





**“Un objeto en programación es una entidad que representa una colección de datos y el comportamiento asociado a esos datos. Piensa en ello como una caja que contiene información (atributos) y acciones que puedes realizar con esa información (métodos)..”**

“Un objeto, por otro lado, es una instancia de una clase. Es decir, es un **elemento concreto que se crea a partir de una clase** y tiene sus propias propiedades y métodos”.

# Añadir métodos a nuestras *classes*

```
4  class Cliente {  
5  
6      constructor( nombre, saldo ) {  
7          this.nombre = nombre;  
8          this.saldo = saldo;  
9      }  
10
```

Cualquier **método** agregado a la clase será parte del prototipo

```
    imprimirSaldo() {  
        return `Hola ${this.nombre}, tu saldo es: ${this.saldo}`;  
    }  
  
    retiraSaldo(retiro) {  
        this.saldo -= retiro;  
    }
```



También existe algo llamado las ***propiedades estáticas***, estas no requieren ser instanciadas

```
✓ static bienvenida(){  
  return `Bienvenido al cajero`;  
}  
}
```

Cliente { nombre: 'Juan', saldo: 400 }  
Hola Juan, tu saldo es: 400  
Hola Juan, tu saldo es: 200  
Bienvenido al cajero



Son propiedades que **pertenecen a la clase en sí**, en lugar de a las instancias individuales de la clase. Esto significa ***que todas las instancias de la clase comparten el mismo valor para la propiedad estática.***

# Aquí hay algunos ejemplos de cómo se pueden usar las propiedades estáticas:

- **Definir una configuración predeterminada:** Las propiedades estáticas se pueden usar **para definir una configuración predeterminada para una clase**. Esta configuración se puede anular para instancias individuales de la clase.
- **Crear métodos estáticos:** Las propiedades estáticas se pueden usar **para definir métodos estáticos. Los métodos estáticos son métodos que pertenecen a la clase en sí y no a las instancias individuales** de la clase. Los métodos estáticos se pueden utilizar para realizar tareas que son relevantes para toda la clase, como validar datos o convertir cadenas.

```
7 class Persona {
8     static contador = 0;
9
10    constructor(nombre, edad) {
11        this.nombre = nombre;
12        this.edad = edad;
13
14        Persona.contador++;
15    }
16 }
17
18 const persona1 = new Persona('Juan', 30);
19 const persona2 = new Persona('María', 25);
20
21 console.log(Persona.contador); // Salida: 2
```



```
class Persona {  
  static paisPredeterminado = 'Argentina';  
  
  constructor(nombre, edad, pais) {  
    this.nombre = nombre;  
    this.edad = edad;  
    this.pais = pais || Persona.paisPredeterminado;  
  }  
}
```

```
const persona1 = new Persona('Juan', 30);  
const persona2 = new Persona('María', 25, 'España');
```

```
console.log(persona1.pais); // Salida: Argentina  
console.log(persona2.pais); // Salida: España
```

# En un formulario.....

Se pueden usar las propiedades estáticas para ***crear un validador de formularios***. Las propiedades estáticas se pueden usar para implementar una amplia gama de validaciones y lógica en aplicaciones web.

## ¿ Que podría validar?

Verificar si los valores ingresados por el usuario son válidos. Ej: Validar nombre.



# Beneficios del uso de propiedades estáticas en formularios:

- **Mejora la organización del código:** Al separar la lógica de validación en propiedades estáticas, el código se vuelve más organizado y fácil de leer.
- **Reutilización de código:** Las propiedades estáticas se pueden reutilizar en diferentes partes de la aplicación.
- **Facilita el mantenimiento:** Es más fácil modificar y mantener la lógica de validación si está encapsulada en propiedades estáticas.



# Bibliografía

- JavaScript Bible, Séptima Edición 2010.Dany Goodman, Ed.Hungry Minds
- El gran libro de HTML5, CSS3 y Javascript 2012.J. Gauchat ,Ed. Marcombo
- Manual de JavaScript, 2011.R. Alva, Editorial CEP.
- Algoritmos y Pseudocodigos, 2012 Delgado y Velazques,
- The Art of Computer Programming)**Autor:** Donald Knuth (1968)