Angular Developer 2

TypeScript

JS devs?

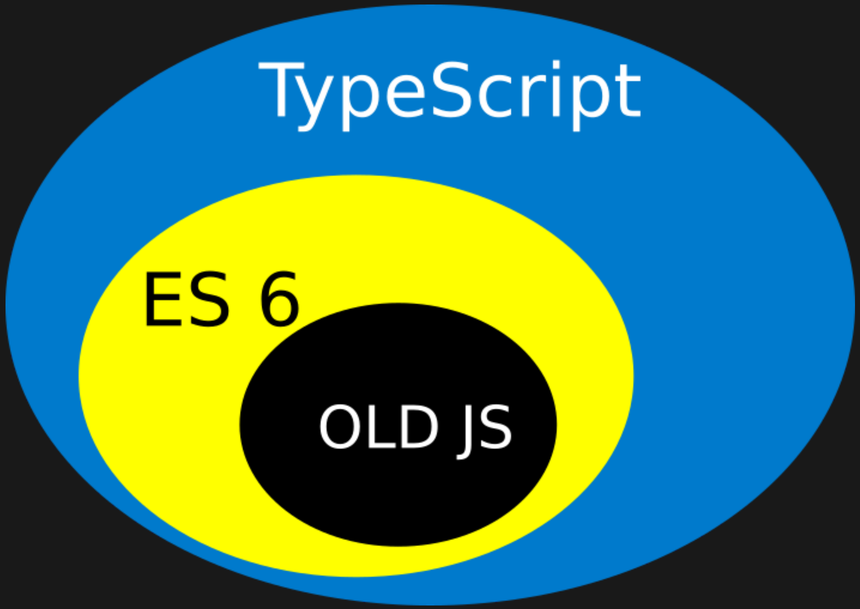
Pessimistically

- leash on our creativity?
- demands more of our attention
- complicates things that are already complex
- another thing we need to learn

TypeScript devs?

What is TypeScript?

- JavaScripts superset
- Modern JS features
- Futuristic JS features
- Scales well


TypeScript
ES 6
OLD JS

'Type' in TypeScript

Static typing? Types checking?

TS ES6 JS

JS

```js
var someNumber = 1;
var someString = 'chrystian';
```

TS ES6

```ts
let someNumber = 1;
const someString = 'chrystian';
```

JS

```
function someFunction() {
  ...
}
var otherFunction = function () { ... }
```

TS ES6

```
const modernFunction = () => {
  ...
}
```

Object Oriented Programming
(classes)

JS ES6 TS

```
var Exam = function Exam(q, a) {
  this.question = q;
  this._correctAnswer = a;
}

const exam = new Exam('Angular or React?', 'ng');
```

ES6

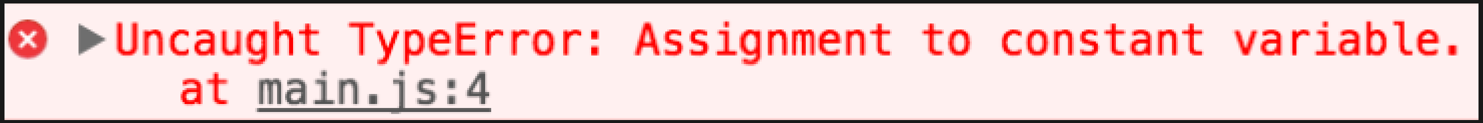
```
class Exam {
  constructor(q, a) {
      this.question = q;
      this._correctAnswer = a;
  }
}

const exam = new Exam('Angular or React?', 'ng');
```

```
class Exam {
  public question;
  private _correctAnswer;

  constructor(q, a) {
      this.question = q;
      this._correctAnswer = a;
  }
}

const exam = new Exam('Angular or React?', 'ng');
```

Static Types?

No types

```
1  const name = 'Chrystian';
2  const points = 0;
3
4  points = 10;
5
6  ...
7
8  points = 'milion';
```



```
Uncaught TypeError: Assignment to constant variable.
    at main.js:4
```

No types

```
const name = 'Chrystian';
let points = 0;

points = 10;

...

points = 'milion';
```

No types

```javascript
const name = 'Chrystian';
let points = 0;

points = 10;

...

points = 'milion';

...
const timePoints = 100;
const totalPoints = timePoints + points;
```

With types

```
const name: string = 'Chrystian';
let points: number = 0;
points = 10;

...
points = 'milion';
```

```
// TypeScript Error
Type 'string' is not assignable to type 'number'.(2322)
```

Basic types (docs)

```
const myName: string = 'Chrystian';

const someNumber: number = 10;

const someFlag: boolean = true;

function asd(): void {
  return 'asd'; // TS will complain
}
```

Try to avoid

```
let badType: any = 'Chrystian';
badType = 1; // no error
```

Array

```
const nums: number[] = [];
const nums2: Array<number> = [];

const names: Array<string> = [];
```

```
...
nums.push(1);
nums.push('one'); // Argument of type 'string'
                  // is not assignable to parameter
                  // of type 'number'
...
```

```
// Options: 'card', 'cash'

let payment: string = 'card';
...
payment = 'kard';
```

Enum

```
enum PaymentMethod {
  CARD,
  CASH,
}
let payment: PaymentMethod;

payment = PaymentMethod.CARD;
```

Union types

```
function print(toPrint: string | string[]) {
  ...
}

print(['chrystian', ' ', 'ruminowicz']);
print('chrystian');
```

Union - many types separated with | (pipe)

More complex types...

(docs)

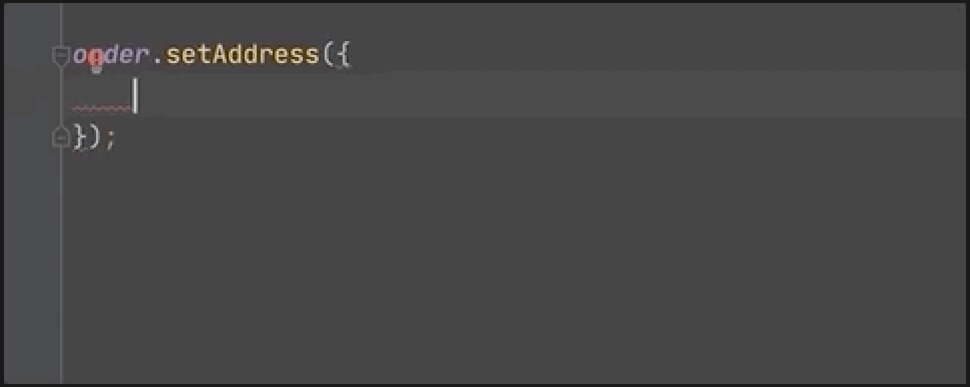
```typescript
class Order {
  private _address;

  setAddress(addr) {
    this._address = addr;
  }
}

const order = new Order();

order.setAddress({
  ulica: 'Saturna'
});
```

```
interface Address {
  streetLine: string;
  streetLine2?: string; // optional property
  postcode: number;
  city: string;
  country: string;
}
class Order {
  private _address;

  setAddress(addr) {
    this._address = addr;
  }
}

```

```typescript
interface Address {
  streetLine: string;
  streetLine2?: string; // optional property
  postcode: number;
  city: string;
  country: string;
}
class Order {
  private _address: Address;

  setAddress(addr: Address) {
    this._address = addr;
  }
}

```

```
interface Address {
  streetLine: string;
  streetLine2?: string; // optional property
  postcode: number;
  city: string;
  country: string;
}
class Order {
  private _address: Address;

  setAddress(addr: Address) {
    this._address = addr;
  }
}

```

Intellisense / Code completetition



```
order.setAddress({
    
});
```

Lets complicate it a bit more

```
class Address {
  streetLine: string;
  streetLine2?: string;
  postcode: number;
  city: string;
  country: string;
}
```

just stick to *interface*

Decorators (docs)

- Functions
- '@' prefixed
- Add additional properties
- Attach metadata

(Metadata - data about data)

What can we decorate?

```
class TbDcrtd { }

class WithMethods {
  tbDcrtd() { }
}

class WithProp {
  tbDcrtd = 1;
}
```

(params and accessors too)

Class decorators

```
@Injectable()
export class GpsService { }

@Component({
  selector: 'app-root',
  templateUrl: 'app.component.html',
  styleUrls: ['app.component.scss']
})
export class AppComponent { }
```

Property / Method decorators

```
@Component({
  selector: 'app-feature',
  templateUrl: 'feature.component.html',
  styleUrls: ['feature.component.scss']
})
export class FeatureComponent {
  @Input() name;

  @HostListener('scroll')
  onScroll() {
    console.log('scrolled');
  }
}
```

SOME ANGULAR DECORATORS

- @NgModule
- @Component
- @Injectable
- @Input
- @Output
- @HostListener

Modules (docs)

TypeScript or ES6, not Angular

```
// some-file.ts
export enum PaymentMethod {
  CARD = 'card',
  CASH = 'cash',
}
```

```
// final-file.ts
import { PaymentMethod } from './some-file';
import { Order } from './order-file';

const order = new Order();
order.setPaymentMethod(PaymentMethod.CARD);
```

Modules are...

...TypeScript and ES6 modules, <u>not</u> Angular

FILES

(not quite)

Modules and Scope

```
let window = 'anything';
```

```javascript
let window = 'anything';

// Uncaught SyntaxError:
// Identifier 'window' has already been declared
```

```
import { something } from '...'

let window = 'anything';
```

```
// import { something } from '...'
export function sum() {}
let window = 'anything';
```

Modules operate in their own scope

any *import* or *export* statement converts to module

require vs import

(we want modules!)

require() - CommonJS

- ES 5
- Node.js

there was (is) RequireJS also

import

- Inspired by CommonJS and RequireJS
- ES 6 standard
- Browsers start to support it

Used by angular (through webpack) to connect all the pieces (build)

main.ts

```
import { enableProdMode } from '@angular/core';
import { platformBrowserDynamic }
  from '@angular/platform-browser-dynamic';

import { AppModule } from './app/app.module';
import { environment } from './environments/environment';

if (environment.production) {
  enableProdMode();
}

platformBrowserDynamic().bootstrapModule(AppModule)
  .catch(err => console.error(err));
```

main.ts ⇽ app/app.module ⇽ app/app.component

Browser compatibility

0%

When will browsers support TypeScript?

NEVER

Wtf, why?

- Types checking - huge performance hit
- All those cool features - no browser support
- Its not really just a language...
- ...its a tool

Transpilation

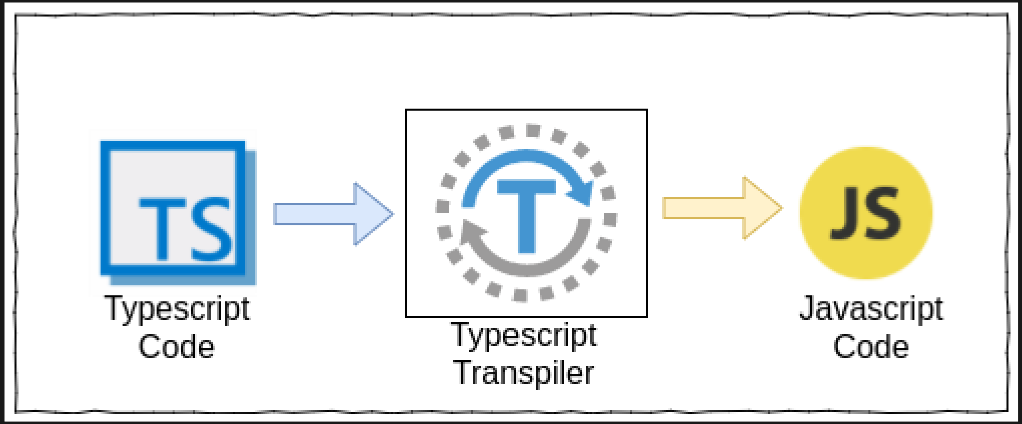


image by intelligenia

TS input

```
var variable1 = 2;
var myName = 'Chrystian';

function add(a, b) {
    return a + b;
}
```

JS output

```
"use strict";
var variable1 = 2;
var myName = 'Chrystian';
function add(a, b) {
    return a + b;
}
```

TS input

```
class SomeClass {
  public field;

  constructor(q: string) {
    this.field = q;
  }
}

const itsInstance = new SomeClass('asd')
```

JS output

```
"use strict";
var SomeClass = /** @class */ (function () {
    function SomeClass(q) {
        this.field = q;
    }
    return SomeClass;
}());

var itsInstance = new SomeClass('asd');
```