# JavaScript Herencia e Interfaces en Orientación a Objetos

## Conclusión 1: Conociendo el problema del cliente

* Recordamos los conceptos de la Orientación a objetos: Clases, métodos y atributos/propiedades
* Buenas prácticas a la hora de declarar atributos.
* Declarar los atributos privados en la medida de lo posible
* Problemas relacionados a la repetición de código

## Conclusión 2: Herencia

* Especialización del código en clases
* Herencia de atributos/propiedades y métodos
* Sobreescritura de métodos
* Compartiendo el código de forma segura.
* Evitar código duplicado usando el concepto de herencia.
* Palabra reservada super
* Métodos y atributos privados y protegidos.

## Conclusión 3: Clases abstractas

* Cómo identificar que tipo de clase está siendo instanciada.
* Proteger nuestro sistema
* Mostrar errores de ejecución para condiciones personalizadas
* Clases abstractas
* Métodos abstractos.

## Conclusión 4: Sistema interno

* Repaso de herencia, encapsulamiento y buenas prácticas en el código.
* Polimorfismo
* Métodos estáticos
* Maneras de tratar objetos de diferentes tipos, de forma similar.

## Conclusión 5: Interfaces y Duck Type

* Operador in
* Operador instanceOf
* Como validar propiedades y tipos para proteger nuestros sistemas
* Interfaces normalizadas
* Duck Type

## Anexos:

* Articulo: <https://ljcl79.medium.com/herencia-en-programación-orientada-a-objetos-370cf3f97402>
* Articulo: <https://ljcl79.medium.com/las-clases-abstractas-qué-son-y-para-qué-sirven-8328b92db680>
* <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/instanceof>
* <https://developer.mozilla.org/es/docs/Web/JavaScript/Reference/Operators/in>
* <https://es.wikipedia.org/wiki/Duck_typing>

## Código:

* Extends: puede dar herencia a un hijo
* Super: se puede buscar la herencia de un padre
* .error

## Vocabulario:

* Herencia:
* Clases abstractas:
* Funciones abstractas
* Métodos estáticos
* Funciones estaticas
* Métodos abstractos:
* Polimorfismo: Polimorfismo es cuando tienes una función, tienes un método o tienes un atributo que se llama de igual forma en diferentes clases, solo que estas clases no están relacionadas entre sí en un esquema de herencia sino que son clases que están separadas, que se pueden gestionar de manera individual y ellas tienen un método nombrado de la misma forma que nos va a permitir ejecutar ese método de diferentes formas.
* Funciones estáticas
* Interfaz:
* Polimorfismos:

## Capturas:

|  |
| --- |
|  |