La magia de TypeScript

Me presento

Pepe Fabra Valverde

Senior Software Engineer Developer Advocate

C&CA CLOUD & CUSTOM APPLICATIONS

SDO

Parte de C&CA

VLC


TS

Qué es TypeScript

Qué es TypeScript

Sistemas de tipados

Contexto y propósito

Contexto: 2012

Propósito: Ofrecer un sistema de tipados integral

> “Una herramienta que ayudase a crear aplicaciones JavaScript que escalasen”

Ecosistema

CoffeeScript (2009), Flow (2014)

Hindley-Milner

Sistema de tipados con inferencias para cálculo lambda

Modelo mental

Cómo enfocar TypeScript

TypeScript nos ha de ayudar como desarrolladores

¿Cuándo queremos el error?

- devtime → IDE
- buildtime (compiletime) → CI/CD
- Runtime → Usuarios

Introducción

Introducción

Tipado

En JavaScript Vanilla

Primitivos

typeof

string

number

BigInt

Existe… pero no es muy usado

undefined

boolean

Objetos

Objetos

{ K: V }

keyof

[k: type] vs [k in type]

Object[keyof Object]

Funciones

method

function(params: any): TReturn

lambda

(props: any) => TReturn

Opcionales y defaults

Opcionales y defaults

- Booleanos siempre a false

“?”, “| undefined” y “=”

Introducción

Type

Type

- constante
- herencia múltiple con unión
- Extender pero no modificar
- NoKV (Not only Key Value)

Introducción

Interface

Interface

- Extender y modificar
- Más parecido a var
- Semántica más similar
- Key Value

Introducción

Type vs Interface

Type vs Interface

¿Cuándo usamos uno o el otro?

Type vs Interface

- const
  - Sólo puede ser declarado una vez
- var
  - Puede declararse muchas veces

Type vs Interface

- const
  - Sólo puede ser declarado una vez
- Herencia con “|” o “&”

- var
  - Puede declararse muchas veces
- Herencia con extends y redeclaración

Type vs Interface

- const
  - Sólo puede ser declarado una vez
- Herencia con “|” o “&”
- Declaración como variable

- var
  - Puede declararse muchas veces
- Herencia con extends y redeclaración
- Declaración como clase

Type vs Interface

- const
  - Sólo puede ser declarado una vez
- Herencia con “|” o “&”
- Declaración como variable
- Mejor para aplicaciones

- var
  - Puede declararse muchas veces
- Herencia con extends y redeclaración
- Declaración como clase
- Mejor para librerías

Type vs Interface

- const
  - Sólo puede ser declarado una vez
- Herencia con “|” o “&”
- Declaración como variable
- Mejor para aplicaciones
- Herencia múltiple con union

- var
  - Puede declararse muchas veces
- Herencia con extends y redeclaración
- Declaración como clase
- Mejor para librerías
- Herencia múltiple con extends

Introducción

Enums

Los enums de TypeScript son una 

- Michigan typescript

Enums en TypeScript

El único elemento de TypeScript que se mantiene en la transpilación

Enums bien hechos

as const

Target time: 5m

Demo

Experimentar con los diferentes enums

- Prueba a mostrarlos en consola

[Conceptos]
Los ladrillos

[Conceptos] Los ladrillos

Tipos especiales

void

never

Prohibir argumentos

{}

any

unknown

const

[Conceptos] Los ladrillos

Keywords

as

as Type vs variable: Type

as

as Type vs variable: Type

Coerción vs Restricción

using

Resuelve ambigüedades

extends

Constraints, **más adelante**

[Conceptos] Los ladrillos

Operaciones

! - bang

“Fíate de mí”

& - union

Polimorfismo y mixin

| - discriminator/intersection

Ambigüedad

Tipos ambiguos

string | number

¿está arriba o abajo?

Props específicas

Option

{ success: true, data: TData }

{ success: false, error: TError }

[Conceptos]
Los andamios

[Conceptos] Los andamios

Literales

string as const

interpolación

Igual que en Vanilla

prefijos y sufijos

[Conceptos] Los andamios

Comparación

Filosofía

De genérico a específico

Typescript funciona comprando de lo más genérico a lo más específico, no al revés.

El mínimo de especificidad se lo pones a la izquierda, si a la izquierda le pones algo muy específico ya no le va a gustar

Puede hacerse pasar por

El tipo de la derecha ¿dará error al hacerse pasar por el de la izquierda?

De más genérico a específico (moderadamente)

De más específico a genérico (suficientemente específico)

Serán ejemplos colaborativos, se agradece la participación

Puede hacerse pasar por

De más genérico a específico (moderadamente)

string

string

De más específico a genérico (suficientemente específico)

“literal”

Puede hacerse pasar por

De más genérico a específico (moderadamente)

✓ string

string

✗ “literal”

De más específico a genérico (suficientemente específico)

✓ “literal”

string

Puede hacerse pasar por

De más genérico a específico (moderadamente)

✓ string

✓ string

“literal”

✗ “{number}-ex”

De más específico a genérico (suficientemente específico)

✓ “literal”

✓ string

string

Puede hacerse pasar por

De más genérico a específico (moderadamente)

✓ string

✓ string

✗ “literal”

“{number}-ex”

✗ number

De más específico a genérico (suficientemente específico)

✓ “literal”

✓ string

✗ string

“tt-ex”

Puede hacerse pasar por

De más genérico a específico (moderadamente)

✓ string

✓ string

× “literal”

× “{number}-ex”

number

✓ {}

De más específico a genérico (suficientemente específico)

✓ “literal”

✓ string

× string

× “tt-ex”

“0”

Puede hacerse pasar por

De más genérico a específico (moderadamente)

✓ string

✓ string

× “literal”

× “{number}-ex”

× number

{}

× unknown

De más específico a genérico (suficientemente específico)

✓ “literal”

✓ string

× string

× “tt-ex”

× “0”

{ foo: “bar” }

Puede hacerse pasar por

De más genérico a específico (moderadamente)

✓ string

✓ string

× “literal”

× “{number}-ex”

× number

✓ {}

unknown

De más específico a genérico (suficientemente específico)

✓ “literal”

✓ string

× string

× “tt-ex”

× “0”

✓ { foo: “bar” }

undefined

Puede hacerse pasar por

De más genérico a específico (moderadamente)

- ✓ string
- ✓ string
- × “literal”
- × “{number}-ex”
- × number
- ✓ {}
- × unknown

De más específico a genérico (suficientemente específico)

- ✓ “literal”
- ✓ string
- × string
- × “tt-ex”
- × “0”
- ✓ { foo: “bar” }
- × undefined

¿Alguna duda?

- ✓ string
- ✓ string
- ✗ “literal”
- ✗ “{number}-ex”
- ✗ number
- ✓ {}
- ✗ unknown

- ✓ “literal”
- ✓ string
- ✗ string
- ✗ “tt-ex”
- ✗ “0”
- ✓ { foo: “bar” }
- ✗ undefined

[Conceptos] Los andamios

Utility Types

Pick<T extends Object>

Omit<T extends Object>

Exclude<T extends Object>

Reduce ambigüedad

ReturnType<T extends Function>

Parameters<T extends Function>

…y muchos más

[Conceptos] Los andamios

Array

Conjunto

string[]

Tupla

[string, string]

Array<T extends []>

Con utility type

¿cuál usamos?

[Conceptos] Los andamios

tsconfig.json

strict

JavaScript y los errores

Total TypeScript

https://www.totaltypescript.com/tsconfig-cheat-sheet

[Conceptos] Los andamios

Module declarations

JavaScript tipado

Pero sigue siendo JavaScript

.d.ts

vite-env.d.ts

.mts

.cts

Target time: 5m

Demo

Cómo extender un tipo de una librería, readaptarlo o hacerlo más abierto

Checkpoint

Checkpoint

¿Dudas?

[Conceptos]
El baño

[Conceptos] El baño

Genéricos

Filosofía de Genéricos

- Son nuestras variables, hiperparámetros de lambda
- Tipos, funciones, clases pueden recebir genéricos

extends

Crean constraints

herencia

Es flexible

defaults

¡Cuidado! pueden romper la herencia

Target time: 10m

Demo

GenericBags

Cómo se crean y utilizan

Cuándo usarlo

[Conceptos] El baño

Inferencias

Concepto

No explicitar lo implícito

Tipos de inferencia

- Directa, dices lo que es (no hay genéricos)
- Implícita, identifica a qué haces referencia
- Parcial/complementaria, especificas los genéricos

Tipos de inferencia

- Directa, dices lo que es (no hay genéricos)
- Implícita, identifica a qué haces referencia
- Parcial/complementaria, especificas los genéricos

Tipos de inferencia

- Directa, dices lo que es (no hay genéricos)
- Implícita, identifica a qué haces referencia
- Parcial/complementaria, especificas los genéricos

infer

A medio camino entre extends y un genérico extra

Ternarios y recursividad

Tipados complejos y adaptativos

[Conceptos] El baño

i18n type-safe

DEMO: Live code

https://github.com/jofaval/typescript-examples/blob/master/utilities/nested-object-from-scratch.utilities.ts

El objetivo es usar TypeScript (libland) para tener que usarlo (userland) lo menos posible

[Conceptos]
La tienda de campaña

@use JSDoc

[Conceptos] La tienda de campaña

Propósito

Por qué usar JSDoc si tenemos TypeScript

Contexto

Por qué

[Conceptos] La tienda de campaña

Sintáxis

```
/**
 *
 */
```

[Conceptos] La tienda de campaña

Documentación

@author

@source

@returns

@var

@param

[Conceptos] El baño

TSDoc

¿Y esto funciona en TypeScript?

TSDoc

- https://tsdoc.org/
- https://github.com/microsoft/tsdoc

[Conceptos] La tienda de campaña

Conceptos avanzados

Referencias a otros tipos

@link @see

Genéricos

@template

Inferencias

Con TypeScript en vscode

TypeScript por debajo

A nivel de IDE/LSP

[Conceptos] La tienda de campaña

Buildtime

Buildtime

El principal beneficio de esta práctica

DHH


Elon Musk by Walter Isaacson
#1 New York Times Bestseller
Corporate needs you to find the differences between this picture and this picture.


They're the same picture.

[Conceptos] La tienda de campaña > Buildtime

Librerías

Svelte

- https://github.com/sveltejs/svelte

Turbo

- https://github.com/hotwired/turbo

Target time: 10m

Demo

- Cómo funciona JSDoc, e integración con VSCode
- Cuánto de efectivo es
- ¿Es nativo? ¿Lo soportarán los navegadores?

Cierre

Cierre

Ecosistema

Comunidad

Muy buena adopción y comunidad

Cierre > Ecosistema

Librerías

Librerías

- ts-toolbelt
- zod

ts-toolbelt

El batcinturón

Zod… y yup

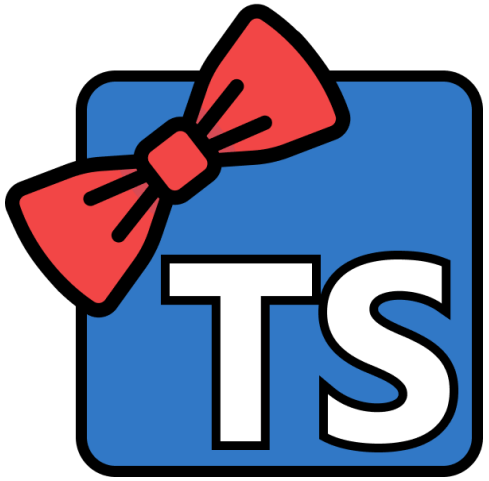
Validar no había sido tan cómodo

Cierre > Ecosistema

Extensiones

Pretty TypeScript Errors

- Los errores de TypeScript no son los más fáciles de leer
- No sólo simplifica errores de TypeScript, sino todo tipo de errores en el Visual Studio Code



Error Lens

- Lee errores a nivel de línea
- Señala de una manera más clara las líneas erróneas (incluso con TypeScript)

TotalTypescript

- Te va dando pistas y tips de aprendizaje de TypeScript en el código

Cierre > Ecosistema

IDEs

Visual Studio Code

- Con extensiones
- De Microsoft

WebStorm

- Previo pago
- De JetBrains

Terminal

- Se pueden configurar plugins y LSPs, pero es más específico y no hay recomendaciones exactas *out of the box*

Algunas opciones podrían ser:

- NVim
- Vim
- Emacs

Cierre

Utilidades

Etiquetas de comentarios

- @ts-check
- @ts-nocheck
- @ts-ignore
- @ts-expect-error

Prettify<T>

- { [k in T]: T[k] }

ObjectValues

- T[keyof T]
- Se le puede hacer un zip ({ [k in {T}]: {T}[k] })

Extensión de lookup

- Twoslash Query Comments
- // ^?
  - Donde sea que pongas el ^? te chivará su tipo

Retorno de keys dinámicas

https://medium.com/@jofaval/objecy-with-dynamic-keys-in-typescript-468c358a5f8b

- Keys nombradas dinámicamente correctamente tipadas

Siempre podrás experimentar

En tsplay.dev

Cierre

Type-safe

type-safety

Cuando vamos más allá de cubrir los tipos, aunque son nuestra primera línea de defensa.

Tipados > Genéricos > Inferencia > Apenas uso TypeScript como Dev

Cierre

Developer Experience

Developer Experience

No es solamente DevOps, métricas, DORA, SPACE, etc.

Developer Experience es un **Developer Journey accesible y cómodo**

Cierre

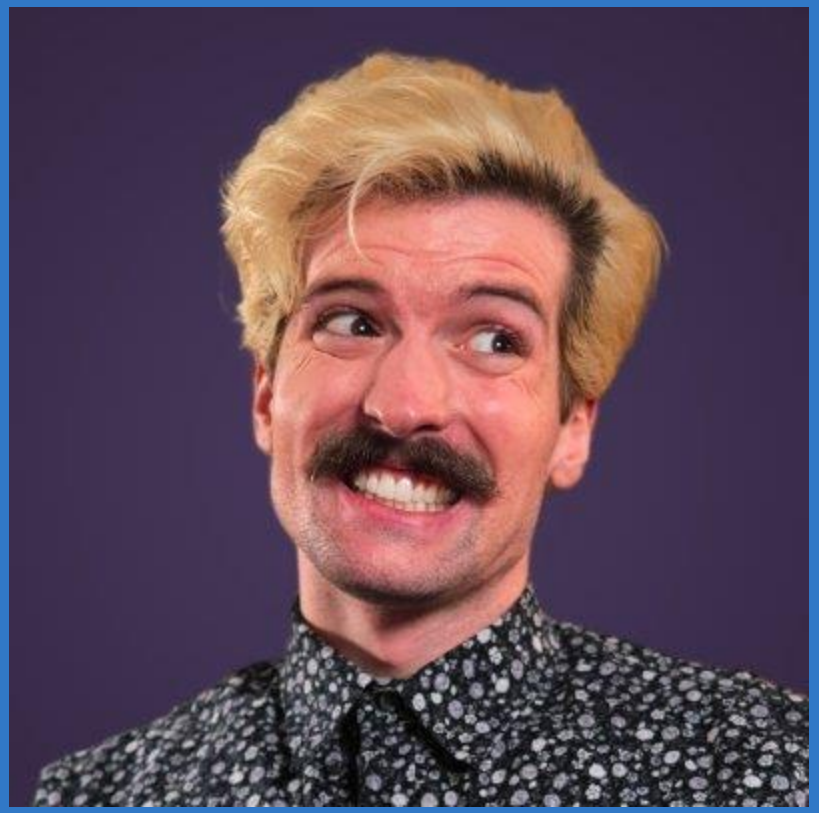
Referencias

TotalTypescript y Matt Pocock

Theo Browne (t3dotgg)

Creador de Contenido

Ex-Twitch y Ex-Amazon



Tanner Linsley

Open-Source, creador de React Table, React Query y muchos más

Cierre

Siguientes pasos

TypeScript Handbook

https://www.typescriptlang.org/docs/handbook/intro.html

Total TypeScript

- Curso completo con secciones gratuitas

TypeChallenges

- Retos increcendo de TypeScript
- Leetcode pero de tipados

TypeHero

- Advent of TypeScript

Advent of TS

- Advent of Code de thrashdev (Netflix Engineering)
- Las soluciones son tipados de TypeScript

Advent of JS

- Advent of Code de midu.dev
- Permite soluciones con TypeScript

JJS

Explore the JavaScript Universe

Rebuild your mental model from the inside out.

WITH
Dan Abramov & Maggie Appleton

Conclusiones

QR de las slides

/tech-talks/valencia-js/la-magia-de-typescript

QR del workshop

/talks-about/workshops/typescript/

Feedback

Por LinkedIn

En persona

Encuéntrame en

- LinkedIn - https://www.linkedin.com/in/jofaval/
- Github - https://github.com/jofaval

La magia de TypeScript

Gracias por la atención

Preguntas