TypeScript

TypeScript és un llenguatge de programació que, en el nostre cas, no s'executa directament sinó que es transpila a JavaScript de forma que pugui ser executat a un navegador web.

Al fitxer tsconfig.json del nostres projecte podem veure les opcions de transpilació.

L'aportació principal de TypeScript sobre JavaScript és el tipatge de dades.

Tipus de dades

La forma de tipar en TypeScript és afegint : darrera la definició de la variable, paràmetre, funció...

```
const age: number = 33;
function max(numbers: number[]): number {...}
```

Arrays

```
const numbers: number[] = [0,1,2];

const numbers: Array<number> = [0,1];

// Varis tipus al mateix temps
const mixed: (number|string)[] = [0,'hola'];

const mixed: Array<number|string> = [0,'hola'];
```

Enum

Definició d'enumerats

```
enum Cardtype {Hearts, Diamons, Spades, Clubs}

let myCard: Cardtype = Cardtype.Hearts;
let name: string = Cardtype[2];
```

any

Es equivalent a no definir tipus

```
const data: any = "4";
const mix: any[] = [1,"asd",false];
```

void

Per indicar que un mètode no retorna res

```
function greet(name: string): void {
    return;
}
```

Objectes

Amb TypeScript podem definir clases i herència

```
class Person {
    // propietats
    name: string;

    // constructor
    constructor(name: string) {
        this.name = name;
    }

    // mètodes
    greet(): string {
        return `Hello ${this.name}`;
    }
}
```

Les propietats poden tenir modificadors

```
class Person {
    // Per defecte
    public name: string;
    // Impedeix l'acces directe
    private age: number:
    // Accesible per subclases
    protected lastname: string;
    // No es por modificar
    readonly car: string;
}
```

Interface i implements

Es poden definir interficies i aquestes ser implementades

```
interface IUser {
  name: string;
```

```
  age: number;
}

class Person implements IUser {...}
```

Herencia

Les classes i les interficies es poden extendre entre elles

```
// Classes
class Person {...}
class Hero extends Person {...}

// Interficies
interface IPerson {...}
interface IHero extends IPerson {...}

// Es poden combinar
class Hero extends Person implements IHero {...}
```

També es poden definir classes abstractes que no podran ser instanciades però si exteses

```
abstract class Human {
   public name: string;
   public age: number;
   constructor(name: string){
      this.name = name;
   }

   abstract greet(): void;
}

class Person extends Human {
   constructor(name: string) {
      super(name);
   }
   greet(): void {
      console.log('Hello')
   }
}
```

Generics

Es poden definir tipus genèrics que s'establiran durant la declaració

```
function picker<T>(args: T[]):T {
   const randomIndex = Math.floor(Math.random()*args.lenght);
```

```
    return args[randomIndex];
}
```