

TypeScript



agenda

**** min: wprowadzenie do TypeScript**

5 min: przerwa

**** min: zabawa (z) klasami**

5 min: przerwa

**** min: zabawa (z) funkcjami**

koniec ;)

A satellite image of a tropical cyclone, showing a well-defined eye and a dense, swirling cloud structure over a dark blue ocean. The text "wprowadzenie" is overlaid on the left side of the image.

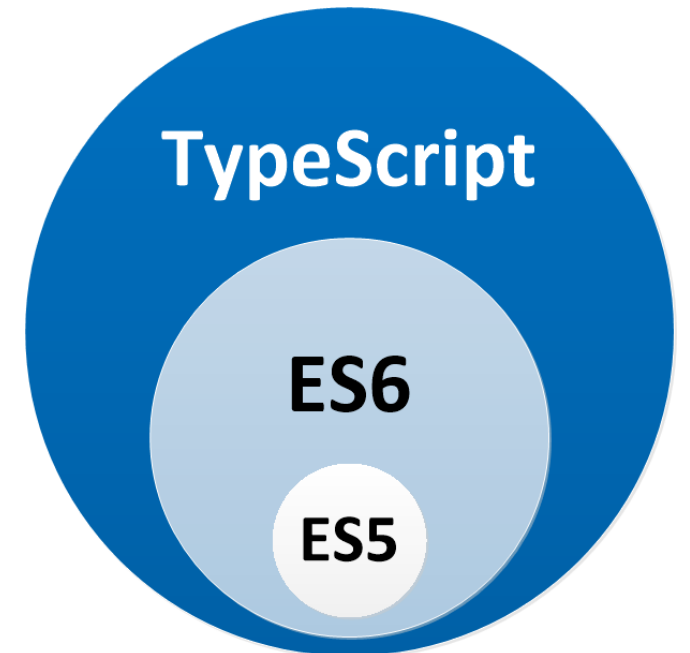
wprowadzenie

typescript

- code transpiled into js
- decorators (~annotations) - allow us to decorate classes with annotations and properties as well as meta-functionality

TS = ES6 + Types + Annotations

<http://www.typescriptlang.org/>

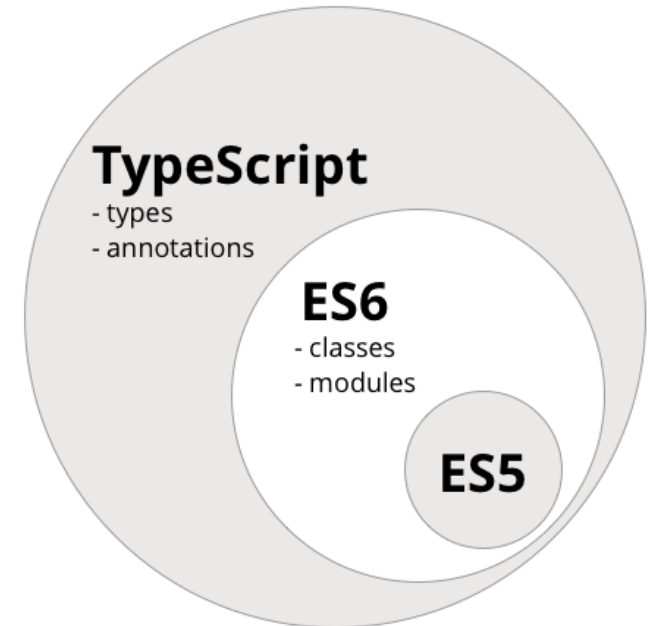


typescript

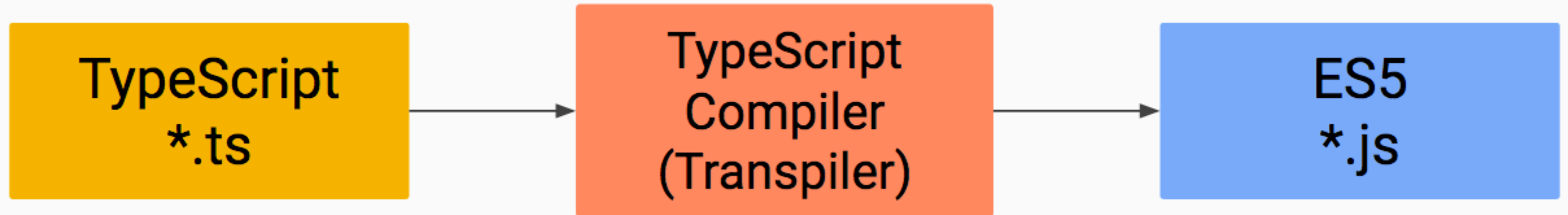
- code transpiled into js
- decorators (~annotations) - allow us to decorate classes with annotations and properties as well as meta-functionality

TS = ES6 + Types + Annotations

<http://www.typescriptlang.org/>



typescript transpilation



typescript vs javascript

```
class Greeter {  
  greeting: string;  
  constructor(message: string) {  
    this.greeting = message;  
  }  
  
  greet() {  
    return "Hello, " + this.greeting;  
  }  
}  
  
let greeter = new Greeter("world");
```

TS

```
var Greeter = /** @class */  
(function () {  
  function Greeter(message) {  
    this.greeting = message;  
  }  
  
  Greeter.prototype.greet  
    = function () {  
      return "Hello, " + this.greeting;  
    };  
  
  return Greeter;  
})();  
  
var greeter = new Greeter("world");  
greeter.greet();
```

JS

cli

files + options >> tsc >> core compiler >> js files

use:

- cmd:

```
tsc **/*.ts --target=es5 --sourcemap=true
```

- tsconfig.json:

```
1  {
2    "compileOnSave": false,
3    "compilerOptions": {
4      "baseUrl": ".",
5      "outDir": "dist/out-tsc",
6      "sourceMap": true,
7      "declaration": false,
```


typescript basic types

```
// boolean
var isTrue: boolean = true;

//number
var num: number = 1;

//string
var name: string = "CodingDefined";

//array
var arr: number[] = [3, 4, 5];
```

typescript basic types

```
// Number
```

```
let decimal: number = 6;
```

```
let hex: number = 0xf00d;
```

```
let binary: number = 0b1010;
```

```
let octal: number = 0o744;
```

typescript basic types

```
// String
```

```
let color: string = "blue";  
color = 'red';
```

```
let fullName: string = `Bob Bobbington`;  
let age: number = 37;  
let sentence: string = `Hello, my name is ${ fullName }.  
I'll be ${ age + 1 } years old next month.`;
```


typescript basic types

// Array

```
let list: number[] = [1, 2, 3];
```

```
let list2: Array<number> = [1, 2, 3];
```

// Enum

```
enum Color {Red = 1, Green, Blue}
```

```
let c: Color = Color.Green;
```

typescript basic types

```
// any
```

```
let notSure: any = 4;  
notSure.ifItExists(); // okay, ...might exist at runtime  
notSure.toFixed(); // okay, ...exists (but the compiler  
doesn't check)
```

```
let prettySure: Object = 4;  
prettySure.toFixed(); // Error: Property 'toFixed'  
doesn't exist on type 'Object'.
```

typescript basic types

```
// Void
```

```
function warnUser(): void {  
    console.log('This is my warning message');  
}
```

```
// Not much else we can assign to these variables!
```

```
let u: undefined = undefined; // --strictNullChecks
```

```
let n: null = null; // --strictNullChecks
```


typescript basic types

```
// Function returning never must have unreachable end point
function error(message: string): never {
    throw new Error(message);
}
```

```
// Function returning never must have unreachable end point
function infiniteLoop(): never {
    while (true) {
    }
}
```

type assertions

```
let someValue: any = "this is a string";  
let strLength: number =  
    (<string>someValue).length;
```

```
let someValue: any = "this is a string";  
let strLength: number =  
    (someValue as string).length;
```

union types

```
let data: string | number;
```

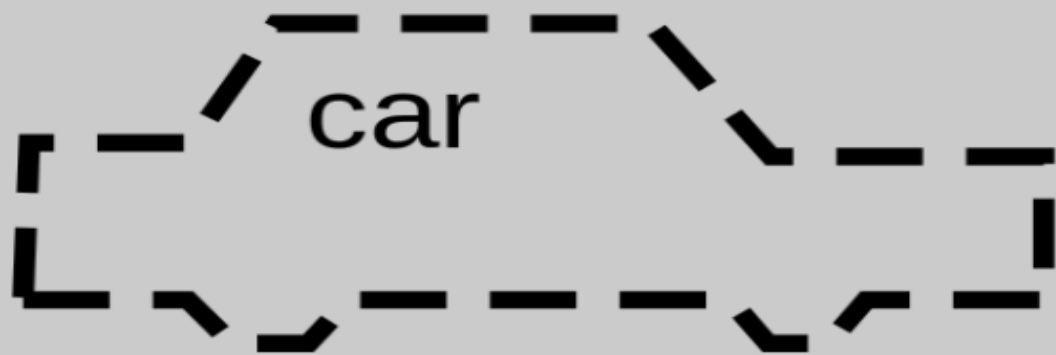
```
data = 10;
```

```
data = 'John';
```


przerwa



class



zabawa (z) klasami

objects



classes in typescript

Inheritance

Polymorphism

Encapsulation

Abstraction

class definition

```
class Book {  
    public author: string;  
    public title: string;  
    public length: number;  
    getFullTitle(): string {  
        return `${this.title} by ${this.author}`;  
    }  
}
```

```
let typeScript = new Book();  
typeScript.title = 'TypeScript';  
typeScript.author = 'Someone';  
typeScript.length = 300;
```

class definition

```
class Book {  
    public author: string;  
    public title: string;  
    public length: number;  
    constructor(author: string, title: string, length:  
        number) {  
        this.author = author;  
        this.title = title;  
        this.length = length;  
    }  
}
```

class definition

```
class Book {  
    constructor(  
        public author: string,  
        public title: string,  
        public length: number) {  
    }  
}
```

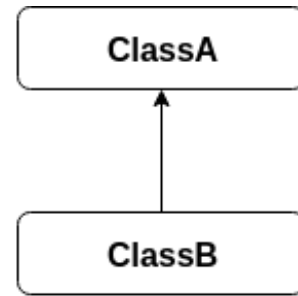

properties

```
class Book {  
    private _title: string;  
  
    get title(): string { return this._title; }  
    set title(value: string) {  
        if (value !== '') {  
            this._title = value;  
        }  
    }  
}
```

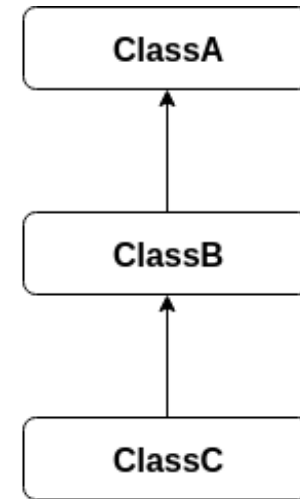
```
let typeScript = new Book();  
typeScript.title = 'TypeScript';
```


inheritance

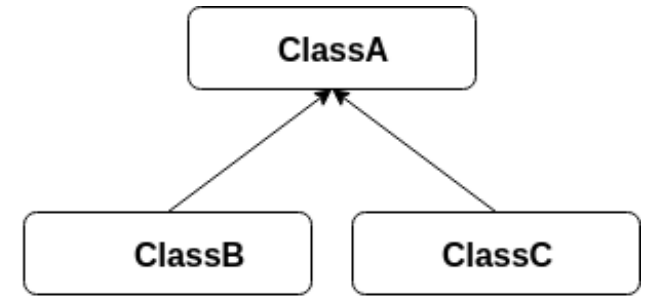
- single
- multilevel
- hierarchical



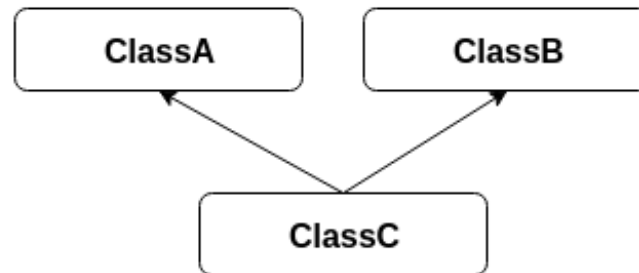
Single Inheritance



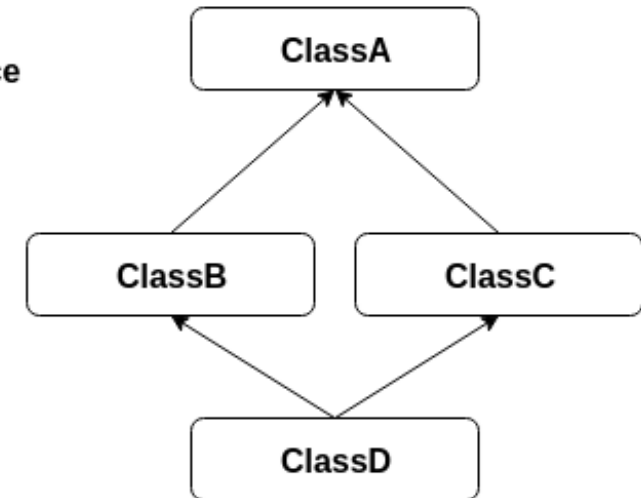
Multilevel Inheritance



Hierarchical Inheritance



Multiple Inheritance



Hybrid Inheritance

przerwa



functions

arrow functions

this keyword

optional and default parameters

can be overloaded

functions

```
function Book(title: string, length?: number) { ...  
}
```

```
function Book(title: string, length: number = 300) {  
...  
}
```

```
function School (name: string, ...id: number[]) { ...  
}
```

function overloads

```
function getCustomer(name: string): string;
function getCustomer(id: number): string;
function getCustomer(property: any): string {
    if (typeof property === 'string') {
        // return customer info based on customer name
    } else if (typeof property === 'number') {
        // return customer info based on customer id
    }
    return 'customer';
};
```

stuff

types unions & intersections

type aliases

interfaces

generics

...



<https://github.com/Banndzior>

#slack

kamil.mijacz@gmail.com

kamil.mijacz@softwarehut.com