Workshop de TypeScript

Me presento

Pepe Fabra Valverde

Líder y Arquitecto de Front

Agenda del taller

Agenda del taller

1h 15m – Introducción de los conceptos, hasta intermedio

Agenda del taller

1h 15m – Introducción de los conceptos, hasta intermedio

30m aprox. – Descansito

Agenda del taller

1h 15m – Introducción de los conceptos, hasta intermedio

30m aprox. – Descansito

1h 15m – Intermedio hasta avanzado

Disclaimer

- La sesión se va a grabar
- Se compartirán las slides al terminar
- Pregunta sin miedo

TypeScript

Qué es TypeScript

JavaScript con tipos… o eso dicen

Qué es TypeScript

JavaScript con tipos… o eso dicen

Es un sistema de tipados que ofrece solución al problema de no saber con qué se está trabajando en JavaScript

- Tipos, interfaces, enums (hablaremos más adelante)

Orígenes de TypeScript

- Desarrollado por Microsoft en 2012

Orígenes de TypeScript

- Desarrollado por Microsoft en 2012
- Ya existían otras soluciones de tipado en JavaScript
  - CoffeeScript, Flow (Meta), etc.

Orígenes de TypeScript

- Desarrollado por Microsoft en 2012
- Ya existían otras soluciones de tipado en JavaScript
  - CoffeeScript, Flow (Meta), etc.
- Solución integral con un sistema más complejo
  - No solamente tipados básicos
  - Inferencias, genéricos, herencia completa

Sistemas de tipados

Sistemas de tipados

- Qué son
- Qué esperamos de ellos
- Otros lenguajes

Qué es un sistema de tipados

Qué esperamos de un sistema de tipados

Otros lenguajes

No todos los lenguajes son tipados, y los que lo son, no son iguales

Existen dos clases de lenguajes tipados

- Fuertemente tipados
  - Java, C#, Rust, Go
- Débilmente tipados
  - JavaScript, Python, PHP… TypeScript

Fuertemente tipado

Débilmente tipado

Primitivos

Primitivos de JavaScript

typeof

Objetos

Funciones

Funciones en TypeScript

Existen dos maneras de tipar funciones

(…params: Parameters) => ReturnType

Nos vamos a centrar en la más conocida y usada

Un único argumento

- A las funciones que generemos, idealmente, les pasaremos un único argumento
- Un objeto `key: value` que contenga lo que necesitemos

Enums

Qué son los Enums

- Son enumeraciones
- Categorizaciones para un único valor
- Comúnmente representados como números

```
enum ValoracionTaller {
  pésimo = 0,
  decente = 1,
  increíble = 2,
}
```

Enums en TypeScript

Enums no oficiales

Target time: 5m

Demo

Experimentar con los diferentes enums

- Prueba a mostrarlos en consola

Opcionales y defaults

Opcionales

Existen dos maneras de representar opcionales

- Usar un interrogante

Tipos especiales

Los internals de TypeScript

Hay muchos

Así que cubriremos los más usados y comúnes

void

No hay tipo, no hay nada

Una función puede devolver void, en más de una ocasión lo hará

Se pueden tipar props como void (y que no se le pasen nunca)

never

- Nunca se ha de llegar aquí
- Si una posibilidad llega a never, se devolverá error
- Usado para switches exhaustivos
  - Agotar todas las posibilidades de un enum/conjunto de literales en un switch

No pasar argumentos

Tenemos dos opciones para *prohibirlo*

- void
- never

{}

unknown

any

const

as

Valor *as* {Type}

const

En TypeScript una keyword bastante mágica es *const*

Permite hacer una inferencia más cercana a la realidad. Son tipados más estrictos, pero porque su intención es facilitar el desarrollo.

Usados para devolver literales, objetos *narrowly typed*

as const

Operaciones

&

Literales

Literales

Los vamos a considerar tipos, aunque sean definidos en desarrollo

Literales abiertos

A veces podemos querer tipar literales, pero no restringir más opciones

La solución es:

“literal” | “otraOpción” | (string & {})

¿Quién sabría decirme por qué?

Porque

Prefijos y sufijos

Limitaciones de inputs

Comparación

Filosofía de comparación

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

Ejemplos colaborativos

De más genérico a específico (moderadamente)

`string`

De más específico a genérico (suficientemente específico)

`“literal”`

Ejemplos colaborativos

De más genérico a específico (moderadamente)

✓ string

string

De más específico a genérico (suficientemente específico)

✓ “literal”

string

Ejemplos colaborativos

De más genérico a específico (moderadamente)

✓ string

✓ string

“literal”

De más específico a genérico (suficientemente específico)

✓ “literal”

✓ string

string

Ejemplos colaborativos

De más genérico a específico (moderadamente)

✓ string

✓ string

× “literal”

“{number}-ex”

De más específico a genérico (suficientemente específico)

✓ “literal”

✓ string

× string

“tt-ex”

Ejemplos colaborativos

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”

Ejemplos colaborativos

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”

{ foo: “bar” }

Ejemplos colaborativos

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

Ejemplos colaborativos

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

Type e Interface

Type

- Se comporta como *const*
- Herencia con unión types

Interface

- Se comporta como *var*
- Usado en desarrollo de librerías
- Herencia

Diferencias

¿Cuándo usar cada uno?

Type vs Interface

Utility Types

Pick

Omit

Exclude

ReturnType

Parameters

Genéricos

Genéricos

- Qué es un genérico
- Operador diamante
- Filosofía de los genéricos

Qué es un genérico

Cuándo se dan los casos de uso

- Un genérico no siempre hace falta
- Cuando los aprendemos, es común querer ponerlos en todos lados

Genéricos usando *as*

tsconfig.json

Inversión de tiempo

Salvo desarrollo de librerías, la mayoría de aplicaciones deberían usar configuraciones muy similares

strict

Debería ir siempre a *true*

noUncheckedIndexedAccess

Debería ir siempre a *true*

TotalTypescript

Module declarations

.d.ts

Estructura del fichero

vite-env.d.ts

Target time: 5m

Demo

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

Extensiones y su conversión

- TypeScript .ts -> .js
- Common TypeScript .cts -> .cjs
  - Usado con node, para CLIs
- .mts -> .mjs
  - Exportación a módulos, no retrocompatible con todos los navegadores

Caso de uso

Tipar código antiguo

Preguntas

Descanso

Vuelta del descanso

¿Qué nos queda?

- Genéricos
- Inferencias
- JSDoc

¿Qué nos queda?

- Genéricos
- Inferencias
- JSDoc

Poco pero contundente

Inferencias

La verdadera magia de TypeScript

Saber más – Inferencia

- Qué es una inferencia y qué es inferir
- De dónde viene
- En qué campos más se usa

Inferencias en TypeScript

Cuándo se infiere un valor

- Los tres casos, vídeo de Matt Pocock
- Función sin tipar

Tipado de retornos, ¿cuándo?

GenericBags

Para cuando tienes demasiados genéricos

A veces tendrás que pasar varios genéricos, en un orden específico, y luego vendrán los opcionales

- Usa un generic bag para que sea más fácil de consumir

infer

JSDoc

Cómo documentar código en JavaScript

Y tiparlo **nativamente**

Qué es JSDoc

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

¿Se pueden documentar tipos en TypeScript?

Para saber más

https://tsdoc.org/

Casos de uso

- Desarrollo y mantenimiento de librerías
  - Svelte
- DDH (Ruby on Rails)
  - El Elon Musk de la programación
- Saltarte paso de build

@var {Type} name

@returns {Type}

@author y @source

Conceptos avanzados

Referencias a otros tipos

Genéricos

Inferencia

Target time: 5m

Demo

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

Utilidades

Etiquetas de comentarios

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

Etiquetas de comentarios

- @ts-check
  - Comprueba TypeScript en ficheros de JavaScript
- @ts-nocheck
  - No comprueba TypeScript en el fichero
- @ts-ignore
  - No comprueba la siguiente línea
- @ts-expect-error
  - Esperará un error en la siguiente línea
  - Si no lo hay, dará un error (de ts, no de js)

Prettify

ObjectValues

Extensión de lookup

Retorno con keys dinámicas

- Artículo medium

Siempre podrás experimentar

En tsplay.dev

Zod

Validaciones en runtime

Qué es Zod

Validador y luego tipado

- Type-safe
- Realidad y dev-time iguales

¿Qué es type-safe?

Bonus

Referencias

TotalTypescript y Matt Pocock

Theo Browne (t3dotgg)

Creador de Contenido

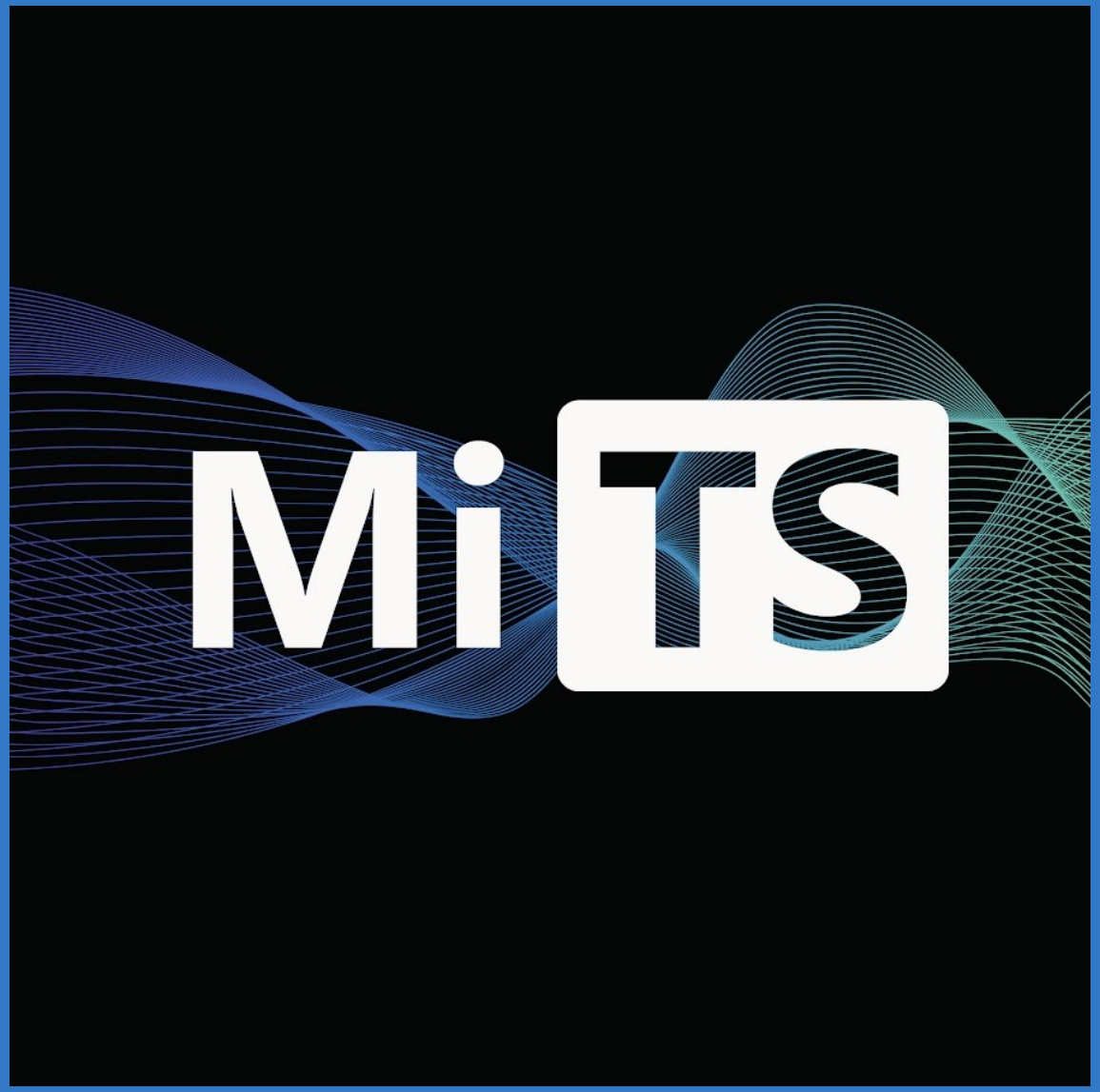
Ex-Twitch y Ex-Amazon

Tanner Linsley

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

Michigan TypeScript

Canal y comunidad



Librería de utilidades

ts-toolbelt

TypeChallenges

Retos increcendo de TypeScript

Leetcode pero de tipados

TypeHero

Advent of TypeScript

Advent of JS

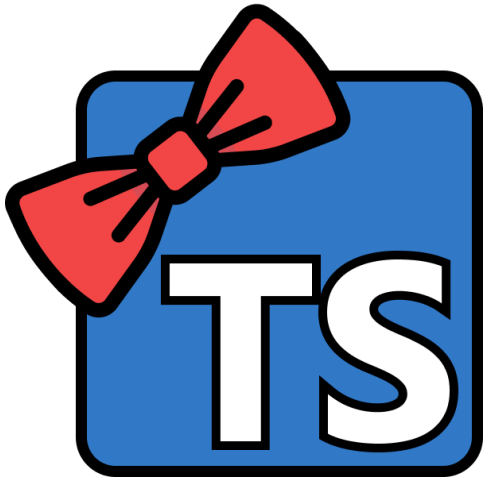
Advent of Code de midu.dev

Permite soluciones con TypeScript

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

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

Conclusiones

QR de las slides

/talks-about/workshops/typescript/

Encuéntrame en

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

Preguntas

Workshop de TypeScript

Gracias por la atención