TypeScript

WOJCIECH PATYNA

Definicja

- Typowany język programowania stworzony jako nadzbiór języka JavaScript...

Trochę historii

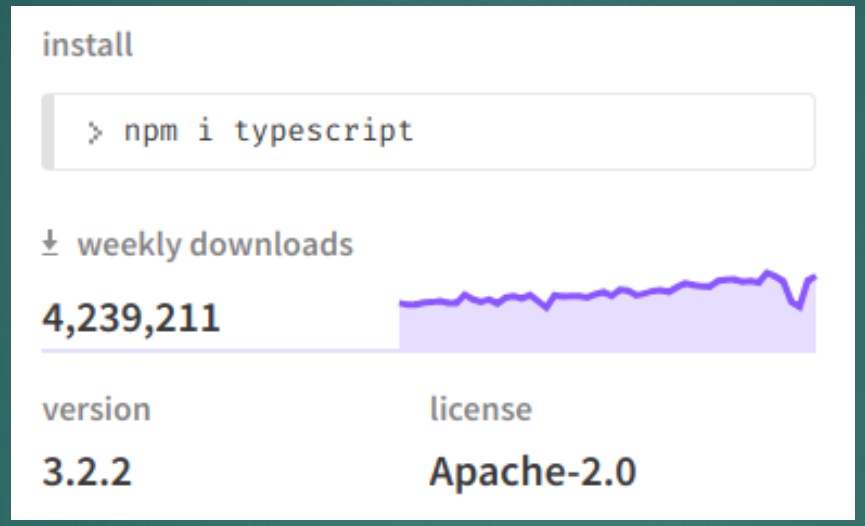
- Pierwsza wersja: Październik 2012 (wersja 0.8)
- Pierwsza stabilna wersja: Styczeń 2016 (wersja 1.0)
- Licencja: Apache License 2.0
- Stworzony i rozwijany: Microsoft

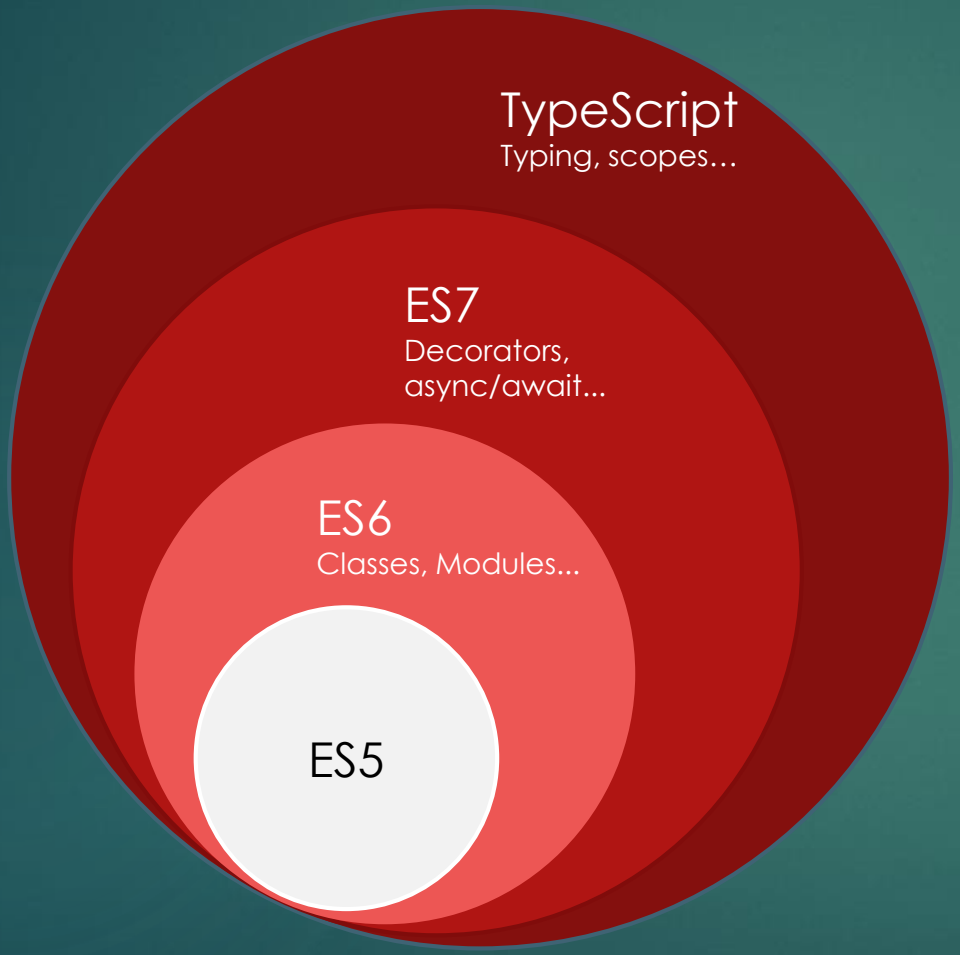
Kompilator Typescript został napisany w TypeScript i następnie przekompilowany do JavaScript

Kto stworzył TypeScript?

Anders Hejlsberg

Teraźniejszość




TypeScript
Typing, scopes…
ES7
Decorators,
async/await...
ES6
Classes, Modules...
ES5

Podstawowe typy

```
let month: number;
 let name: string;
let isVisible: boolean;
 let person: object;
let employee: any;
```

```
let cities: string[];
let customers: Array<any>;
 let value: null;
let value2: undefined;
 let value3: void;
```

```
let address: [string, number, string, string];
```

Podstawowe typy
asercja

```
let someValue: any = "this is a string";
let strLength: number = (someValue as string).length;
 let strLength2: number = (<string>someValue).length;
```

Podstawowe typy
unia

```
let area: string| undefined = property('area');
let element: HTMLElement | null = document.getElementById('passion');
 let menu: Array<Link | Divider> = getMenu();
```

Enums

```
enum WeatherType {
    Sunny = 1,
Cloudy = 2,
Rainy = 3,
Storm = 4,
Frost = 5,
Nightly = 6
}
```

```
enum WeatherType {
   Sunny = 'weather_sunny',
   Cloudy = 'weather_cloudy',
   Rainy = 'weather_rain',
   Storm = 'weather_storm',
   Frost = 'weather_frost',
   Nightly = 'weather_nightly',
}
```

Typy funkcji

```
function add(x: number, y: number): number {
    return x + y;
}
function multiply(x: number, y: number):
 number {
 return x * y;
}
let myAdd: (x: number, y: number) => number =
 add;

type twoOperandsType = (x: number, y: number)
 => number;

let myMultiply: twoOperandsType = multiply;
```

Interfejsy

```
interface TreeLink
   {  type: string;
   title: string;
   url: string;
   isCurrent: boolean;
    preferences: any;
children: TreeLink[];
}
```

```
const treeLink: TreeLink = {
   type: 'external',
    title: 'Google',
   url:
   'http://www.googl
   e.nl',
   isCurrent: false,
    preferences: {
       navIcon:
       'accounts'
   },
   children: []
};
```

Interfejsy

```
interface Flyable { fly: () => void }
interface Quackable { quack: () => void; }

class Duck implements Flyable, Quackable {
    fly() { console.log('Fly, Fly,
   Fly'); }
quack() { console.log('Quack, Quack'); }
}
class Turkey implements Flyable {
fly() { console.log('Ufff!'); }
}
```

Enkapsulacja

```
class Animal {
    private name: string;
    constructor(theName: string) { this.name = theName; }
}

new Animal("Cat").name; // Error: 'name' is private;
```

Klasy abstrakcyjne

```
abstract class Animal {
    constructor(
 public readonly name:
    string
) {}
abstract makeSound(): void;
   abstract move(): void;
   abstract getOffspring(): void;
}
```

```
abstract class Mammal extends Animal{
    constructor(
 public readonly name: string,
    public readonly legs: number
) {
    super(name);
 }
}
```

Typy generyczne

```
function identity<T>(arg: T): T {
    return arg;
}

let myIdentity: <T>(arg: T) => T = identity;

let myIdentity2: <U>(arg: U) => U = identity;
```

Klasy generyczne

```
class Operator<T> {
    initialValue:
   T;
    operation: (x: T, y: T) => T;
}
const numAdder = new
 Operator<number>();
numAdder.initValue = 10;
 numAdder.operation = function(x, y) {
   return this.initValue + x + y;
};
```

```
const strConc = new Operator<string>();
 strConc.initValue = '';
strConc.operation = function (x, y) {
  return `${this.initValue} ${x} ${y}`;
};
```

JS przed TypeScript

TypeScript + React

- npm install webpack –g
- npm install webpack-cli –g
- npm install --save react react-dom @types/react @types/react-dom
- npm install --save-dev typescript awesome-typescript-loader source-map-loader

DEMO

Wady i zalety

Wady

- **Konieczność utrzymywania typów dla bibliotek i pakietów (**https://github.com/DefinitelyTyped/DefinitelyTyped)
- Kompliacja zabiera trochę czasu
- TypeScript nie rozwiązauje nadal istniejących problemów, które JS posiada

Wady i zalety

Zalety

- **Statyczne typowanie**
- **Idealny do dużych aplikacji**
- **Wieloplatformowy**
- **Open Source**
- **Dobra dokumentacja i community**
- **Statystycznie 15% mniej błędów podczas pisania**

Znane firmy używające TypeScript

Alternatywy

ES6

Źródła

- https://www.wakefly.com/blog/what-is-typescript-and-why-should-you-use-it/
- https://www.sitepoint.com/introduction-to-typescript/
- https://basarat.gitbooks.io/typescript/
- http://www.typescriptlang.org/docs/home.html
- https://tutorialzine.com/2016/07/learn-typescript-in-30-minutes
- https://www.slideshare.net/amischol/introduction-to-typescript-122641834?qid=90bf34a5-0bf8-4cd7-aef3-5da74b0aefd9&v=&b=&from_search=34
- https://www.slideshare.net/NexThoughts/introduction-to-typescript-102445619?qid=de272b75-32db-4adb-9125-bb593ebceaf0&v=&b=&from_search=5

Dzięki!