TYPESCRIPT

LE JAVASCRIPT STATIQUEMENT TYPÉ

DEVOXX™ France

QUI SUIS-JE ?

- Benoit Lemoine
- Développeur full-stack chez Xebia
- @benoit_lemoine

JAVASCRIPT C'EST BIEN, MAIS...

HISTORIQUE DE TYPESCRIPT

- Made in Microsoft en 2012...
- ... mais open-source et libre (License Apache 2) : https://github.com/Microsoft/TypeScript
- Super-ensemble d'ES5
- Typage statique
- Polyfill pour ES6

LES TYPES - DÉCLARER UN TYPE

```
var name:string = 'Dolan';
var nbLegs:number = 2;

var isMamal = false;

//doesn't compile
//number is not assignable to type boolean
isMamal = 3;
```

DEVOXX™ France

LES TYPES - LES GÉNÉRIQUES

```
var featherColors:Array<string> = ['green', 'red', 'grey'];


var lengthOfClors: Array<number> = featherColors.map(function(colo
 return color.length;
}); // [5, 3, 4]
```

LES TYPES - LES CLASSES

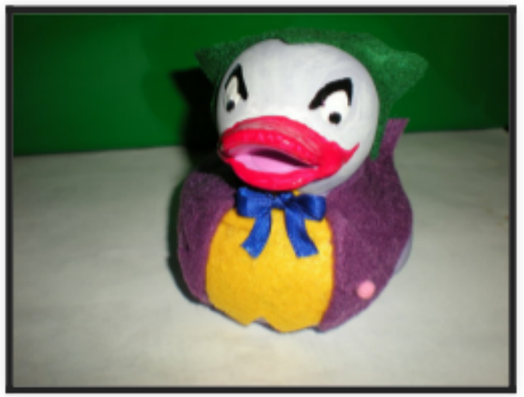
```
class Animal {
  constructor(public name) { }
}

class Duck extends Animal {
  quack() {
    return 'quack';
  }
}

class Platypus extends Animal {}

var dolan:Duck = new Duck('Dolan');
console.log(dolan.quack());
```

LES TYPES - TYPAGE GRADUEL



```
var scrooge = new Duck('Scrooge');
var perry = new Platypus('Perry');
//Doesn't compile Platypus is not assignable to type Duck
scrooge = perry;

//we can assign anything to any
var jokerDuck:any = perry;

//We can assign any to everything
scrooge = jokerDuck;
```

LES TYPES - LES INTERFACES

```
interface Quacker {
  name:string
  quack():string;
}

class Goose extends Animal implements Quacker {
  quack() {
    return 'honk';
  }
}
var daffie:Quacker = new Goose('daffie');
```

LES TYPES - LE TYPAGE STRUCTUREL

```
interface Quacker {
  name:string
  quack():string;
}

var chicken:Quacker = {
  name:'Chicken',
  quack: function() {
    return 'cluck cluck';
  }
};
```

LES TYPES - UNION TYPE

```
var perry = new Platypus('Perry');
var donald = new Duck('Donald');

var animals:Array<Animal> = [perry, donald];
var duckOrPlatypus:Array<Duck|Platypus> = [perry, donald];
```

ECMASCRIPT 2015 - ECMASCRIPT 6

```
function lordify(names:Array<string> = []):Array<string> {
  return names.map(name => `sir ${name} McScrooge`);
}

let noDucks:Array<string> = lordify();
// []

const duckNames = ['Huey', 'Dewey', 'Louie'];

let ducks:Array<Duck> = lordify(duckNames);
// ['sir Huey McScrooge', 'sir Dewey McScrooge',
//  'sir Louie McScrooge']
```

LES MODULES INTERNES

```
//Fichier Animal.ts
module Animal {}

//Fichier Duck.ts
/// <reference path="Animal.ts" />
module Animal {
    var thisVariableCannotBeUsedElesewhere = "test";
    export class Duck {}
}

new Animal.Duck();

import Duck = Animal.Duck;
new Duck();
```

DEVOXX France

LES MODULES EXTERNES AMD OU COMMON JS

```
//Fichier Duck.ts
import Animal = require('Animal');

export class Duck extends Animal {}
```

LES DÉCLARATIONS D'AMBIANCE

```
declare var _;

_.filter([1,2,3], (i) => i%2 === 0);
```

DEFINITELY TYPED

http://definitelytyped.org/

```
/// <reference path="lodash/lodash.d.ts" />

_.filter([1, 2, 3, 4], (i) => i%2 === 0) // [2,4]
_.filter([1, 2, 3, 4], (i:string) => i%2 === 0) //ne compile pas
```

TSD

http://definitelytyped.org/tsd/

- Gestionnaire de dépendances pour les fichiers de définition
- Pas de gestion de version (mais ça arrive...)

EXEMPLE ANGULAR - AVANT

```
angular.module('MyCtrl', ['myService','$scope',
  function(myService, $scope) {

    $scope.myValue = myService.maValue;

    $scope.changeValue = function() {
        $scope.myValue = Math.random();
    };
  }
])

//template
<div ng-controller="MyCtrl">
 {{myValue}}
</div>
```

EXEMPLE ANGULAR - APRÈS

```
class MyCtrl {
    myValue:number;

    static $inject = ['myService'];
    constructor(myService:MyService) {
     this.myValue = myService.maValue;
    }
    changeValue() {
        this.myValue = Math.random();
    }
}
angular.module(MyCtrl.name, MyCtrl);
//template
<div ng-controller="MyCtrl as myCtrl">
 {{myCtrl.myValue}}
</div>
```

TYPESCRIPT 1.5 - RTTI

```
function displayFirstCharacter(word : string) {
  document.body.innerHTML = word.charAt(0);
}

var x : any = 3;
displayFirstCharacter(x);
```

- TypeScript 1.4 : "undefined is not a function" sur `charAt`
- TypeScript 1.5 : "'word' is not a string"

DEVOXX France

TYPESCRIPT 1.5 - ANNOTATIONS

```
function addAnnotation<A>(c: A, annotation: any): A {
    ((<any>c).annotations || ((<any>c).annotations = []))
      .push(annotation);
    return c;
}

function Template(arg: { url: string, directives: any[] }) {
    return c => addAnnotation(c, new TemplateAnnotation(arg));
}

@Template({
  url: '/todo.html',
  directives: [Foreach]
})
class TodoApp {
}
```

TYPESCRIPT 1.5 - ES6

```
var [m, d, y] = [3, 14, 1977];
console.log(d) // 14

for (let word of ["one", "two", "three"])
  console.log(word);  //affiche "one","two","three"

function* idMaker() {
    var index = 0;
    while(true)
        yield index++;
}

var gen = idMaker();

console.log(gen.next().value); // 0
console.log(gen.next().value); // 1
```

TYPESCRIPT 2.0

- ES6 (modules, template String, etc.)
- ES7 (async/await)

LES LIMITES

- Un précompileur de plus
- La vitesse de compilation
- Les fichiers de définition
- Pas compatible avec JSX

CONCLUSION

- Migration Progressive
- Compatibilité avec ES6
- Refactoring simplifié
- Assistance de l'IDE améliorée

DEVOXX™ France

QUESTIONS ?