

# ¿Qué es Typescript

## ¿Por qué Vue usa Typescript?

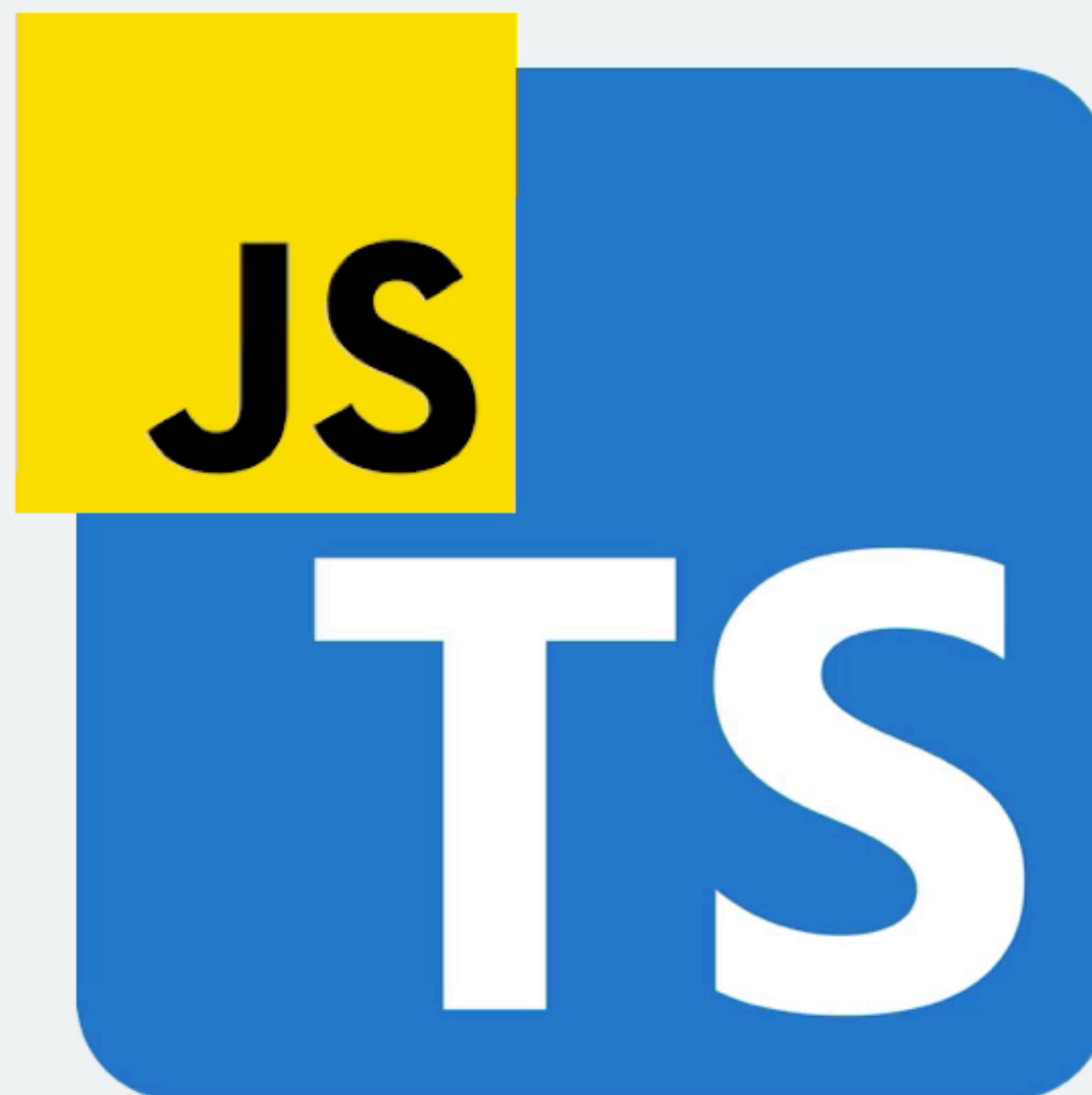
El navegador no puede ejecutar Typescript y por ello se compila a Javascript

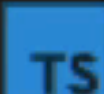
# ¿Qué es TypeScript?

Si sabes JavaScript, conoces el 80% de TypeScript

Ts es un superset del standard Js:

Es el mismo Js pero le extiende y le añade características que no tiene Js



 01-ejercicios-basicos.ts

```
let nombre = 'Strider';  
let puntosVida = 95;  
let arma = 'Espada';
```

Los archivos .ts son de código typescript

Microsoft es quien  
mantiene y soporta a Ts.

# ¿Qué es TypeScript?

- Es fácil de aprender y de leer.
- Tiene muy buena ayuda contextual
- Tiene autocompletado
- Es un super set de JavaScript.
- Ofrece tipado estricto y flexible.
- Mejora enormemente la legibilidad del código.
- Nos permite usar características modernas.
- Se trabaja con módulos de forma muy robusta
- El navegador no puede ejecutar TS y por ello se compila a Js





# El tipado estricto nos ayuda a saber cómo funcionan las cosas

JS

```
function calculaISV( productos ){  
  let total = 0;  
  productos.forEach( ({ precio }) => {  
    total += precio;  
  });  
  return [total, total * 0.15];  
}
```

- ¿Qué tipo de dato son los productos?
- ¿Cómo luce un producto?
- ¿Qué propiedades debe de tener el producto?



# Ayudas instantánea de funciones y métodos

Arreglo de Producto



```
interface Producto {  
  desc: string;  
  precio: number;  
}
```



```
function calculaISV( productos: Producto[] ): [number, number] {  
  
  let total = 0;  
  
  productos.forEach( ({ precio }) => {  
    total += precio;  
  });  
  
  return [total, total * 0.15];  
}
```

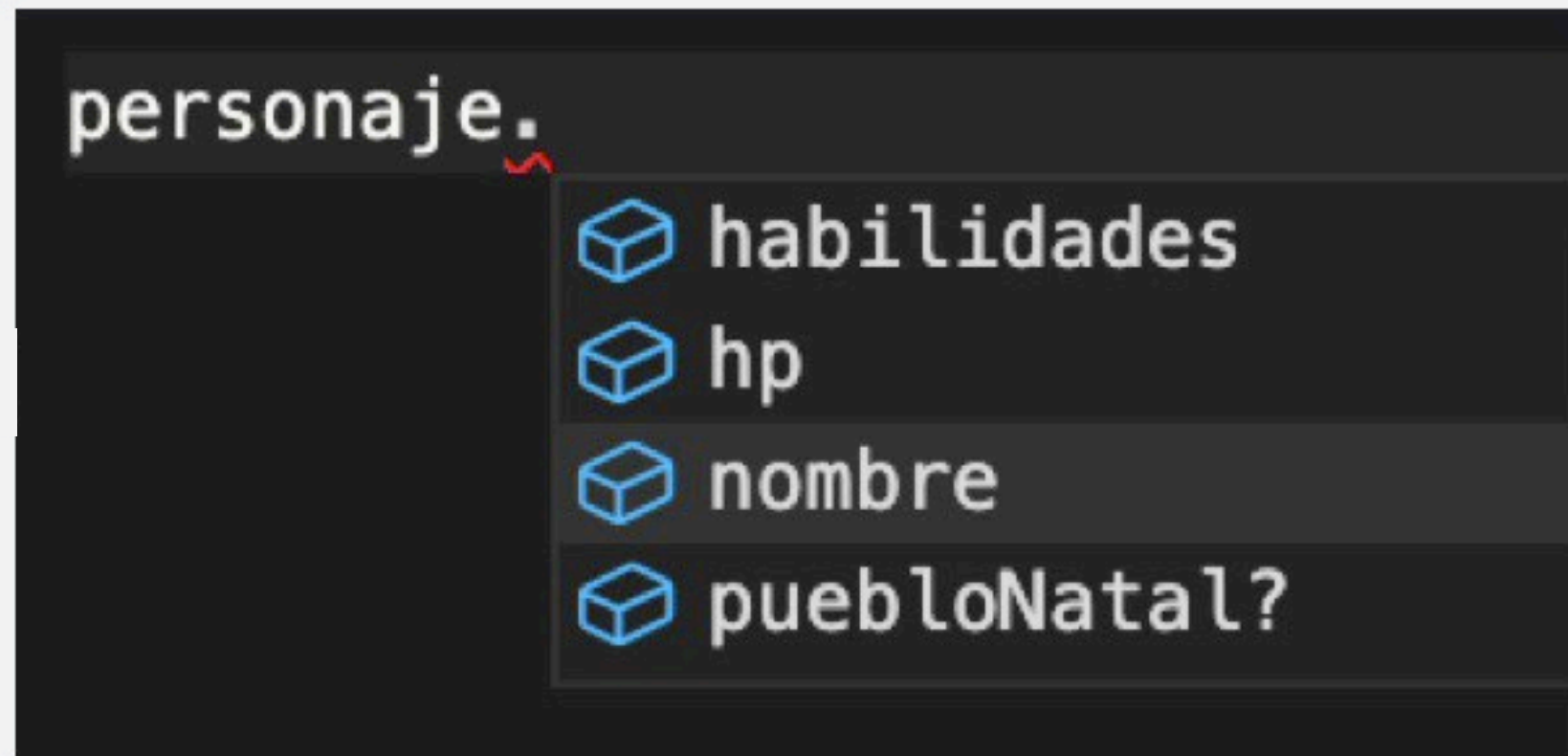


Retorna un arreglo con dos números

TS



# Mejora enormemente el intellisense

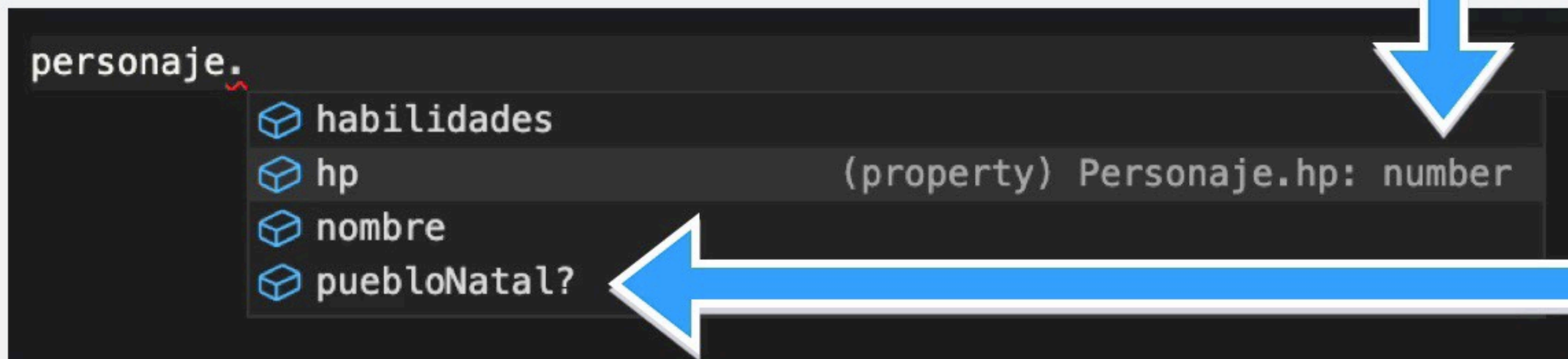


```
function sumar(a: number, b: number): number  
const resultado = sumar(10, 20);
```

Argumento “a” y “b” son números

HP es un número

Retorna un número

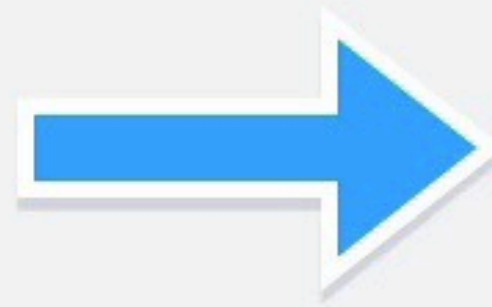


puebloNatal es opcional



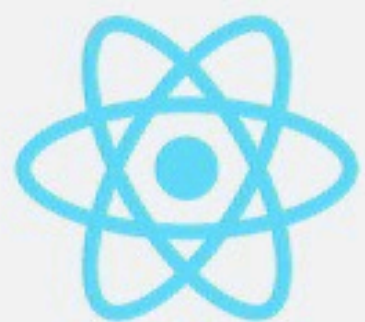
# ¿Por qué Vue usa TypeScript?

```
class Persona {  
  nombre: string;  
  edad: number  
  
  constructor() {}  
}
```



```
class ProductosComponent {  
  
}  
  
class ProductosService {  
  
}  
  
class ProductosDirective {  
  
}  
  
class ProductosPipe {  
  
}
```

Tener todos los beneficios de TypeScript



@Decoradores

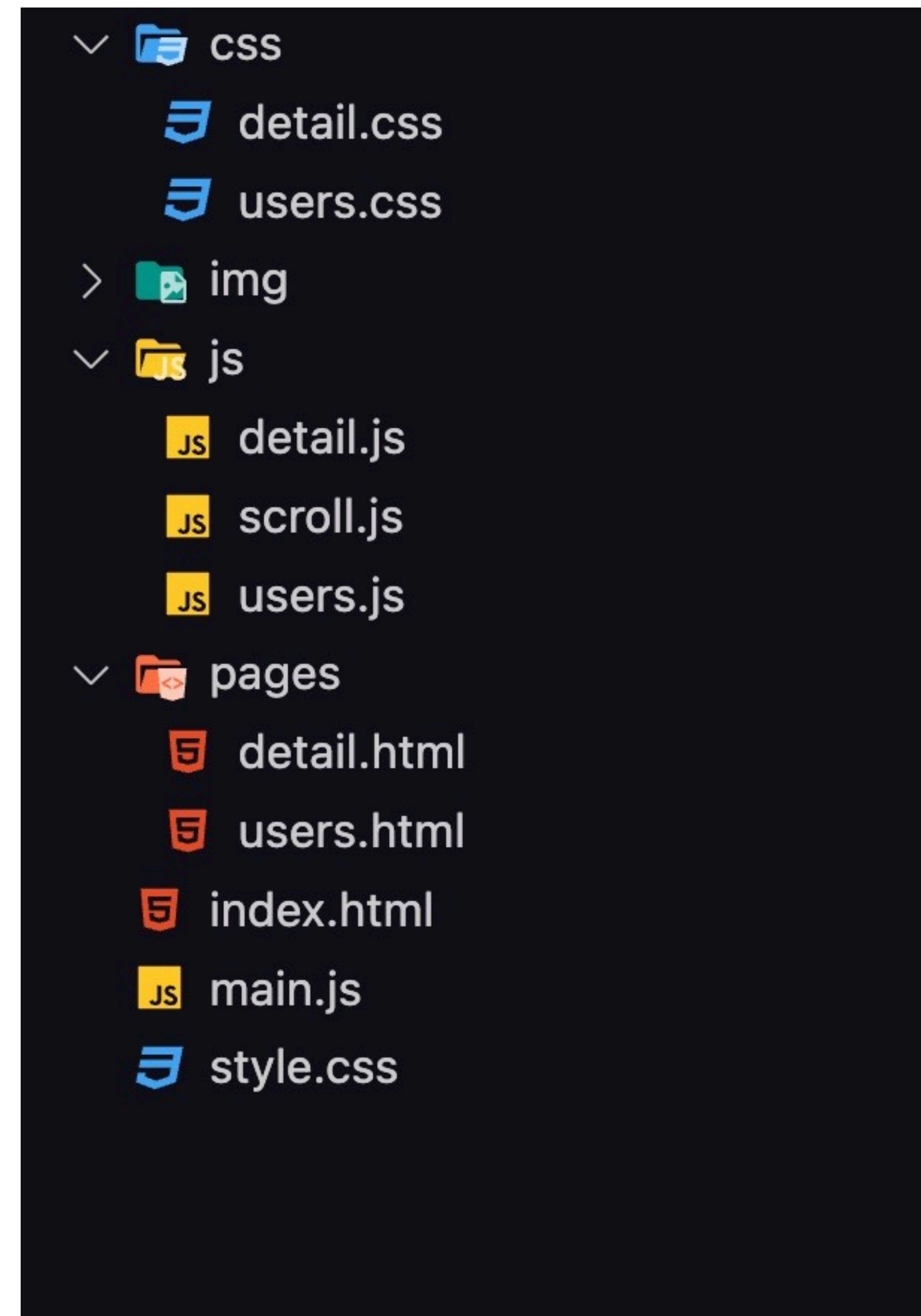
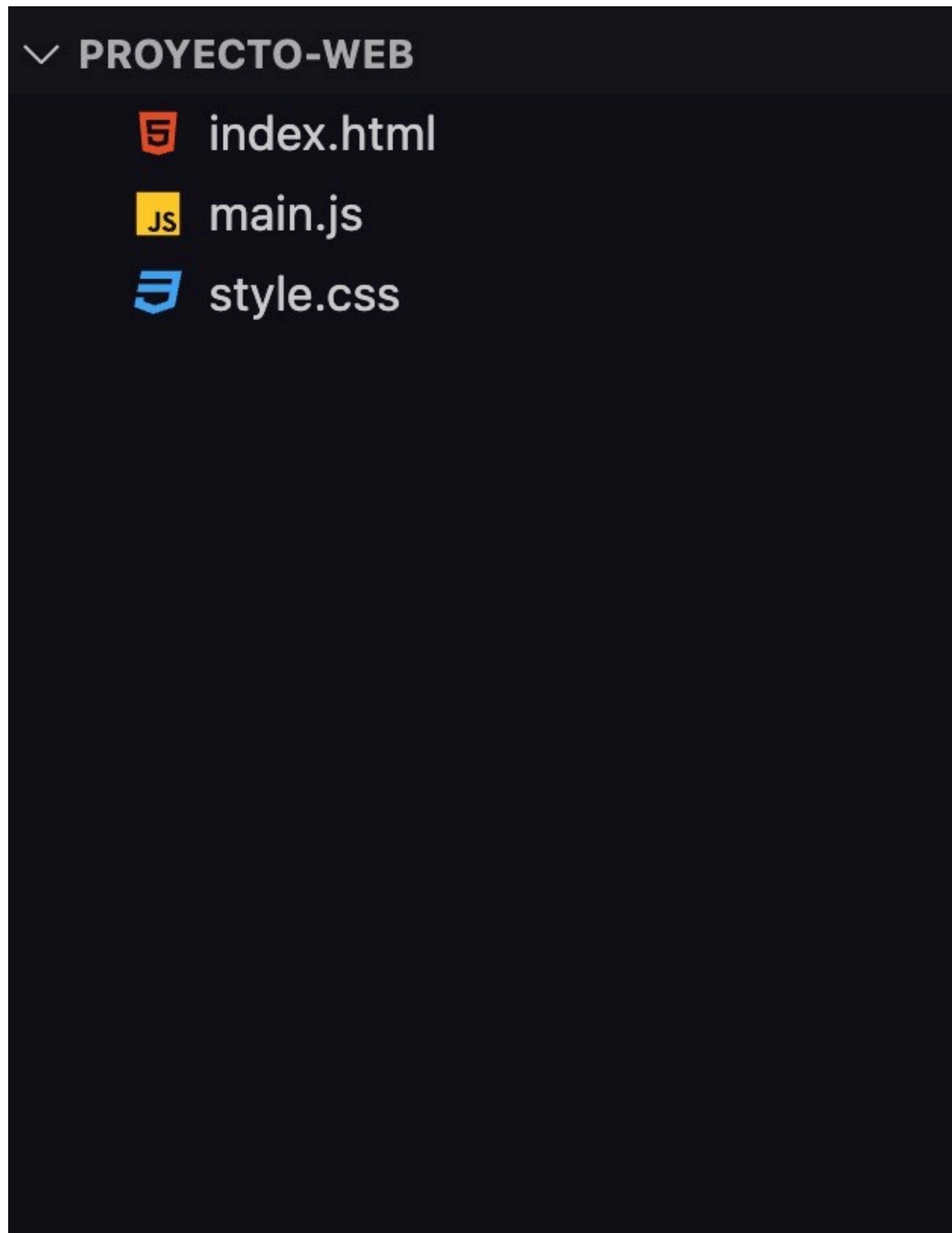


# ¿Por qué Vue usa Typescript?

- Datos genéricos: reutilización de código, funciones flexibles.
- Interfaces: objetos con características concretas (que el editor sabe)
- Enums y Tuplas
- Tenemos la ayuda y un intellisense fuerte.
- Tenemos tipado estricto y errores en momento de escritura.
- Nos permite la inyección de dependencias.



El modo tradicional es crear un archivo html y vincularle un archivo css y otro js. Luego se pueden ir añadiendo más directorios con más archivos.



Una vez terminado el proyecto hay que preparar todo para subirlo a un servidor.

Tenemos que tener en cuenta que corra en todos los navegadores y que lo haga de forma fluida, sin mucha demora.

Hacer esto a la manera tradicional de desarrollo web es muy laborioso.



Cuando salgáis a la calle a trabajar veréis que ya nadie lo hace así. Se trabaja con frameworks como Qwik, Angular, React, Vue o con herramientas como Vite o Webpack. Y todos ellos están basados en los estándares más actuales de Js.



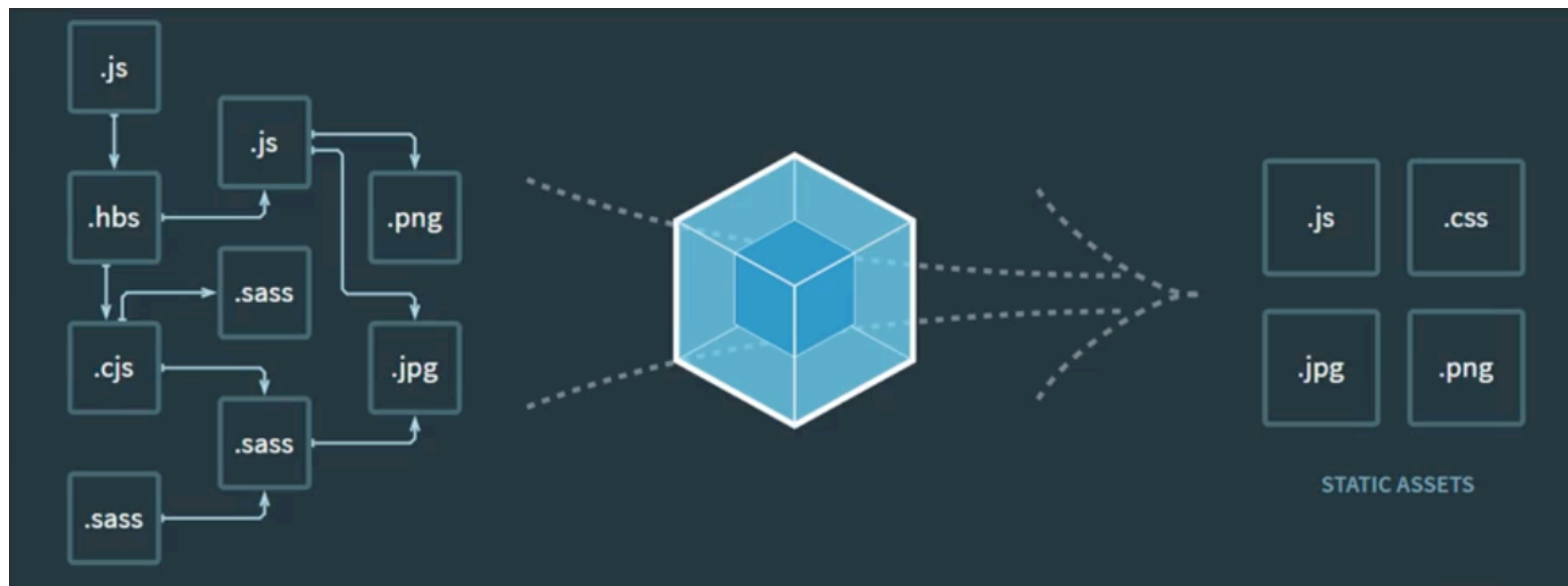
Vite



Vite nos ayuda a generar el build de producción de nuestra app para poder desplegarla fácilmente en un hosting y que otras personas lo puedan ver en internet.

Y también hace el cambio de módulos en caliente. De ese modo cuando hacemos modificaciones en cualquier de nuestros archivos el resultado es inmediato en nuestro navegador web.

Trabaja con las últimas características de Javascript y finalmente traduce el código a los estándares de Js que soportan todos los navegadores.





# Lenguaje Typescript (Javascript)



- Typescript es un lenguaje de “tipado estricto”.
- Tipos primitivos: number, string, boolean, null y undefined.
- Si no se pone el tipo explícitamente Typescript lo infiere.

# Tipos y valores

```
let numero: number = 10;           // Comentario de una línea
const anotherNumber = 2 * 3;       /* Comentario
const saludo: string = "hola";     de varias líneas */
const cadena = 'Esto es una frase';
let boolTrue = true;
const negacion: boolean = !false;
let mayorQue: boolean = 2 > 1;
let nullValue = null;
let valorUndefined;
```



# Estructura condicional

Se trata de una sentencia de control que nos permite controlar si una expresión es verdadera o falsa.

```
if (5 > 6) {  
    return false;  
} else {  
    return true;  
}
```

Operador ternario

```
const resultado = 5 < 6 ? false : true
```

# Arrays o Listas

- Un array es una estructura que nos permite almacenar más de un valor.
- Tiene forma de una lista ordenada de elementos. Los elementos puede ser cualquier tipo de valores primitivos, otro array o un objeto.
- Todos los arrays nos ofrecen un conjunto de métodos para operar sobre ellos, como por ejemplo para añadir o sacar elementos.

# Arrays o Listas

```
let soyUnArrayVacio = [];
```

Todos los arrays tiene una propiedad “length” que nos permite saber su longitud.

```
soyUnArrayVacio.length; // length === 0
```

Con el método “push” añadimos elementos al final de la lista

```
soyUnArrayVacio.push(1); // soyUnArrayVacio === [1]
```

Con el método “pop” sacamos el ultimo elementos de la lista

```
soyUnArrayVacio.pop(); // soyUnArrayVacio === []
```

También podemos inicializar un array con unos valores por defecto

```
const homogeneo: number[] = [ 1, 2, 45, 88 ];
```

```
let heterogeneo: (number | boolean)[] = [ 2, true, 9 ];
```



# Objetos

- En JavaScript y Typescript casi todo son objetos.
- Los objetos pueden contener propiedades (variables) o métodos (funciones).

```
let unObjeto = {  
  nombre: 'Fernando',  
  apellidos: 'Cordón',  
  edad: 22,  
  saludo: () => 'Buenos días'  
};
```

## Objeto tipado

```
let uObjeto: {nombre:string, apellidos:string, edad:number, saludo:()=>string} = {  
  nombre: 'Fernando',  
  apellidos: 'Cordón',  
  edad: 22,  
  saludo: () => 'Buenos días'  
};
```

# JSON - Javascript Object Notation

- Es un formato para intercambio de datos, derivado de la notación literal de objetos de Javascript.
- Se usa habitualmente para serializar objetos o estructuras de datos.
- Se ha popularizado mucho principalmente como alternativa a XML, por ser más ligero que este.

# Convirtiendo un objeto a JSON

```
let empleado = {  
  nombre: 'Thomas Anderson',  
  profesion: 'Developer'  
};
```

```
JSON.stringify(empleado);
```

```
// produce un string
```

```
'{"nombre": "Thomas Anderson", "profesion": "Developer"}'
```



# Convirtiendo un JSON a objeto

```
let textoJSON = '{  
  "nombre": "Thomas Anderson", "profesion": "Developer"  
';  
  
JSON.parse(textoJSON);  
  
// produce un objeto javascript  
{  
  nombre: "Thomas Anderson",  
  profesion: "Developer"  
}
```

# Funciones

- Una función es un conjunto de acciones (sentencias) que se ejecutan de forma secuencial para realizar una tarea concreta.
- Una función consta de un nombre (opcional), unos parámetros y su alcance, definido por unos corchetes ( { } ).

# Funciones

```
function soyUnaFuncion() {  
    return 'Hola';  
}
```

```
let funcionAnonima = function(): string {  
    return 'Hola anónima';  
}
```

Para ejecutar una función solo necesitamos poner su nombre seguido por unos paréntesis ( ).

```
soyUnaFuncion();
```

```
function conParametros(parametro: number): void {  
    console.log(parametro);  
}
```

```
conParametros(1);
```



# Arrow Functions

```
let arrowFunc1 = (): string => {  
    return 'Hola' + nombre;  
}
```

```
let arrowFunc1 = (): string => 'Hola'
```

```
const arrowFunc2 = (param1): string => 'Hola arrow';
```

```
const arrowFunc3 = (param1, param2) => {  
    return 'Hola arrow';  
}
```

```
const arrowFunc3 = (param1, param2) => 'Hola arrow';
```