



6. POO

b. Forma Clásica

1. Uso de una función constructora

- En JavaScript clásico para definir una clase nos valemos del uso de **function**.
- Conceptos como el de **this** siguen valiendo para este tipo de definición.

1. Uso de una función constructora

```
function Punto(ejex=0, ejey=0) { //clase Punto
|   // constructor
this.x=ejex;
this.y=ejey;

// metodos

this.mostrar=function() { // representa el punto (x,y)
|   document.write("(");
|   document.write(this.x);
|   document.write(",");
|   document.write(this.y);
|   document.write(")");
| }
}
```

1. Uso de una función constructora

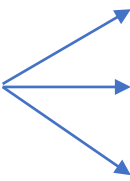
- Los atributos son declarados con **this.nombreatributo** al igual que los métodos, que, de nuevo, vuelven a tener asociada una function.
- Los atributos y métodos, de nuevo, vuelven a separarse por ; en lugar de , como se hacia en la forma anterior.



2. Constructor por defecto

- Como en JavaScript no existe sobrecarga, se suele utilizar un constructor por defecto asignando valores por defecto al constructor de la clase.
- De esta forma cuando creemos un objeto nuevo de esta clase podemos pasarlo con o sin parámetros.

```
function Punto(ejex=0, ejey=0) {  
    ...  
}
```



```
var punto1=new Punto(); // (0,0)  
var punto2=new Punto(2,4); // (2,4)  
var punto3=new Punto(1); // (1,0)
```

3. Clase con varios métodos

- Podemos definir tantos métodos como queramos, siempre y cuando estén igualados a la palabra reservada function.

```
this.mostrar=function() { // representa el punto (x,y)
    document.write("(");
    document.write(this.x);
    document.write(",");
    document.write(this.y);
    document.write(")");
}
```

```
this.modulo=function() {
    // raiz cuadrada de la suma de los cuadrados sqrt((x*x)+(y*y))
    let sumacuadrados=(this.x*this.x)+(this.y*this.y);
    let solucion=Math.sqrt(sumacuadrados);
    return solucion;
}
```