Una introducción a TypeScript

GH Gustavo Hernán Dohara

¿Qué es Typescript?

Typescript es *Javascript* con esteroides, dicho de otra forma, es una versión mejorada de *Javascript* con funciones y características que no tiene *Javascript*.

Además, Typescript es fuertemente tipado a diferencia de *Javascript,* en donde vos podés definir una variable y luego cambiarle el tipo asignándole otra variable de diferente tipo, en Typescript no se puede hacer eso.

También, todo código escrito en Typescript puede convertirse a *Javascript,* lo cuál es recontra útil (por un tema de compatibilidad). Y, por último, es de Código Abierto, lo que quiere decir que cualquiera puede ver el código y modificarlo a su antojo.

Características (más importantes)

Es tipado :)

Tal y como su nombre lo indica, TypeScript (de hecho su nombre sale de esta características). Con Typescript se pueden definir Tipos de datos propios. Vos dirás, pero en *Javascript* también me puedo crear tipos, puedo hacer:

```
var persona = {nombre: 'Gustavo'}
```

y luego usarlo de la siguiente forma

```
console.log(persona.nombre);
```

Y yo te digo… y sí, pero el tipo persona en realidad es un Tipo genérico (Object), de hecho, si vos hacés lo siguiente:

```
console.log(persona.apellido)
```

el código no se te va a romper, pero, pero, peeeeero, apellido en realidad no existe, no está definido, y lo más probable es que metas un bug en tu código, por esta característica flexible de *Javascript*.

En cambio, si hacés lo mismo en Typescript:

```
var persona = {nombre: 'Gustavo'};
console.log(persona.nombre);
console.log(persona.apellido); //acá el código se rompe
```

Tu código se rompe y se rompe en tiempo de compilación, o sea, no podés ejecutarlo y vas a tener que arreglarlo. Esto es bastante útil para evitar meter bugs.

Por otro lado, si tenés una IDE copada, hasta te puede auto-completar los campos de los tipos que definas ;)

Ah, por cierto, la línea:

```
var persona = {nombre: 'Gustavo'};
```

lo que hace es crear un tipo con la propiedad nombre y asignárselo automáticamente a la variable persona, esta características se llama Type Inference, o sea, asignarle un tipo a una variable cuando se la inicializa. Esto es de mucha utilidad, ya que sería un garronazo definir el tipo a variable por variable.

Type Inference

se llama Type Inference (Inferencia del Tipo) al mecanismo de asignarle a una variable su Tipo cuando se está inicializando. El Tipo será el del valor que se le está asignando a la variable.

Y, ya que estamos con definiciones, en la línea:

```
console.log(persona.apellido);
```

la propiedad de no dejarte usar propiedades o métodos que no estén definidos se llama Type Safety.

“

Type Safety:

se llama Type Safety al mecanismo que valida el uso de atributos y métodos definidos en un Tipo de dato

Y vos podrías preguntar, che.. ¿ y si quiero tener una variable que permita dos tipos distintos? Porque en *Javascript* ¡eso se podía!

Bueno, ¡en Typescript también se puede! Pero a su manera.

En *Javascript* vos podías hacer:

```
var mensaje = 'mi mensaje';
mensaje = 1;
```

Definimos un mensaje le asignamos el valor ‘mi mensaje’ que es del Tipo *string,* y luego le asignamos el valor 1 (tipo *número*)

Ahora bien, ¿cómo hacemos lo mismo en Typescript?, Muy fácil, usamos el operador **« | »** de la siguiente manera:

```
var mensaje:string | number = 'mi mensaje';
mensaje = 1;
```

para definir un tipo en Typescript, hacemos lo siguiente:

```
var mensaje:string
```

Y, si queremos definir más de un tipo para una variable, podemos usar el operador union type «|»

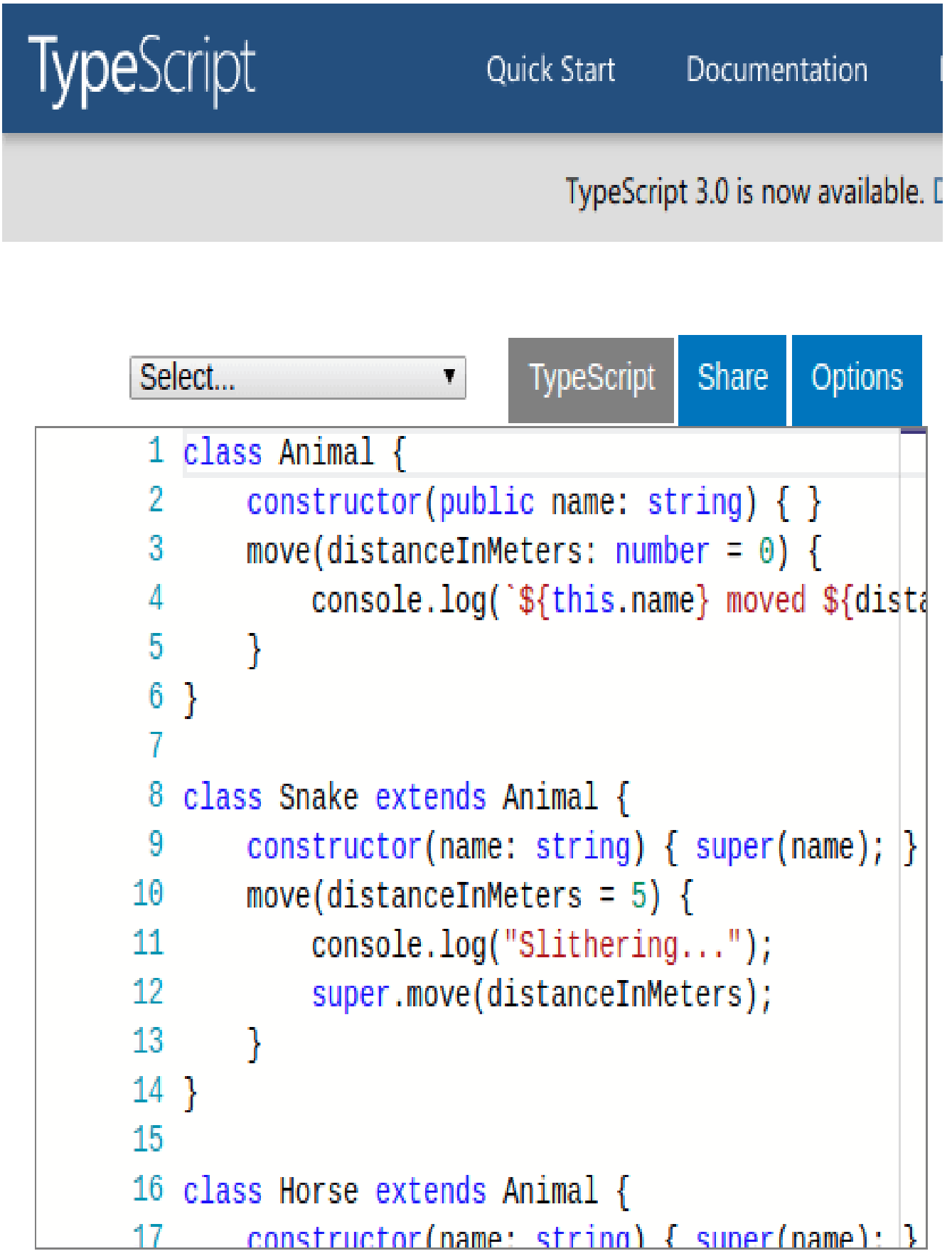
```
var mensaje:string|number
```

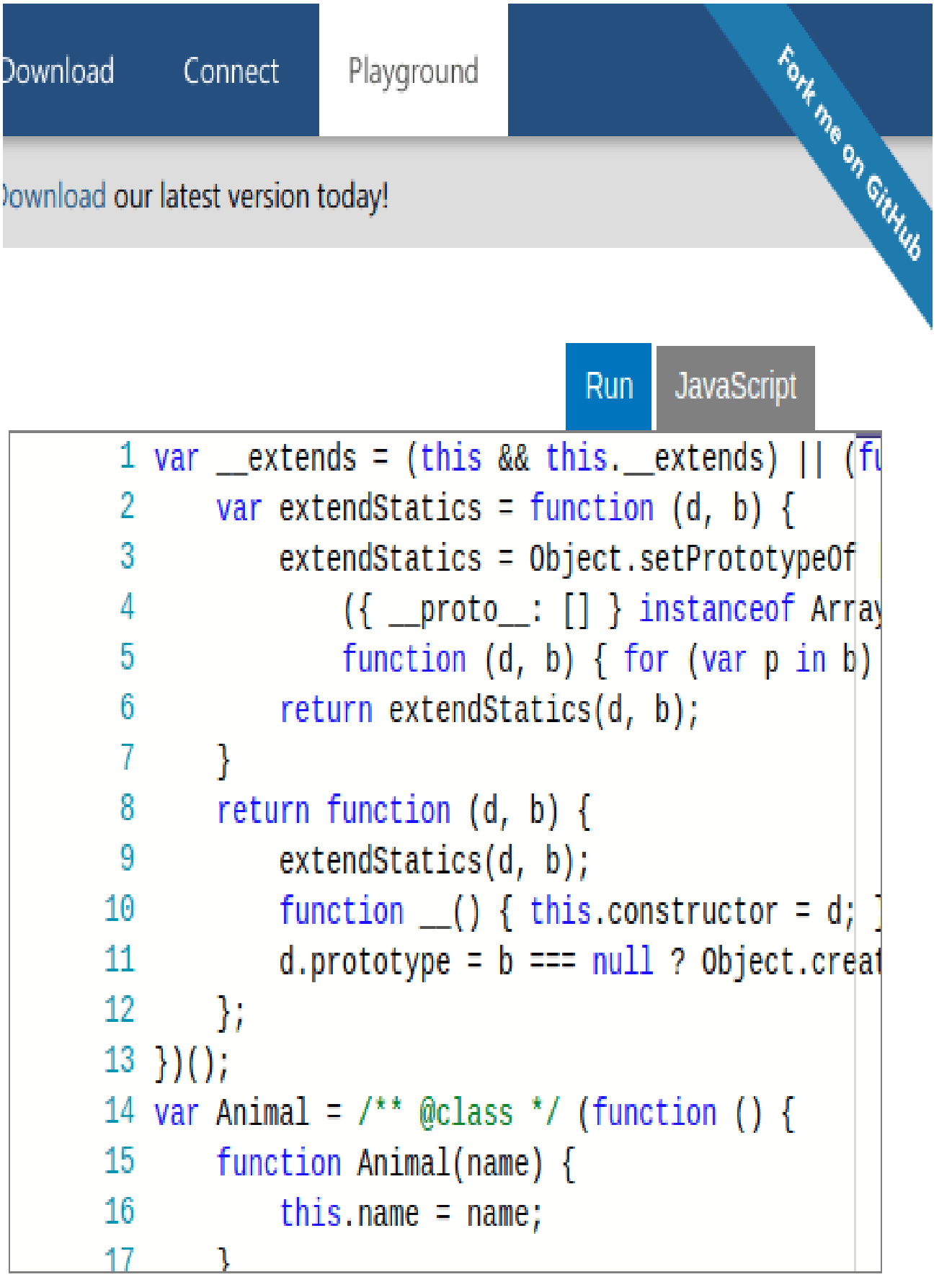
Compila a Javascript

Como comenté al inicio, otra de las características de Typescript es que, el código que escribamos podemos convertirlo (transpiling) a *Javascript* sin problemas. Gracias a ésto, no vas a tener que preocuparte si los browsers mas viejos soportan o no versiones de *Javascript* mas modernos, vos convertís tu código a lo que tu browser necesite. Incluso, podemos convertirlo a diferentes versiones de *Javascript*, ésto permite escribir nuestro código una vez e ir convirtiendo a diferentes versiones de *Javascript* según tu conveniencia.

Transpiling: convertir el código fuente de un lenguaje a otro lenguaje

Hay un playground muy copado en donde se pude ver cómo funciona Typescript y qué termina transpilando:


TypeScript
Quick Start
Documentation
TypeScript 3.0 is now available.
Select...
TypeScript
Share
Options
1 class Animal {
2 constructor(public name: string) { }
3 move(distanceInMeters: number = 0) {
4 console.log(`${this.name} moved ${dista
5 }
6 }
7
8 class Snake extends Animal {
9 constructor(name: string) { super(name); }
10 move(distanceInMeters = 5) {
11 console.log("Slithering...");
12 super.move(distanceInMeters);
13 }
14 }
15
16 class Horse extends Animal {
17 constructor(name: string) { super(name); }


Download
Connect
Playground
Fork me on GitHub
Download our latest version today!
Run
JavaScript
1 var __extends = (this && this.__extends) || (fu
2 var extendStatics = function (d, b) {
3 extendStatics = Object.setPrototypeOf
4 ({ __proto__: [] } instanceof Array
5 function (d, b) { for (var p in b)
6 return extendStatics(d, b);
7 }
8 return function (d, b) {
9 extendStatics(d, b);
10 function __() { this.constructor = d;
11 d.prototype = b === null ? Object.creat
12 };
13 })();
14 var Animal = /** @class */ (function () {
15 function Animal(name) {
16 this.name = name;
17 }

https://www.typescriptlang.org/play

Escribílo una vez, ejecutálo en cualquier lado

Esto viene de la mano del punto anterior, que decía que podés convertir código Typescript a cualquier versión de *Javascript*. Gracias a esto, no sólo vas a poder adaptar tu código a cualquier browser, ya sea, que soporte versiones modernas de *Javascript* o no; también vas a poder escribir tu código para ejecutarlo en tu servidor (por ejemplo Node js) y convertirlo a diferentes versiones de

Javascript, dependiendo de qué versión de Nodejs tengas en tu servidor.

Por todo ésto, escribí tu código una sola vez en Typescript, convertílo a la versión de *Javascript* que te convenga y ejecutálo en cualquier lado ;).

Y… ¿por qué tengo que aprenderlo?

Si todo lo que te dije antes no te alcanzó para comenzar a aprender Typescript (¡ojo! me faltan decirte más cosas, pero para este post es suficiente) tal vez lo próximo que te diga te convenza.

- ¿Te gusta Angular?

Si tu respuesta es ¡SÍ!

- ¿Te gustaría aprender la última versión de Angular (ahora van por la versión 6)?

Si tu respuesta es ¡SÍ!

Bueno… si fue un doble SÍ, entonces tenés que saber que la última versión de Angular tomó como lenguaje de programación estándar Typescript.

Ooooohhhhhh…

Conclusión

Así que ya sabés, no todo es *Javascript* en la vida, y si te cansó que *Javascript* sea tan raro :P, a Typescript no lo veas tan raro. Por algo los muchachos de Angular lo adoptaron como su lenguaje estándar. No veas a Typescript como una molestia, tiene sus reglas nuevas, pero, si le agarrás la mano, te va a ayudar a hacer tu código mas prolijo y legible. ¡Y a perder menos tiempo buscando bugs raros! Ánimo y… ¡A aprender Typescript!

@

@