PROGRAMACIÓN AVANZADA EN JAVASCRIPT

Módulo 3

OTEC PLATAFORMA 5 CHILE

MÓDULO 3

CLASE 1

Introducción

Paradigmas de programación

Programación orientada a Objetos

Clases

Sintaxis

Datos de Objetos

Introducción

Con este nuevo comienzo de Módulo, indagaremos sobre temas más importantes y complejos de JavaScript, los cuales nos darán la posibilidad de realizar proyectos y trabajos más sofisticados.
Serán proyectos donde usted podrá aplicar todo lo aprendido y así demostrar todo su esfuerzo y dedicación en empeñó en un corto período de tiempo.

Comenzaremos a trabajar hoy con la Programación Orientada a Objetos, un tema fundamental en nuestro querido lenguaje.

Iremos paso a paso porque son conceptos nuevo y abstractos, es por eso que nos llevará 2 días de clases.

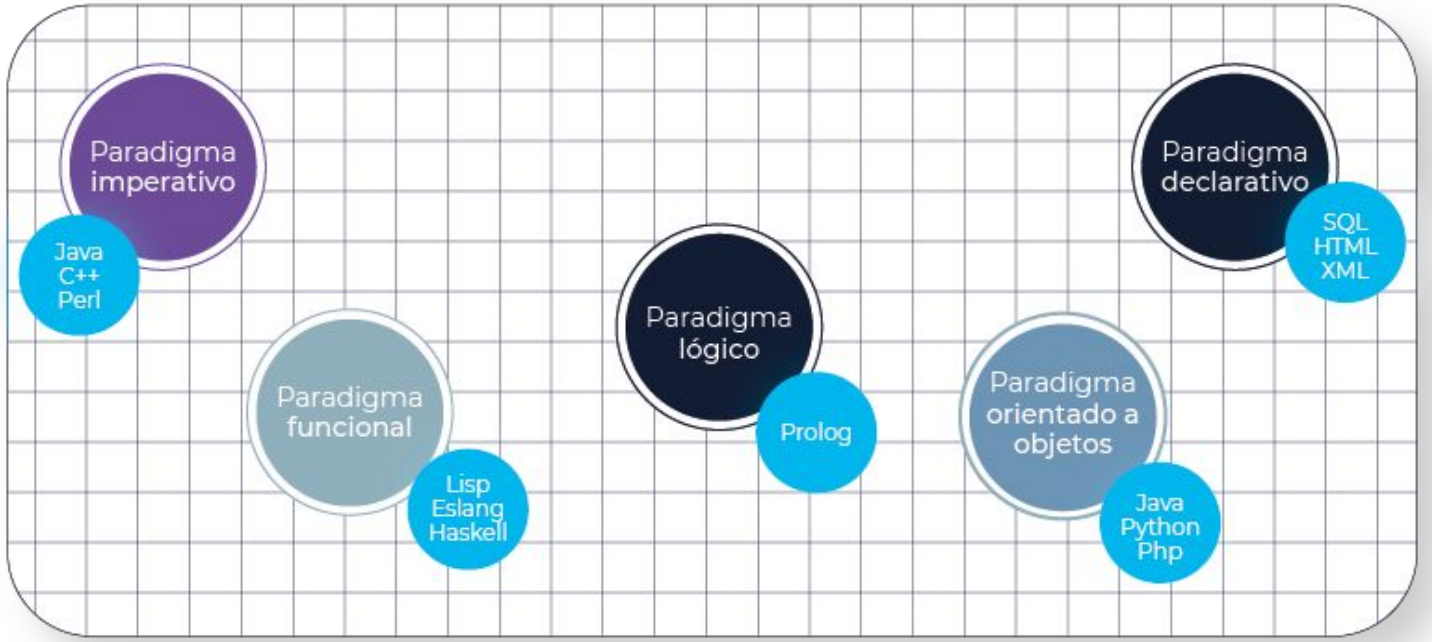
Recuerde que es importantísimo que IGNORE LA COMPLEJIDAD y solo se concentre en lo que se está explicando.

¿Qué son los *Paradigmas de Programación*?

Los paradigmas de programación son las diferentes maneras para resolver un problema.

Hay cinco paradigmas distintos:

1. **Imperativo**
2. **Funcional**
3. **Lógico**
4. **Orientado a Objetos**
5. **Declarativo**



Hasta el momento, trabajamos con JavaScript bajo el paradigma Imperativo, resolviendo los problemas de manera secuencial. Es decir, escribimos instrucciones, paso a paso, y utilizamos ciclos y condiciones para llegar a una solución.

La mayoría de las personas que aprenden a programar se inician bajo este paradigma. Sin embargo, JavaScript se conoce como un lenguaje *multiparadigma*.

En este módulo abordaremos JavaScript desde un nuevo paradigma: el Paradigma Orientado a Objetos.

¿Qué es la *POO*?

La Programación Orientada a Objetos (POO) es una manera de programar que facilita el desarrollo de sistemas complejos y de gran tamaño.

Esta técnica nació a fines de los años 80 y se volvió popular rápidamente porque es más cercana a cómo expresamos las cosas en la vida real: dividiendo los componentes de un programa en Objetos que poseen datos (Propiedades) y comportamientos (Métodos) que se comunican entre sí.

Dentro de sus ventajas, podemos destacar:

- **Fácil de modificar.**
- **Escalabilidad.**
- **Reusabilidad.**
- **Mantenibilidad.**

Clases

Las Clases son un molde para crear Objetos, y los Objetos pueden instanciarse si cumplen con las características de esa Clase.

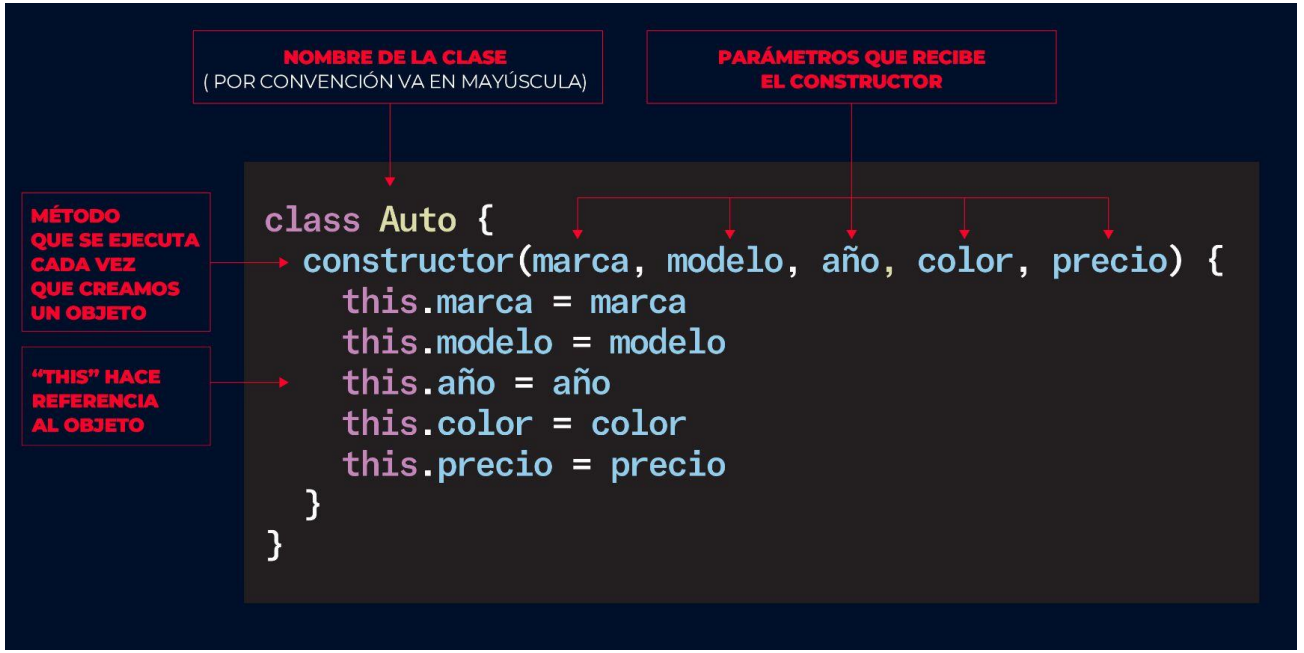
A la hora de armar una Clase debemos tener en cuenta cuáles son las características que se repiten entre los distintos Objetos.

Importante:

Por convención, el nombre de una Clase va siempre en mayúscula.

Sintaxis

- constructor(): Se trata de un método especial para la creación e inicialización de un Objeto. Puede haber solo un constructor por Clase.
- Propiedades: Las Propiedades son similares a las Variables, porque nos permiten definir las características que tendrán los Objetos (posteriormente instanciados).
- this: Hace referencia al atributo propio de la Clase. Esto nos permite diferenciar el valor que llega como Parámetro del Método constructor, de la propiedad en sí.



ACLARACIÓN

La palabra *marca* después del = toma el valor que recibe por *parámetro en el constructor*. Ese valor se asigna a la Propiedad marca, que es propia del Objeto. La keyword this sirve para diferenciar una de la otra.

Cómo acceder a los datos de nuestros Objetos

Para acceder a los valores de nuestros Objetos podemos usar tanto Dot como Bracket notation.

Por ejemplo, para acceder al valor de la Propiedad color de cada auto, deberíamos respetar la siguiente sintaxis:

```
autoIdeal.color
```

```
autoIdeal["color"]
```

En ambos casos, obtenemos el valor `verde`.

Cómo *agregar propiedades* a nuestros Objetos

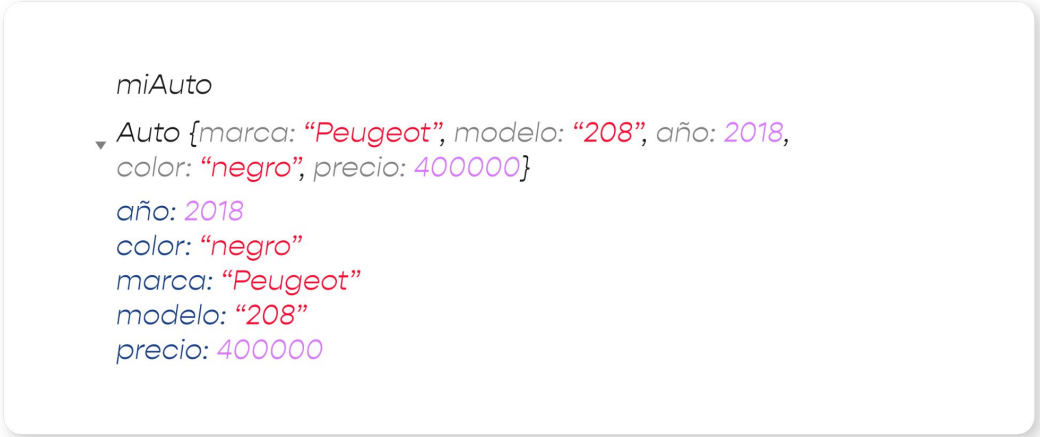
Para agregar Propiedades a nuestros Objetos también podemos usar Dot o Bracket notation.

Por ejemplo, si quisiéramos agregar una Propiedad que tenga un valor booleano (`tieneVTV`) podríamos escribir:

```
autoIdeal.tieneVTV = false
```

De esta manera, modificamos el Objeto `autoIdeal`:

Sin embargo, el Objeto `miAuto` permanece igual:



RECUERDE

Si bien estos Objetos comparten la misma Clase, son independientes uno del otro.

Cómo *eliminar propiedades* de nuestros Objetos

Para eliminar una Propiedad debemos usar el operador `delete`.

Veamos un ejemplo. Si quisiéramos eliminar la Propiedad "`color`" del Objeto `miAuto`, podríamos escribir:

```
delete miAuto.color
```

Ahora, pruébelo en su consola y mire el código: a través del operador delete, eliminamos la Propiedad "color".

MÓDULO 3

GRACIAS

Aquí finaliza la clase n°1 del módulo 3

OTEC PLATAFORMA 5 CHILE