## TEXT CLASS REVIEW

## TEMAS A TRATAR EN EL CUE

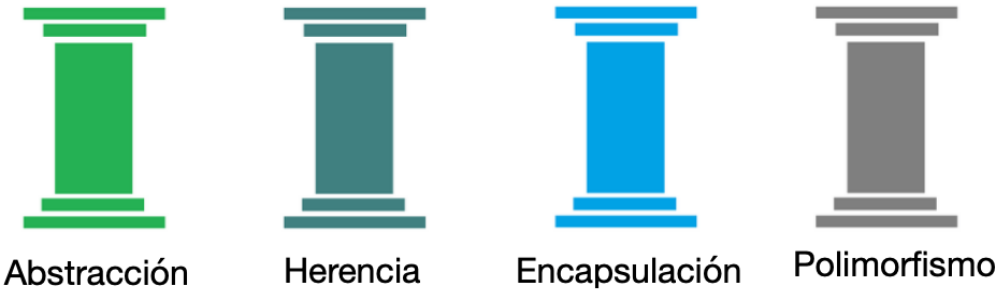
- Funciones de constructores.
- Introducción a programación orientada a objetos con JS.
- Palabra clave This.

### INTRODUCCIÓN A PROGRAMACIÓN ORIENTADA A OBJETOS CON JS

Hasta el momento, hemos adoptado un estilo de programación de procedimientos, el cual utiliza estrictamente funciones y variables para efectuar alguna tarea específica. Existen muchos estilos de programación, y uno de ellos, ésta orientado a objetos, pues se centra en ellos, en vez de funciones. La idea básica de POO, es que se empleen objetos para modelar cosas del mundo real que queremos representar dentro de nuestros programas, y proporcionar una forma simple de acceder a la funcionalidad, que de otra manera sería difícil o imposible de utilizar.

Los objetos pueden contener datos y código relacionados, que representan información sobre lo que se está intentando modelar, y la funcionalidad o el comportamiento que se desee que tenga. Estos datos (incluido las funciones) se pueden almacenar ordenadamente, o ***encapsular*** dentro de un paquete de objetos, lo que facilita la estructura de acceso. Esto es uno, de los cuatro conceptos fundamentales de la programación orientada a objetos, donde también se incluye: la ***abstracción***, la ***herencia*** y el ***polimorfismo***.

# Programación orientada a objetos




**ENCAPSULAR:**

Significa ocultar información o datos. Se refiere a la capacidad del objeto para ejecutar su funcionalidad, sin revelar ningún detalle de ejecución al llamador.

**ABSTRACCIÓN:**

Significa ocultación de implementación. Es una manera de ocultar los detalles de implementación, y mostrar solo las características esenciales a la persona que llama.

**HERENCIA:**

Es un mecanismo que nos permite crear una nueva clase, utilizando la existente. Significa que la clase “hijo” hereda todas las propiedades y comportamientos de la clase “padre/madre”.

**POLIMORFISMO:**

Se refiere a la capacidad de llamar al mismo método en diferentes objetos, y hacer que cada uno responda a su manera.

La programación orientada a objetos tiene varias ventajas sobre la procedimental:

- Proporciona una estructura clara para los programas.
- Ayuda a mantener el código DRY "Don't Repeat Yourself", y hace que éste sea más fácil de mantener, modificar y depurar.
- Hace posible generar aplicaciones reutilizables con menos código, y un tiempo de desarrollo más corto.

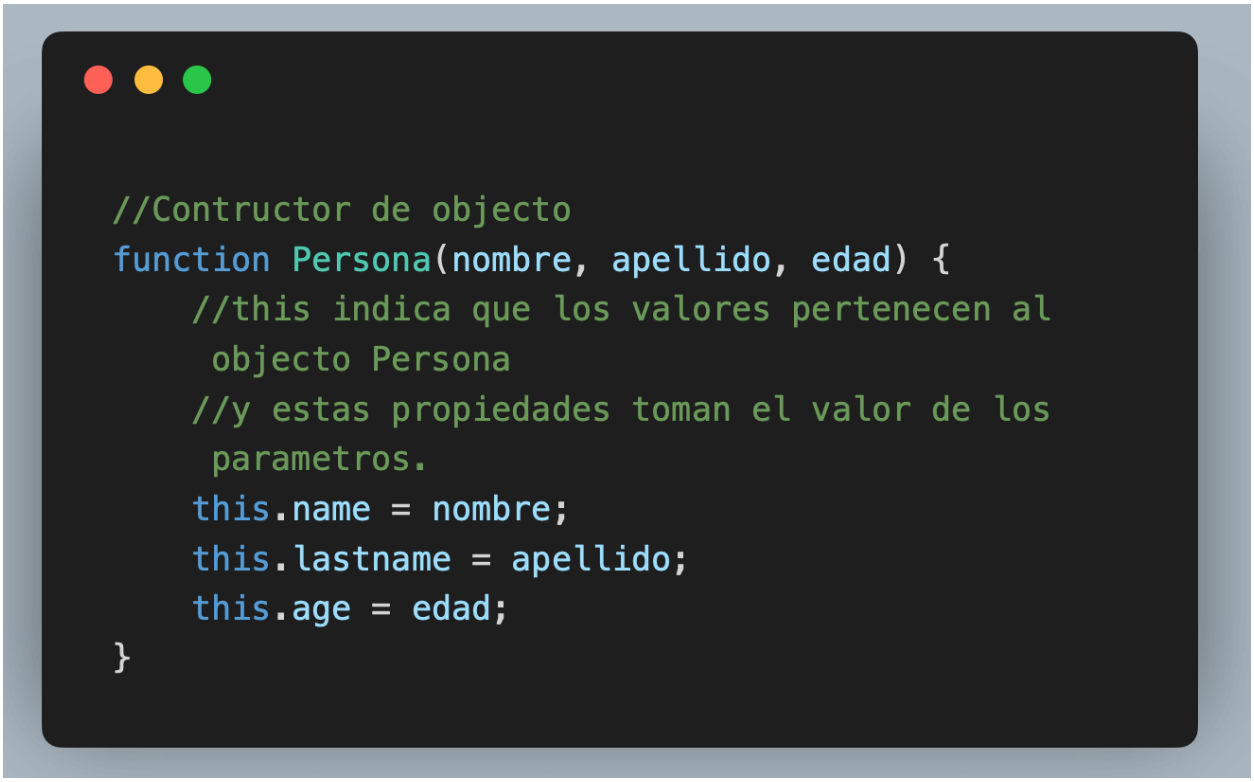
¿Cómo se ve la programación orientada objetos?: los objetos también son variables, y pueden contener muchos valores como **propiedades** (*variables*) y **métodos** (*funciones*). Por ejemplo, si quisiéramos hacer un objeto de tipo persona, pudiésemos añadirle las siguientes propiedades: nombre, apellido, id y oficio. Además, podemos incluir un método que nos permita visualizar el nombre completo de la persona.

La encapsulación de toda esta información en un objeto, se asemeja a lo siguiente:

```
// Objeto Persona
var persona = {
    //Propiedades
    nombre: "John",
    apellido: "Doe",
    oficio: "programador",
    id: 126,
    //Métodos
    nombreCompleto: function () {
        return `${this.nombre} ${this.lastName}`;
    }
};
```

**FUNCIONES DE CONSTRUCTORES**

Algunas veces necesitamos un "plano", para crear muchos objetos del mismo "tipo". La forma de crear un "tipo de objeto", es utilizar una función de constructor de objeto, cuya estructura general se visualiza en la siguiente imagen:



```
//Contructor de objecto
function Persona(nombre, apellido, edad) {
    //this indica que los valores pertenecen al
     objecto Persona
    //y estas propiedades toman el valor de los
     parametros.
    this.name = nombre;
    this.lastname = apellido;
    this.age = edad;
}
```

**PALABRA CLAVE THIS**

Como se puede notar, una función de constructor se vale de la palabra clave **`this`**. En JavaScript, lo que se llama this (*esto*), es el objeto que "posee" el código. Su valor, cuando se usa en un objeto, es el objeto mismo. En una función constructora, this no tiene valor, pues es un sustituto del nuevo objeto; y el valor de éste, se convertirá en el nuevo objeto cuando sea creado.