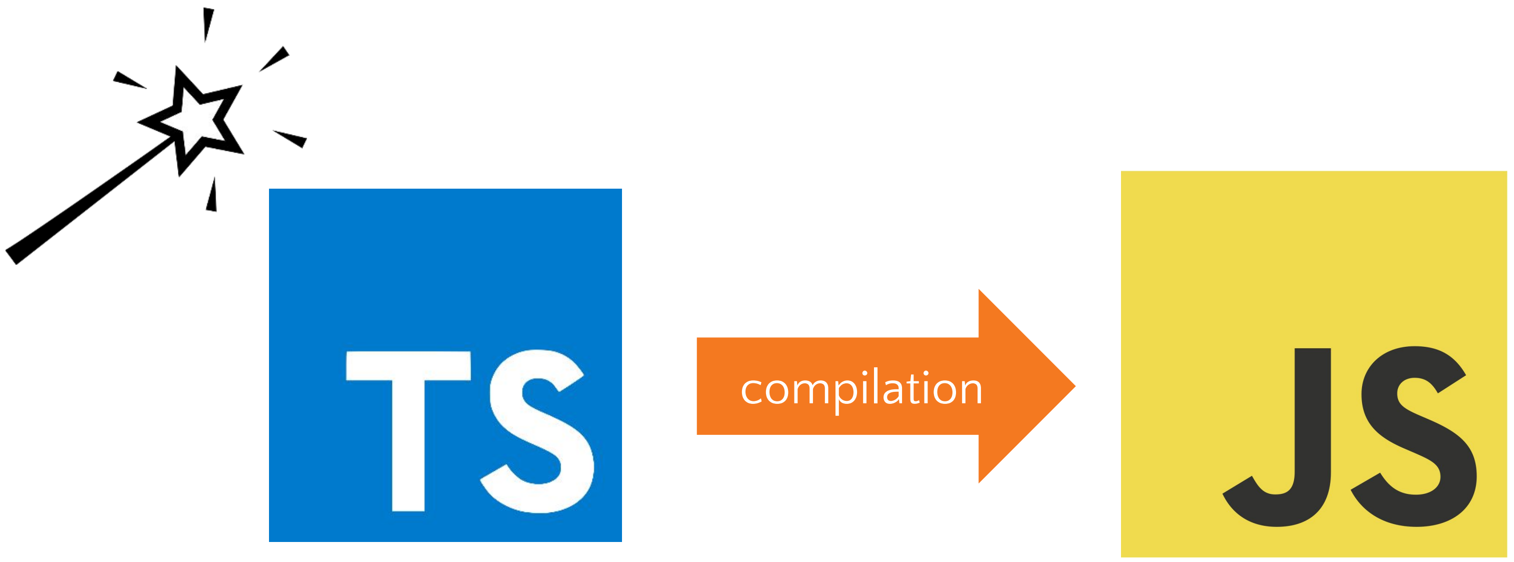
https://github.com/Microsoft/TypeScript

TypeScript **vs** JavaScript



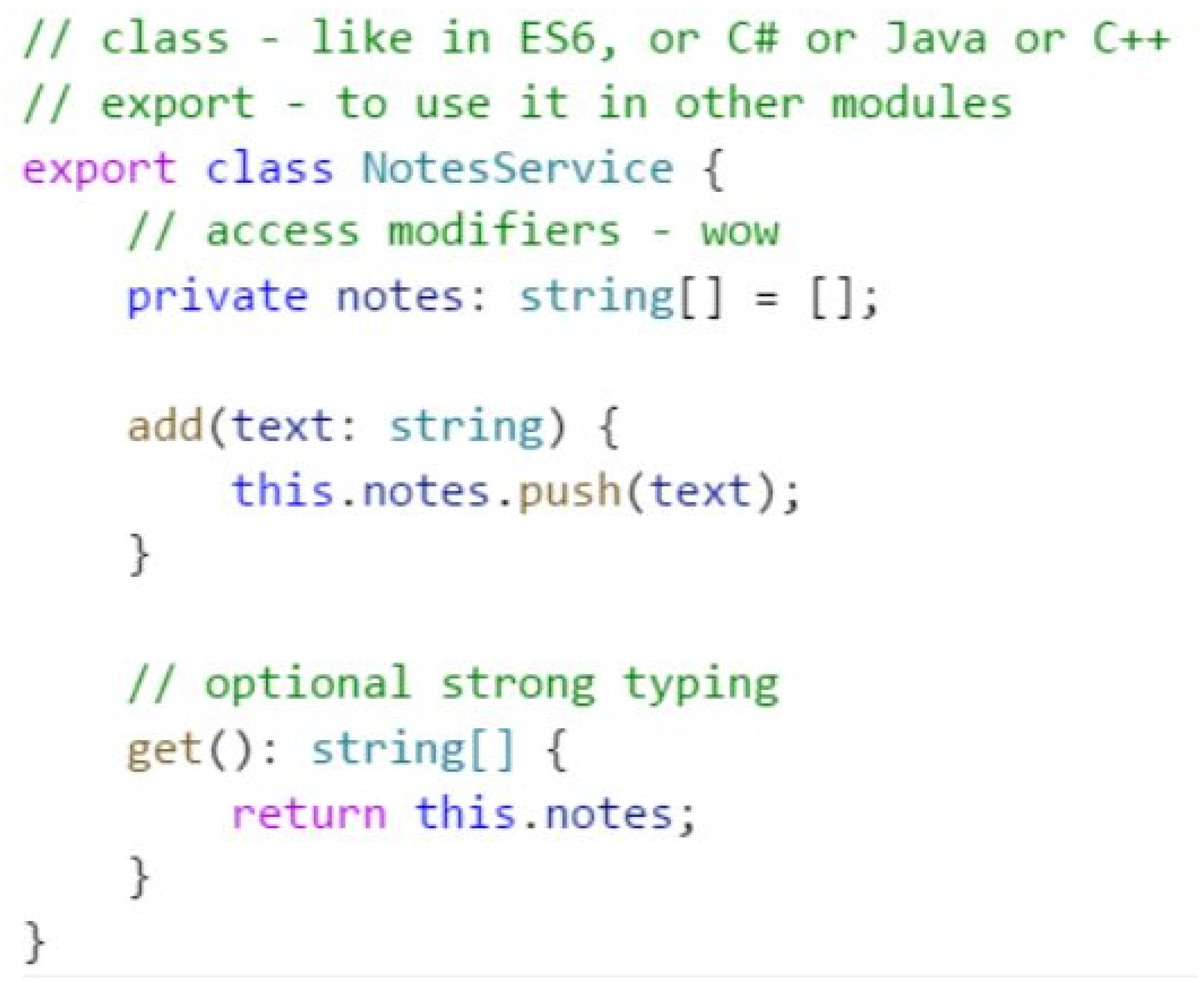
- ES = EcmaScript
- ES5 vs ES6
- TypeScript implements ES6 + ES2017 + more

TypeScript **vs** JavaScript



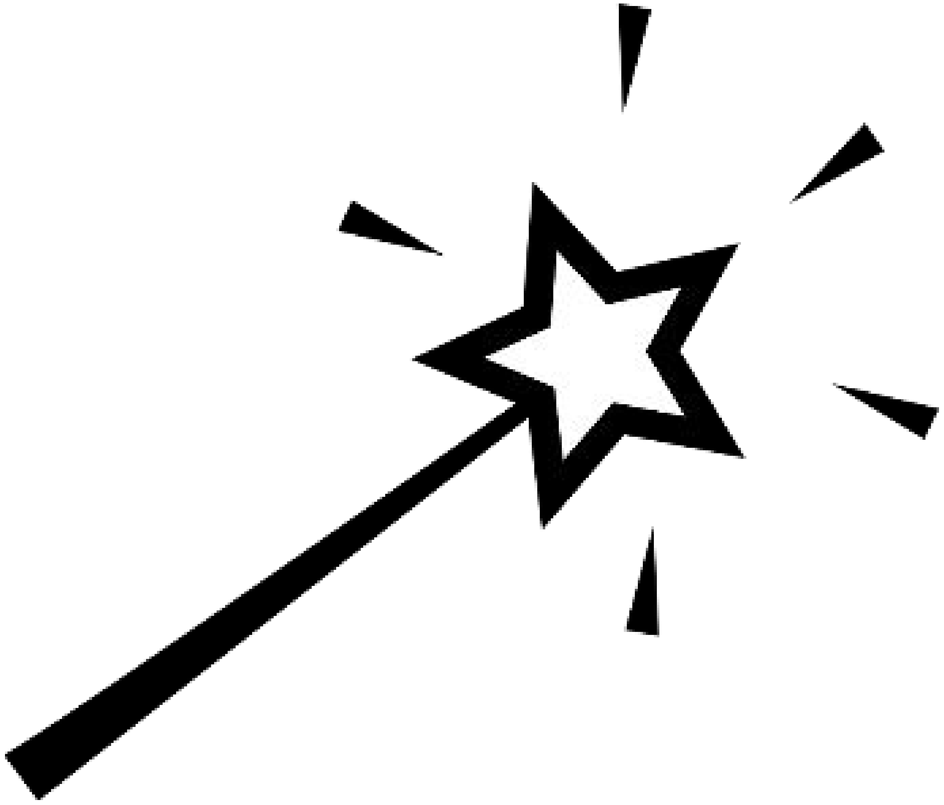
- TypeScript have to be compiled JavaScript
- Browser has to interpret our code (JavaScript)

TypeScript: example

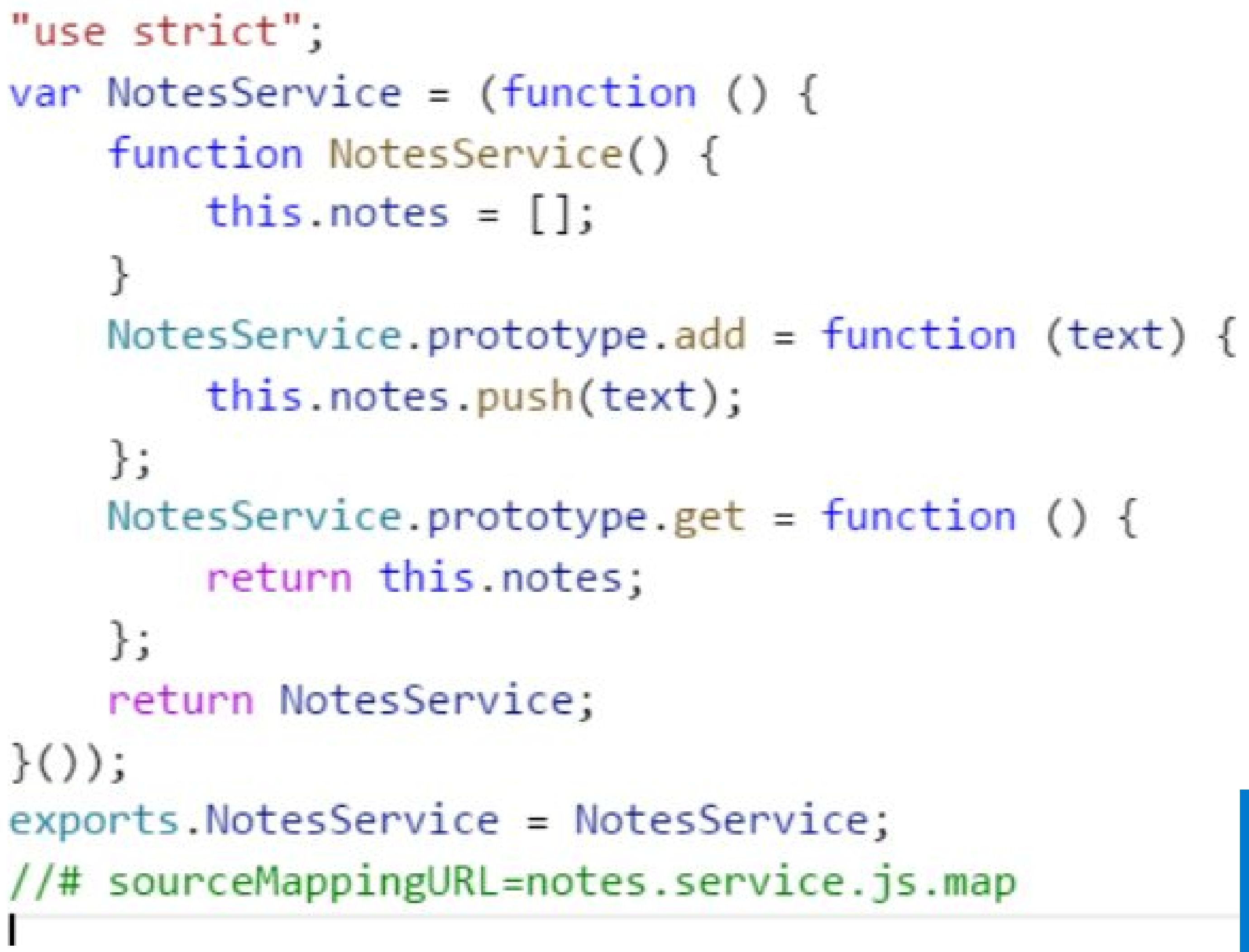


```
// class - like in ES6, or C# or Java or C++
// export - to use it in other modules
export class NotesService {
    // access modifiers - wow
    private notes: string[] = [];

    add(text: string) {
        this.notes.push(text);
    }

    // optional strong typing
    get(): string[] {
        return this.notes;
    }
}
```

Do you know „module” pattern in JavaScript ?

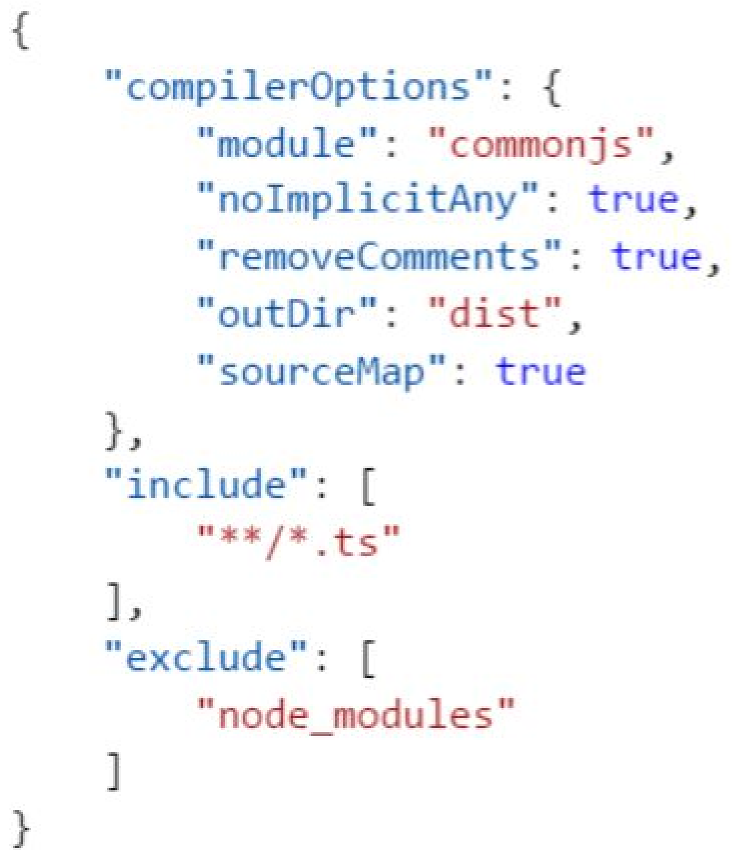


TypeScript: same example after compilation to ES5



```
"use strict";
var NotesService = (function () {
    function NotesService() {
        this.notes = [];
    }
    NotesService.prototype.add = function (text) {
        this.notes.push(text);
    };
    NotesService.prototype.get = function () {
        return this.notes;
    };
    return NotesService;
}());
exports.NotesService = NotesService;
//# sourceMappingURL=notes.service.js.map
```

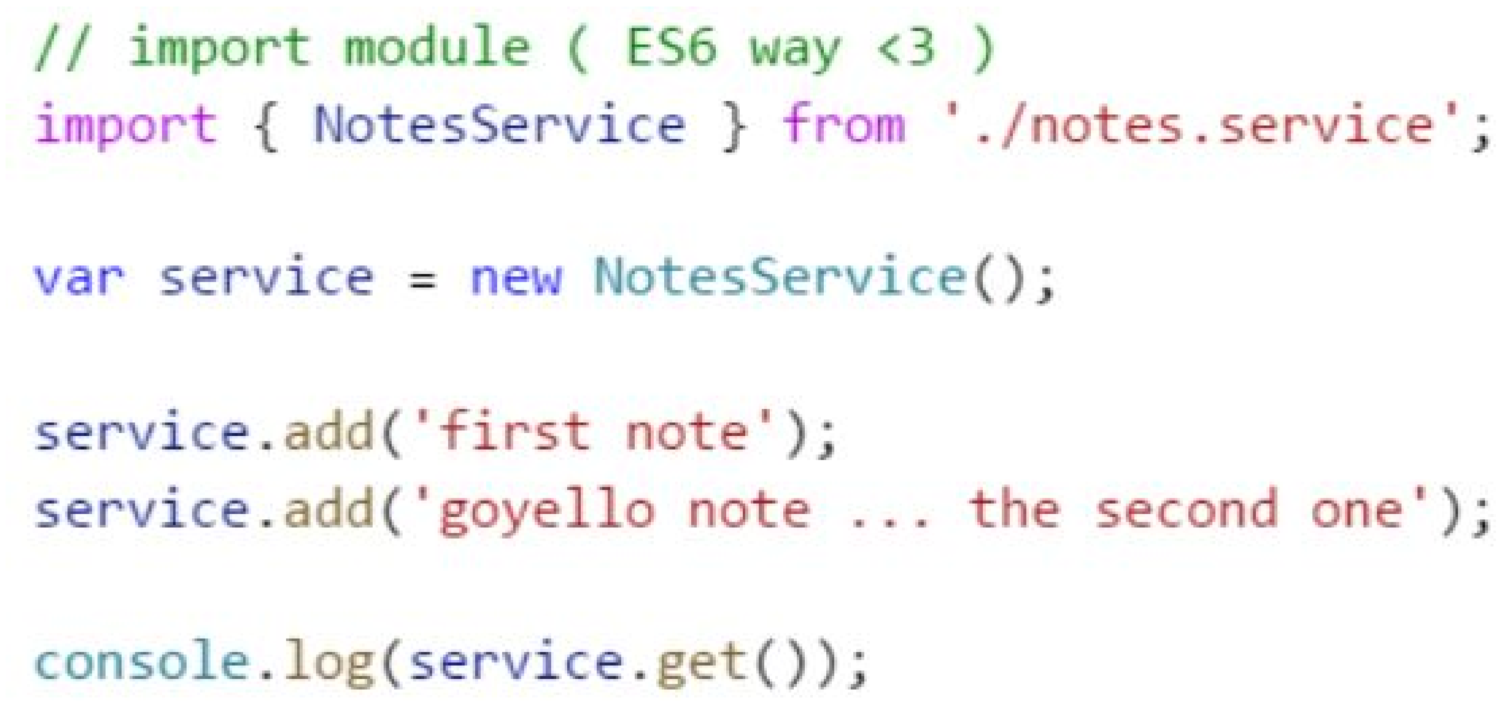
tsconfig.json – setup the compiler



```
{
    "compilerOptions": {
        "module": "commonjs",
        "noImplicitAny": true,
        "removeComments": true,
        "outDir": "dist",
        "sourceMap": true
    },
    "include": [
        "**/*.ts"
    ],
    "exclude": [
        "node_modules"
    ]
}
```

- What files
- Form of compilation result (ES version)
- Where to put result files
- More options (https://www.typescriptlang.org/docs/handbook/tsconfig-json.html)

Let's use our *NoteService*



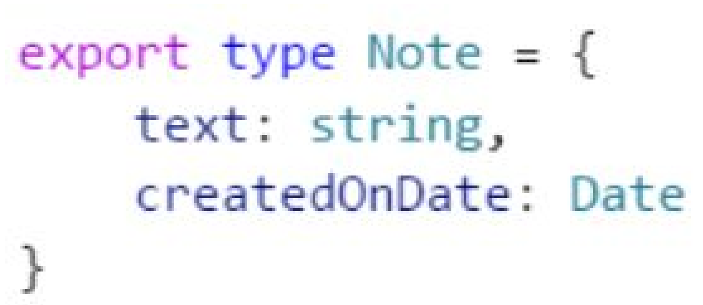
```
// import module ( ES6 way <3 )
import { NotesService } from './notes.service';

var service = new NotesService();

service.add('first note');
service.add('goyello note ... the second one');

console.log(service.get());
```

Few more TypeScript language elements

Interfaces

```
export interface NotesServiceInterface {
    add(text: string): void;
    get(): string[];
}

// class - like in ES6, or C# or Java or C++
// export - to use it in other modules.
// Now it has to implement interface
export class NotesService implements NotesServiceInterface {
    // access modifiers - wow
    private notes: string[] = [];

    add(text: string) {
        this.notes.push(text);
    }

    // optional strong typing
    get(): string[] {
        return this.notes;
    }
}
```

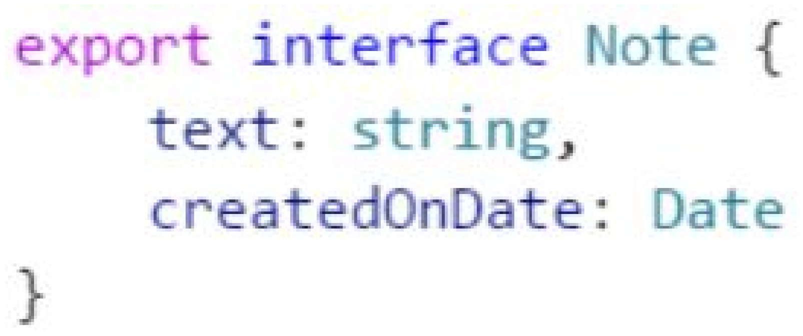
Types and data modeling



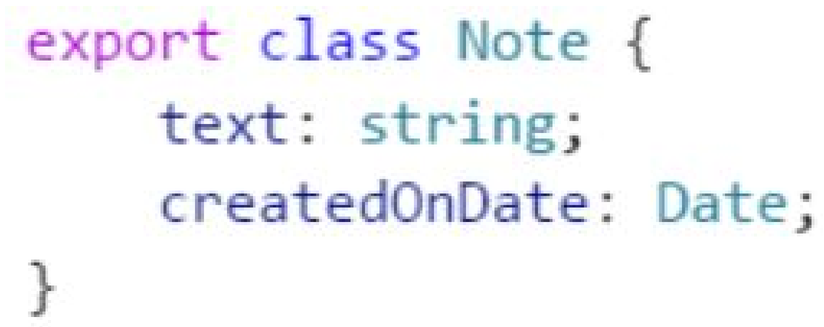
```
export type Note = {
    text: string,
    createdOnDate: Date
}
```

Simple models.

Describe them using `type` or `interface`.



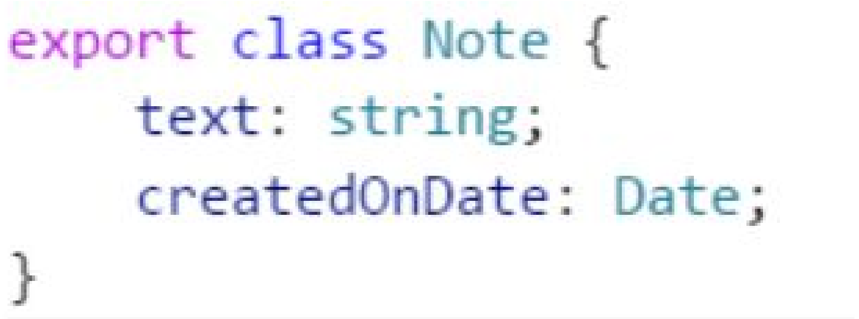
```
export interface Note {
    text: string,
    createdOnDate: Date
}
```



```
export class Note {
    text: string;
    createdOnDate: Date;
}
```

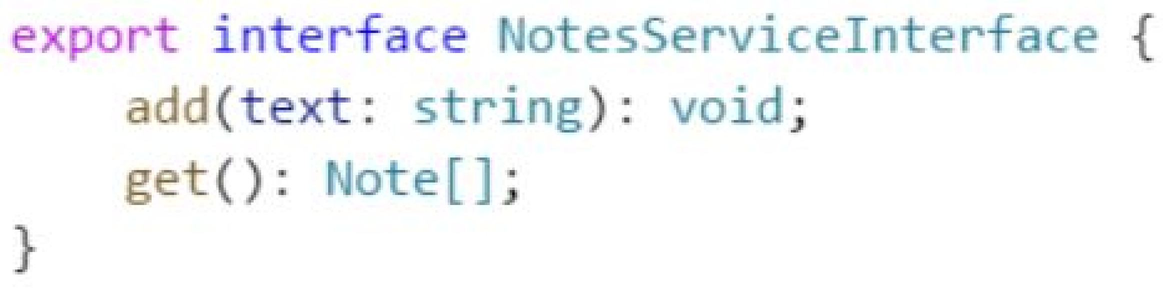
Use classes when you want to model behaviour (methods)

Use *Note* type– to model the note it in our service



```
export class Note {
    text: string;
    createdOnDate: Date;
}
```

Nasz serwis powinien używać `Note`. Zmieniamy interface który implementuje.



```
export interface NotesServiceInterface {
    add(text: string): void;
    get(): Note[];
}
```

NotesService after small change

```
export class NotesService implements NotesServiceInterface {
    // access modifiers - wow
    private notes: Note[] = [];

    add(text: string) {
        const newNote = {
            text: text,
            createdOnDate: new Date()
        } as Note;

        this.notes.push(newNote);
    }

    // optional strong typing
    get(): Note[] {
        return this.notes;
    }
}
```

TS

Short summary

- Optional static typed language
- Superset of JavaScript
- Has to be compiled to JavaScript, to use it in browser
- Uses modules and imports from ES6

What more?

- Advanced types management
- Generic types
- Functional programming – as strong direction of lang. development
- You can use JavaScript in TypeScript
- You can use any JavaScript library

Angular, TypeScript, NPM, angular-cli

Toolbox

What we need to start working

- **Angular** – SPA *client-side* framework
- **TypeScript** – JavaScript superset
- **NPM** – node package manager
- **Angular CLI** – our “magic wand”
  - Create project and generate application template
  - Supports us by code generation features
  - Builds our project and minify it
  - Lunches simple web server, for development purposes

Let's go!

Application for taking notes

github.com/michalczukm/gy-angular-workshops

Thank you for attention

- Application code: https://github.com/michalczukm/gy-angular-workshops
- TypeScript tutorial: https://www.typescriptlang.org/docs/tutorial.html
- Angular tutorial (its really great): https://angular.io/docs/ts/latest/tutorial/
- Angular-CLI: https://github.com/angular/angular-cli#usage

Contact us:

bartosz.bobin@goyello.com

michal.michalczuk@goyello.com