6. POO

C. Clases en JavaScript Moderno

1. Class

- Con JavaScript ES5 aparece la palabra reservada **class**
- Esta class siempre va a llevar asociado un método **constructor**, que será el que defina e inicialice los atributos.
- Igual que antes, es conveniente utilizar valores por defecto para que nos sirva para crear objetos con o sin parámetros.

```
class Punto {
    constructor(ejex = 0, ejey = 0) {
        // constructor
        this.x = ejex;
        this.y = ejey;
    }
```

```
var punto2=new Punto();
var punto1=new Punto(2,4);
```

1. Class

- Los métodos ahora ya no necesitan ni this ni function, tan solo con el nombre es suficiente (y los parámetros, por supuesto).

```
mostrar () {
        document.write("(");
        document.write(this.x);
        document.write(",");
        document.write(this.y);
        document.write(")");
}
```

2. Getters y Setters

- Aunque en JavaScript todos los atributos son públicos (de momento no se pueden hacer privados), es conveniente implementar las funciones para obtener los atributos (**get**) y para establecerlos (**set**)

```
get valorx() {
    return this.x;
}

set valorx(parametro) {
    this.x=parametro;
}

get valory() {
    return this.y;
}

set valory(parametro) {
    this.y=parametro;
}
```

3. toString

- En lugar del método mostrar, se suele implementar un método llamado **toString** que devolverá los valores del objeto que queremos que se representen.

```
toString () {
    var salida="(";
    salida=salida+this.valorx;
    salida=salida+",";
    salida=salida+this.valory;
    salida=salida+")";
    return salida;
}
```

4. Herencia

- Podemos hacer otra clase de herede de la clase padre
- Para ello usaremos en la definición la palabra reservada **extends** y el nombre de la clase padre.
- En este caso el constructor debe incluir una sentencia super(); indicando que estamos utilizando el constructor del padre.

```
class Vector extends Punto {
    constructor() { //
        super(); //utiliza el constructor del padre

    }
```

4. Herencia

- En el caso de clases hijas, podemos definir también una serie de métodos que estarán disponibles para objetos de esta clase

```
class Vector extends Punto {
    constructor() { //
        super(); //utiliza el constructor del padre

    }

    sumaVector(a,b) {   // a es un objeto de tipo punto, b es otro objeto de tipo punto
        this.valorx=(a.valorx+b.valorx);
        this.valory=(a.valory+b.valory);

    }
}
```

5. Métodos privados

- Que queremos que un **método** sea **privado**, es decir, solo pueda usarse dentro de la propia clase, pero fuera no (incluso en la herencia se pierde), debemos usar el modificador **static** antes del nombre del método.

```
static sumaVector(a,b) {
    this.valorx=(a.valorx+b.valorx);
    this.valory=(a.valory+b.valory);

}
```