JavaScript autrement

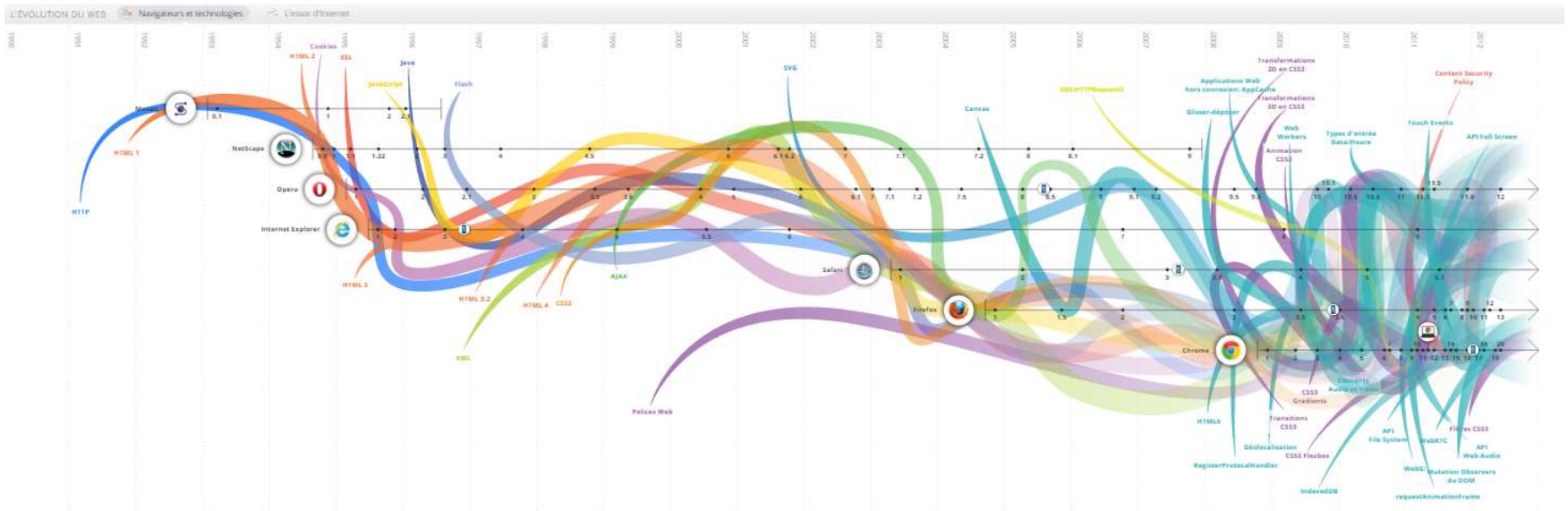
PAR YOUNSS AZZAYANI

Le JavaScript

Un peu d'histoire

- Crée en 10 jours par ***Brendan Eich (@BrendanEich)*** en mai 1995, quand il travaillait pour Netscape
- Son nom original était Mocha
- En septembre on le renomme pour LiveScript
- Vu la popularité de Java, en décembre on le renomme pour JavaScript
- Juin 1997, 1ère spécification ECMA, 2ème spécification en fin d'année
- 3ème spécification en 1999
- 4ème spécification jamais sortie
- 5ème spécification en 2009 (renommage en ES 3.1)
- ES6 est sortie en Juin 2015
- ECMAScript 2016 est sortie en Juin 2016
- ECMAScript 2017 prévue pour 2017 elle contiendra ***async*** et ***Shared memory and atomics***

L’évolution du Web



http://evolutionofweb.appspot.com/?hl=fr

Pourquoi tout cet intérêt

- Enrichissement des pages web
- Validation de données
- Single page App
- Libération des serveurs backend (spécialisation)
- Compréhensible par tout les navigateurs (à l'Exception dE !!!! ;-))
- Flexible, facile, une grande communauté
- Beaucoup de Framework sont bâties autour de JS

Les dangers

- Erreurs de conceptions
- ctrl+c/ctrl+v à partir de StackOverFlow
- Adaptation dans de grand projet
- Code en Mode Read Only
- Code de porc
- Difficilement testable
- Tout le monde pense connaitre JS.

```
$(selector).click(function (e) {
    // Pour que le clic s'arrête
    e.stopPropagation();

    // On part à la recherche de l'image car sa position dépend du template:
    // Pourvu qu'ils ne m'en rajoutent pas
    var image_parente = $(this).find('img');
    if (!image_parente) {
        image_parente = $(this).parent().find('img');
    }
    if (!image_parente) {
        image_parente = $(this).parent().parent().find('img');
    }
    if (!image_parente) {
        image_parente = $(this).parent().parent().parent().parent().find('img');
    }

    // Pour être sûr.
    return false;
});
```

http://code-de-porc.tumblr.com/page/14

Les framework

http://noeticforce.com/best-Javascript-frameworks-for-single-page-modern-web-applications

LE JAVASCRIPT AUTREMENT

TypeScript

- Initialement imaginé et développé par ***Andres Hejlsberg (@ahejlsberg).***, Lead Architect de C#, créateur de Delphi et Turbo Pascal
- Open source, maintenu et développé par le fournisseur d'IE
- Il est orienté à la fois client et serveur: roule sur nodejs
- TypeScript est une surcouche de JavaScript, transcompilé (tsc)
- Les programmes JS sont reconnus dans le code TS
- Développé en 2006
- 1ère sortie publique en 2008 v0.8

Installation

Installation

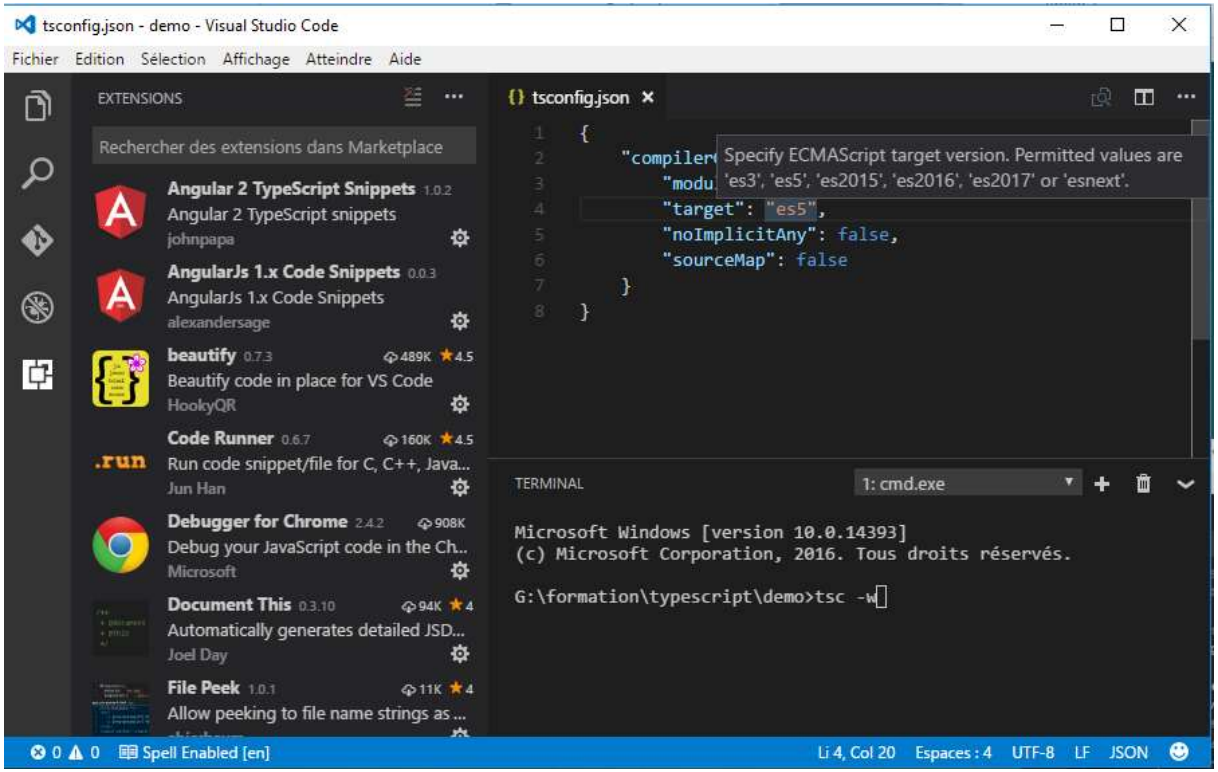
- npm i -g typescript

Compilation

- tsc tp1.ts

Un bon éditeur

http://code.visualstudio.com/



TypeScript vs JavaScript

	JavaScript	TypeScript
Paradigme objet	Prototype	Classe
Typage	Dynamique	Statique et dynamique
Gestion des libraires	Simple	Simple
Debug	Navigateur	Visual Studio/ Plugin dans VS Code, Atom…

1ère Démo

- Création d'un fichier ***tp1.js***
- Écriture d'une fonction qui ***dit bonjour à plusieurs personnes***.
- Rouler la fonction sans paramètres ***node tp1.js***
- Ajouter le tri des noms
- Passer un tableau de noms de personnes
- Rouler la fonction ***node tp1.js***
- Renommer ***tp1.js*** en ***tp1.ts*** ,
- Appeler la fonction sans paramètres et analyser
- Compiler avec ***tsc –w***

Vision

- Évolutif
- Code plus structuré
- Développement modulaire
- Apprentissage rapide pour les gens qui viennent du monde .Net ou Java
- Compile en ECMAScript (code très propre)
- Productif
- Il est en avance par rapport à la spécification ECMAScript
- Facile à tester
- Fonctionnel et Objet
- Visionnaire

Démo

- Tester la fonction de TP1 avec une chaîne de caractère, avec un tableau

Les Types

```
// Boolean
let isValid: boolean = false;

//Number
let age:23;
//String
let fullName: string = "Gad";
let sentence: string = `Bonjour  ${ fullName }.
                        Votre age: ${age}.`;
//Array
let names: string[] = ['Gad','Sara','Cami'];
let ranks: Array<number>= [1,2,3];

//Tuple
let person :[string,number];
person = [fullName,age];
console.log(person[0].substr(1));
person[3]=ranks[0]; // dans ce cas le type doient etre string|number
```

Les Types

```
//Enum
enum RankEnum { FIRST,SECOND, THIRD};
let rank = RankEnum.FIRST;

//Any
let volatile:any;
volatile=1;
volatile=true;
volatile= undefined;
volatile= null;

//Void
function messageBox(message:string): void {
    alert(message);
}

//Null and Undefined
let undf: undefined = undefined;
let nul: null = null;
// tsconfig : --strictNullChecks : null and undefined are only assignable to void and their respective types.
```

Les Types

Never:

- Ce type représente le type de valeurs qui ne se produisent jamais
- C'est le type de retour pour une fonction ou une expression lambda qui lance toujours une exception ou qui ne retourne rien (boucle infinie)

```
//Never

let infinitLoopFct : never = (()=> {while(true){ console.log('hello')}})();

function throwError(msg: string): never {
    throw new Error(msg);
}
```

Type assertions

```
let text: any = "Ceci est un texte";

let strLength: number = (<string>text).length;
let strLength2: number = (text as string).length;
```

Démo

- Spécifier le type du paramètre de la fonction ***ditBonjour()***
- Tester la fonction avec un tableau d'entiers naturels, une chaîne de caractère…
- Remarques: Erreur de compilation.

Les paramètres par défaut

- On peut attribuer des valeurs par défaut aux paramètres d'une fonction
- On utilise le signe **=**

Démo

- Dans ***tp2.ts***, écrire une fonction ***imprimerInformationsPersonnelles()*** qui affiche le nom, prénom et pays de résidence.
- Tester la fonction
- Passer dans la fonction que le nom et le prénom, pas de pays
- Analyser
- Attribuer au paramètre pays la valeur ***CANADA***
- Tester et analyser

Les Templates

- Syntaxe : `${monObjet}`
- Peut accepter une logique (conditions ?:)

Démo

- Dans ***tp3.ts***, écrire une fonction ***imprimerInformationsPersonnellesV2()*** même que ***imprimerInformationsPersonnellesV2()***.
- utiliser une Template de chaîne de caractères
- Tester et analyser

let vs var

- var, dans JS a un scope de fonction
- Question: Est-ce-que ce code est fonctionnel?

```
if(true){
    var logic = true;
}
console.log(logic);
```

- Est ***let***?

Démo

- Dans ***tp4.ts***, essayer d'utiliser ***var*** et ***let*** respectivement, pour mettre à jour le rendu de la section TP4

Les boucles

- Dans JS une boucle peut s'écrire comme suit:

```
function ditBonjour (personnes){
    for(var i = 0; i<personnes.length;i++){
        result = result.concat(' Bonjour ' + personnes.sort()[i] + '<br>');
    }
    console.log(result);
}
```

- Dans type script la syntaxe est : ***for.. of***
  - Pour les index ***for.. in***

Démo

- Dans ***tp5.ts***, récrire la fonction de tp1.ts en utilisant la boucle for de TS et les template String.

Les expressions Lambda

- Une expression lambda est une fonction anonyme
- Elle est utilisé au sein d'un bloc de code pour manipuler les données, comme filtrer, trier, réduire, vérifier…
- Elle a l'avantage de manipuler directement les variables du scope de son bloc contenant.
- Exemple de lambda (tp6)

```
function ditBonjourV3(personnes: Array<String>){
    for(let personne of personnes.sort((p1,p2)=>{ return p1.length<p2.length?1:-1})){
        result = result.concat(`Bonjour ${personne}<br>`);
    }
    console.log(result);
}
ditBonjourV3(["Oliver", "Karl", "Simon"]);
```

Démo : Les expressions Lambda

- Dans tp7.ts on un programme qui incrémente un output suite à un click sur un bouton ***submit***
- Le code

```
var submitElm  = document.getElementById("submit");
var counterElm  = document.getElementById("count");
function Counter(counter,submitter){
    this.count = 0;
    counter.innerHTML = this.count;
    submitter.addEventListener( 'click', function(){

        this.count += 1;
        counter.innerHTML= this.count;
    });

}
new Counter(counterElm,submitElm);
```

- Le code ne fonctionne pas: Pourquoi ?
  - Pour résoudre l'erreur on va utiliser ***_this***
  - On peut aussi utiliser les expressions lambda sans faire appel à ***_this***.

Démo : Les expressions Lambda

- Solution 1: utiliser ***_this***
  - tester
- Solution 2: utiliser une expression lambda
  - tester

Les varargs

tp5.ts | tp6.ts | tp7.ts | index.html | tp8.ts

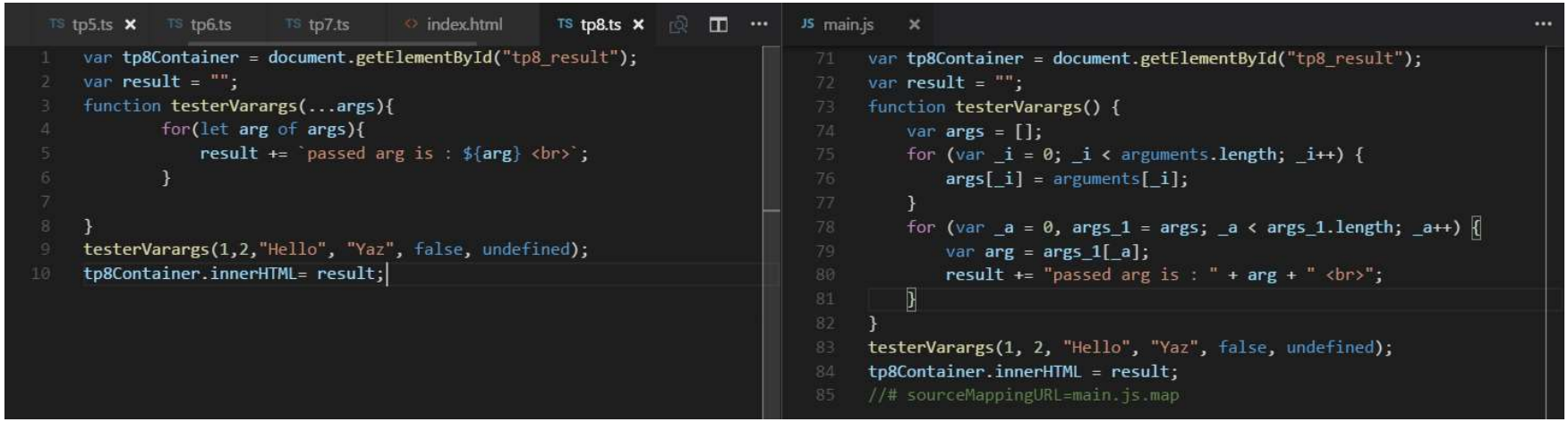
```ts
var tp8Container = document.getElementById("tp8_result");
var result = "";
function testerVarargs(...args){
        for(let arg of args){
            result += `passed arg is : ${arg} <br>`;
        }

}
testerVarargs(1,2,"Hello", "Yaz", false, undefined);
tp8Container.innerHTML= result;
```

main.js

```js
var tp8Container = document.getElementById("tp8_result");
var result = "";
function testerVarargs() {
    var args = [];
    for (var _i = 0; _i < arguments.length; _i++) {
        args[_i] = arguments[_i];
    }
    for (var _a = 0, args_1 = args; _a < args_1.length; _a++) {
        var arg = args_1[_a];
        result += "passed arg is : " + arg + " <br>";
    }
}
testerVarargs(1, 2, "Hello", "Yaz", false, undefined);
tp8Container.innerHTML = result;
//# sourceMappingURL=main.js.map
```

Déclaration de variables dynamiques



tp5.ts | tp6.ts | tp7.ts | index.html | tp8.ts

```ts
var tp8Container = document.getElementById("tp8_result");
var result = "";
function testerVarargs(...args){
        for(let arg of args){
            result += `passed arg is : ${arg} <br>`;
        }

}
testerVarargs(1,2,"Hello", "Yaz", false, undefined);
tp8Container.innerHTML= result;
```

main.js

```js
var tp8Container = document.getElementById("tp8_result");
var result = "";
function testerVarargs() {
    var args = [];
    for (var _i = 0; _i < arguments.length; _i++) {
        args[_i] = arguments[_i];
    }
    for (var _a = 0, args_1 = args; _a < args_1.length; _a++) {
        var arg = args_1[_a];
        result += "passed arg is : " + arg + " <br>";
    }
}
testerVarargs(1, 2, "Hello", "Yaz", false, undefined);
tp8Container.innerHTML = result;
//# sourceMappingURL=main.js.map
```

Surcharge de méthodes

```typescript
function addAtTheEnd( src: string, expr: string):string;
function addAtTheEnd( src: any[], expr: string):any[];
function addAtTheEnd( src: (string |any[]), expr: string):(string |any[]
   if( typeof(src) === 'string' || src instanceof String){

         return src+ expr;
    }
    if( src instanceof Array){

        src.push(expr);
         return src;
    }

}

console.log(
console.log(
addAtTheEnd()
```

```
1/2 addAtTheEnd(src: string, expr: string): string
```

```typescript
    }
}
testVarargs(1, 2, "hello", "boy", false, undefined);
var tablePrefix = "tlb_";
var dataModel = (_a = {},
    _a[tablePrefix + "PERSON"] = "{id:number, firstName:string}",
    _a[tablePrefix + "CREDIT_SCORE"] = "{score:number, scoreDate: date,
    _a);
console.log(dataModel);
var _a;
function addAtTheEnd(src, expr) {
    if (typeof (src) === 'string' || src instanceof String) {
        return src + expr;
    }
    if (src instanceof Array) {
        src.push(expr);
        return src;
    }
}
console.log(addAtTheEnd("Bob", "o"));
console.log(addAtTheEnd(["D", "o", "d"], "o"));
```

Possibilités

- Interfaces
- Enum
- Classes
- Héritage
- Static
- Constructeur
- fonctionnel

…

Démo

- Voir vtp10.ts

Décorateurs

index.html | tp8.ts | tp9.ts | vtp10.ts | vtp11.ts | tp2.ts typescript\demo

```typescript
function logger(log:string) {
   return function (target: any, propertyKey: string, descriptor: PropertyDescriptor) {
        result+= `LOG-- ${new Date()}: ${log} `;
    }
}
class Main {

    constructor(){}
   @logger(" Main.main called ")
    static main(args?:Array<any>):void {
        result+= '<br>Bonjour tout le mode<br>';
    }
}
Main.main();
```

main.js

```javascript
function logger(log) {
    return function (target, propertyKey, descriptor) {
        result += "LOG-- " + new Date() + ": " + log + " ";
    };
}
var Main = (function () {
    function Main() {
    }
    Main.main = function (args) {
        result += '<br>Bonjour tout le mode<br>';
    };
    return Main;
}());
__decorate([
    logger(" Main.main called ")
], Main, "main", null);
Main.main();
```

Support de JS dans TS

POUR LES NOSTALGIQUES

Démo

- Dans ***tsconfig.json*** ajouter ***allowJS***
- Dans la fonction JS, commenter la méthode Party on utilisant ***jsd***

```
/**
 * Create a new Party Object
 * @constructor
 * @param {string} firstName first name
 * @param {string} lastName last name
 * @param {Date} dateOfBirt date of birth
 *
 */

function Party(firstName, lastName, dateOfBirth){
    this.firstName = firstName;
    this.lastName = lastName;
    this.dateOfBirth = dateOfBirth;
}


var policyHolder = new Party("yaz","boy", new Date(1983,7,30));
```

```
var tp12Container = document.getElementById("tp12_result");
var result = "";
let party = new Party("Jonas Moore","Rose", new Date(1970,10,10));
result = `tester js dans ts: Bonjour ${party.firstName}`;
//tp12Container.innerHTML= result;
```

TypeScript & Angular 1

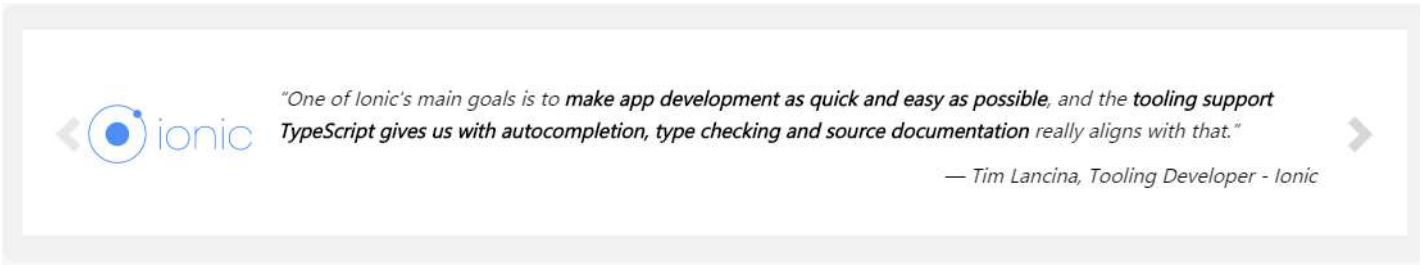
Démo

Amis de TypeScript

Qu’est ce qu’on dit à propos

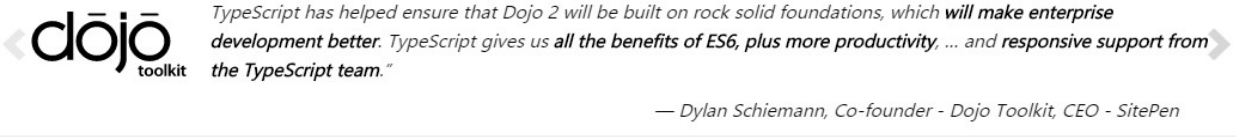
*We love TypeScript for many things... With TypeScript, several of our team members have said things like '**I now actually understand most of our own code!**' because they can easily traverse it and understand relationships much better. And we've **found several bugs via TypeScript's checks**."*

— Brad Green, Engineering Director - AngularJS



*"One of Ionic's main goals is to **make app development as quick and easy as possible**, and the **tooling support TypeScript gives us with autocompletion, type checking and source documentation** really aligns with that."*

— Tim Lancina, Tooling Developer - Ionic



*TypeScript has helped ensure that Dojo 2 will be built on rock solid foundations, which **will make enterprise development better**. TypeScript gives us **all the benefits of ES6, plus more productivity**, ... and **responsive support from the TypeScript team**."*

— Dylan Schiemann, Co-founder - Dojo Toolkit, CEO - SitePen

https://www.typescriptlang.org/

If you don’t use it, you loose it

LES BROWN

Questions