SYNTACTIC SUGAR: CLASES

DEFINIENDO CLASES

```
class Person {
    name;

    constructor(name) {
        this.name = name;
    }

    greet() {
        console.log("Hola, mi nombre es " + this.name);
    }
}
```

Mediante la expresión class, declaramos una clase

DEFINIENDO CLASES SIN CONSTRUCTOR

```
class Person {
    name;

    /*constructor(name) {
        this.name = name;
    }*/

    greet() {
        console.log("Hola, mi nombre es " + this.name);
    }
}
```

El constructor es opcional

MODIFICADORES DE ACCESO

- Propiedades y métodos pueden ser:
  - Públicos (por defecto)
  - Privados
  - Estáticos

MODIFICADOR PÚBLICO DE ACCESO

```
class Person {
    name;

    greet() {
        console.log("Hola, mi nombre es " + this.name);
    }
}

let user = new Person();
user.name;
user.greet();
```

Permite su acceso desde fuera del ámbito de clase

MODIFICADOR PRIVADO DE ACCESO

```
class Person {
    #name;

    greet() {
        console.log("Hola, mi nombre es " + this.#name);
    }
}

let user = new Person();
user.#name;
user.greet();
```

No permite su acceso desde fuera del ámbito de clase

MODIFICADOR ESTÁTICO DE ACCESO

```
class Person {
    static NAME = "algun valor";

    greet() {
        console.log("Hola, mi nombre es " + this.NAME);
    }
}

let user = new Person();
console.log(user.NAME); // <-- undefined
console.log(Person.NAME); // <-- algun valor
```

Permite acceso desde fuera del ámbito de clase sin instancia

GETTER

```
class Person {
    #name = "Manuel";

    get name() {
        return this.#name;
    }
}

let user = new Person();
console.log(user.name);
```

Un getter es un método que retorna el valor de una propiedad

SETTER

```
    #phone;

    set name(value) {
        this.#name = value;
    }

    set phone (value) {
        if (this.#isValidPhone(value)) {
            this.#phone = value;
        }
    }

    #isValidPhone(value) {
        // ...
    }
}

let user = new Person();
user.phone = "adgadg"; // <-- Invalid value for phone.
user.phone = 600123456; // <-- All OK
```

Un setter es un método que establece el valor de una propiedad

HERENCIA

```
class Person {
    #name;

    constructor(name) {
        this.#name = name;
    }
}

class Student extends Person {
    #level;

    constructor(name, level) {
        super(name);

        this.#level = level;
    }
}
```

Una clase puede heredar propiedades y métodos de otra

EXPRESIONES DE CLASES

- new
    - Se usa para crear instancias de clases y tipos
- super()
    - Es un método que se usa para llamar al constructor padre cuando se usa herencia de clases
- this
    - Es una palabra clase para referirse al objeto actual que está en el ámbito donde nos encontramos