PROGRAMACIÓN AVANZADA EN JAVASCRIPT

Módulo 3

OTEC PLATAFORMA 5 CHILE

MÓDULO 3

CLASE 2

Introducción

Hoy continuaremos con la Programación Orientada a Objetos.
Como verán es un tema extenso y debe comprenderse paso a paso para evitar confusiones.

Recuerde que si tiene dudas o no comprende, es necesario que las comunique para repasar el tema en cuestión. Ya que de aquí en más todos los temas son abarcativos.

Herencia

La herencia es uno de los conceptos más importantes de la Programación Orientada a Objetos.

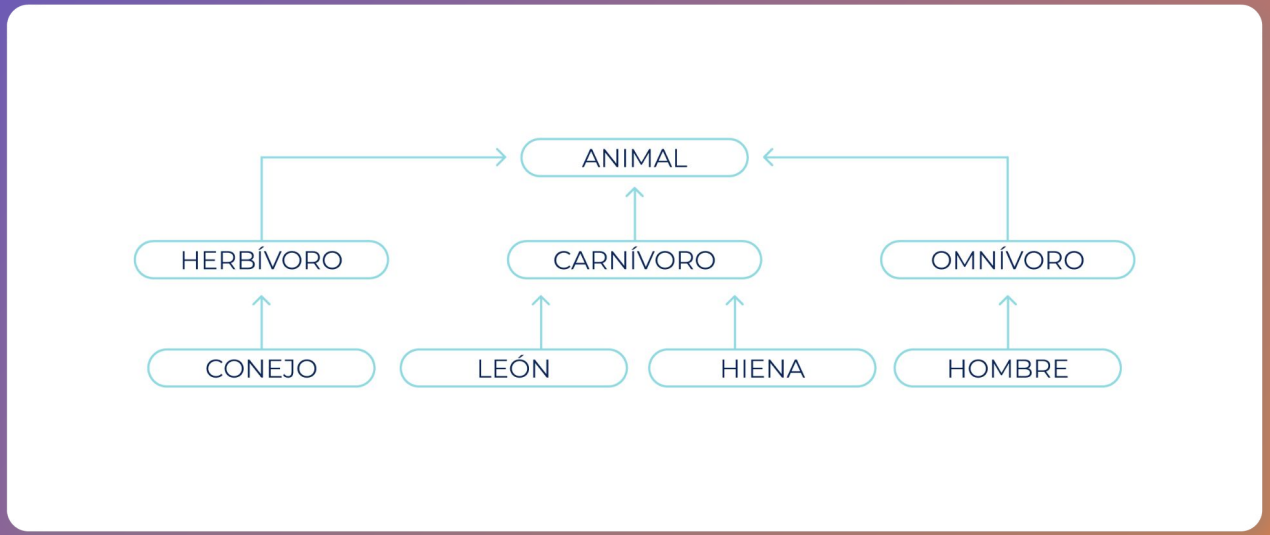
Una Clase puede heredar las características (Propiedades y Métodos) de otra, que estén definidos en una superclase también llamada "Clase Padre".

Es decir, gracias a la herencia podemos definir nuevas Clases basadas en otras preexistentes, generando una jerarquía. De esta manera, reutilizamos el código. Esto es muy útil, ya que la subclase tiene todos los atributos y métodos definidos en la superclase y, a la vez, está abierta a redefinirlos.

Por ejemplo, supongamos que tenemos una `superclase mamifero` cuyo método `reproducción` está definido por defecto como "gestación". La `subclase ornitorrinco` podría redefinir este método, implementando la reproducción por "huevos".

Importante: Si bien una Clase puede heredar las características de otra, también podemos añadir nuevas Propiedades y Métodos para distinguirlas entre sí.

Veamos otro ejemplo.



Generalización Y Especificidad

Si pensamos en la relación de herencia como un árbol o pirámide, cuanto más alto será más genérico y cuanto más bajo será más específico. Es decir, los conceptos más importantes estarán en la copa del árbol.

Objeto *Date()*

El Objeto Date() nos permite trabajar con fechas y horas proporcionadas por nuestro navegador.

Creación de Objetos Date()

Podemos instanciar distintos Objetos Date() dependiendo de la cantidad de argumentos que le proporcionamos al constructor.

Por ejemplo:

```
let ahora = new Date(); // Devuelve día y hora actual
let navidad = new Date(2022, 11, 25); // Navidad del año 2021
let casiNavidad = new Date(2022, 11, 24, 23, 59, 59); // Un segundo
antes de navidad
```

Objeto *Date()*

En el caso de que no hubiera uno, el constructor instanciará un Objeto con la hora local y fecha de hoy.

Por ejemplo:

```
let date = new Date();
console.log(date)
> Tue Nov 02 2021 00:51:52 GTM-0300
```

Importante: el Objeto Date() es estático. Si volvemos a llamar a la Variable asignada date, su valor será el mismo (por más que haya transcurrido tiempo).

Métodos del Objeto Date()

El Objeto Date() puede invocar diferentes métodos. Veamos algunos de ellos:

getFullYear()

El método getFullYear() nos devuelve el año actual.

```
console.log(date.getFullYear())
> 2021
```

getMonth()

El método getMonth() nos devuelve el valor numérico que corresponde al mes actual.

***Importante:* Tené en cuenta que el primer mes (enero) posee el número 0 (cero), por lo que el mes de diciembre tiene como valor el número 11 (once).**

```
console.log(date.getMonth())
> 9
```

Métodos del Objeto Date()

getDate()

El método getDate() nos devuelve el valor numérico que corresponde al día del mes en curso.

```
console.log(date.getDate())
```

```
> 12
```

getDay()

El método getDay() nos devuelve un valor numérico que corresponde al día de la semana.

Importante: **Tené en cuenta que se toma como el primer día de la semana al domingo y se le asigna el valor 0 (cero).**

```
console.log(date.getDay())
```

```
> 1
```

Otros conceptos de la Programación Orientada a Objetos

Además del concepto de Herencia, existen otros que son centrales para POO: Modularidad, Abstracción, Encapsulamiento y Polimorfismo. En esta sección desarrollaremos brevemente cada una.

Supongamos que tuviéramos un concesionario de autos. Podemos ingresar nuevos autos, venderlos y obtener una lista de autos disponibles por marca. Además, cada unidad tiene en común ciertas características (marca, año de fabricación, etc.) y acciones (acelerar). Sin embargo, también pueden tener diferencias: por ejemplo, un auto puede ser manual mientras otro es automático.

Otros conceptos de la Programación Orientada a Objetos

Modularidad

La POO permite dividir nuestra aplicación en partes más pequeñas e independientes (módulos).

Volviendo al ejemplo, esto nos permitiría crear una Clase Concesionario y una Clase Auto. De esta forma, podríamos crear Subclases como AutoNaftero y AutoElectrico.

Abstracción

La abstracción implica captar las características esenciales de un Objeto (por ejemplo: su comportamiento). Es decir, aísla un elemento de su contexto o del resto de los elementos.

En nuestro ejemplo, el concesionario conoce las características de los autos, pero puede abstraerse de los detalles de su funcionamiento interno.

Otros conceptos de la Programación Orientada a Objetos

Encapsulamiento

El encapsulamiento permite restringir y controlar el uso de una Clase, ya que consiste en el ocultamiento de las Propiedades y acciones de un Objeto. De esta manera, se pueden cambiar mediante las operaciones definidas para ese Objeto.
Al controlar, damos ciertos mecanismos para modificar el estado de una Clase.

En nuestro ejemplo, el concesionario tiene atributos públicos (por ejemplo: su dirección y horario de atención) y privados (como el balance del último mes).

Polimorfismo

El polimorfismo nos permite asociar diferentes estrategias de resolución para un mismo comportamiento.

En nuestro ejemplo, todos los autos aceleran. Sin embargo, algunos lo hacen a combustión y otros de manera eléctrica.

¿Qué significa restringir y controlar?

Al restringir un comportamiento o atributo privado de una Clase, indicamos que no podrá ser accedido por otras.

MÓDULO 3

GRACIAS

Aquí finaliza la clase n°2 del módulo 3

OTEC PLATAFORMA 5 CHILE