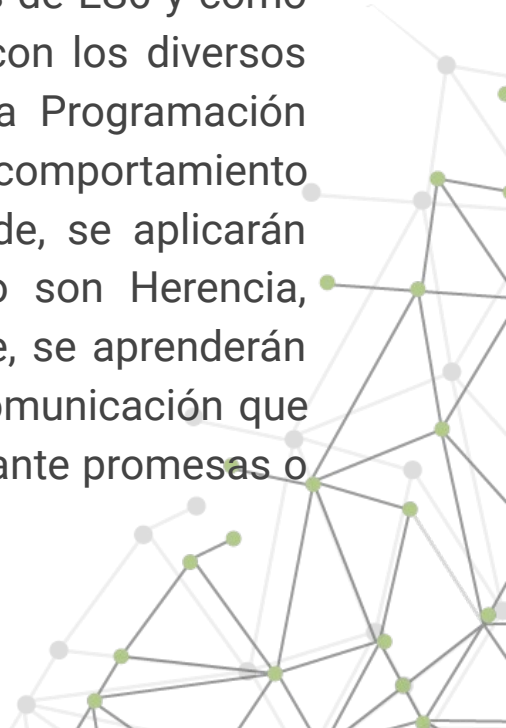
ES6+ y POO

Introducción a la programación orientada a objetos

{desafío}
latam_

¿Qué aprenderemos en este módulo?

A lo largo de este módulo, se aprenderán las nuevas funcionalidades de ES6 y cómo crear un flujo de trabajo eficiente para asegurar la compatibilidad con los diversos ecosistemas de la web. Además, se conocerá el paradigma de la Programación Orientada a Objetos, el cual permite simular de una mejor manera el comportamiento de las cosas del mundo real al nivel de la programación, por ende, se aplicarán conceptos claves de la Programación Orientada a Objetos como son Herencia, Encapsulamiento, Polimorfismo, Abstracción, entre otros. Finalmente, se aprenderán las bases de la comunicación cliente-servidor y los protocolos de comunicación que se emplean para poder solicitar información a una APIs, ya sea mediante promesas o funciones asíncronas que servirán como base para la programación.



Utilizar los conceptos fundamentales de la programación orientada a objetos acorde al lenguaje Javascript para resolver un problema simple.

- Unidad 1:
  ES6+ y POO
- Unidad 2:
  Herencia
- Unidad 2:
  Callbacks y APIs

Te encuentras aquí

{desafío}
latam_

¿Qué aprenderás en esta sesión?

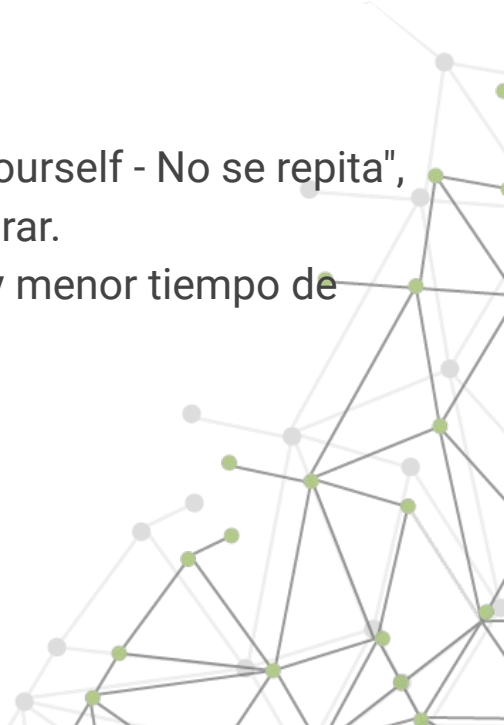
- *Describe las características del paradigma de la programación orientada a objetos para la programación web.*

Podrías describir con tus palabras, ¿Qué es la POO? y ¿Cuáles crees que son sus principales ventajas?

Programación orientada a objetos

Algunas de sus principales ventajas son:

- Proporciona una estructura clara para los programas.
- Ayuda a mantener el código bajo el concepto DRY "Don't repeat yourself - No se repita", y hace que el código sea más fácil de mantener, modificar y depurar.
- Hace posible crear aplicaciones reutilizables con menos código y menor tiempo de desarrollo.



/* Programación Orientada a Objetos */

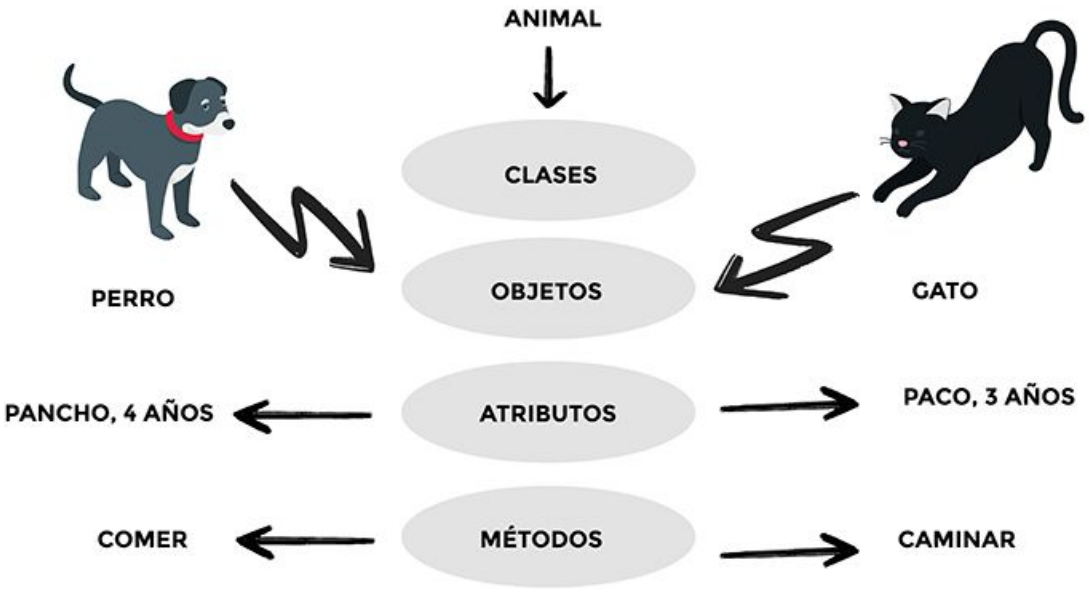
{desafío}
latam_

Programación Orientada a Objetos

En la Programación Orientada a Objetos (POO) se busca describir algo del mundo real de manera abstracta (es decir, en el código). Todo es un objeto, y cualquier acción que necesitemos realizar en los datos (lógica, modificaciones, entre otros) se escriben como métodos de un objeto. En otras palabras, la Programación Orientada a Objetos puede mejorar la capacidad del desarrollador para crear prototipos de software rápidamente, ampliar la funcionalidad existente, refactorizar el código y mantenerlo a medida que se desarrolla.

Para comprender mejor en qué consiste este paradigma y cómo se relaciona con JavaScript, a lo largo de la unidad veremos algunas terminologías básicas, como lo son: atributos, métodos, abstracción, encapsulamiento, entre otros.

Programación Orientada a Objetos



{desafío} latam_

profile

Fuente: profile.es/blog/

/* Objetos y propiedades */

Objetos y propiedades

Un objeto es una representación de algo en el mundo real. Puede ser un auto, una casa, una bicicleta, un conjunto de números, puede ser una página web o incluso un carro de compras. Un objeto puede estar compuesto de más objetos. Así mismo, los objetos también tienen propiedades o datos, también conocidos como atributos, como por ejemplo, color, tamaño, nombre, edad, valores o cualquier otra cosa que imagines. Mientras que las funcionalidades, también conocidas como métodos tienden a modificar y/o utilizar estos estados o atributos, ya que son una capacidad del objeto en sí, como caminar. Por ejemplo cuando un auto se mueve, éste cambia su posición.

Demostración - Carro

Demostración

La instrucción new es la que nos permite crear un nuevo objeto, a esto le llamaremos instanciar (simplemente es crear un nuevo objeto partiendo de otro). Luego sobre este objeto que acabamos de instanciar definiremos diversas propiedades, por ejemplo, el modelo y el año.

```
var carro = new Object();
carro.marca = 'Toyota';
carro.modelo = 'Corolla';
carro.fecha = 2020;
```

Demostración

Podemos observar el objeto creado utilizando console.log(carro) y observando los valores dentro del inspector de elementos.

```
Object { marca: "Toyota", modelo: "Corolla", fecha: 2020 }
```

Ejercicio guiado: Creando objetos

{desafío} latam_

Ejercicio guiado

Creando objetos

Crear un rectángulo con las propiedades de largo y ancho, dando la libertad para escoger los valores a asignar en cada propiedad.

Paso 1: En la consola del navegador debemos primeramente crear el nuevo objeto denominado rectángulo:

```
var rectángulo = new Object();
```

Ejercicio guiado

Creando objetos

Paso 2: En la misma consola del navegador web, vamos a crear las propiedades de ancho y de largo para el objeto “rectangulo”:

```
rectangulo.ancho = 10;
rectangulo.largo = 5;
```

Paso 3: Finalmente, podemos observar el valor utilizando console.log(rectangulo); arrojando como resultado sobre la misma consola del navegador web lo siguiente:

```
Object { ancho: 10, largo: 5 }
```

Ejercicio guiado

Creando objetos

Así como existe la metodología para crear objetos, vista anteriormente, también existe otra forma de crearlos, la cual se logra implementando la siguiente sintaxis:

```
cuadrado = {ancho: 10, largo: 5}
```

Utilizando la instrucción “console.log” podemos observar que obtenemos el mismo resultado. Por consiguiente, esta notación recibe el nombre de notación literal.

Por otra parte, también podemos mostrar o modificar el valor de estas propiedades de la siguiente forma:

```
console.log(cuadrado.ancho);
cuadrado.ancho = 7;
cuadrado['ancho'] = 8;
```

Ejercicio guiado: Creando objetos con métodos

{desafío} latam_

Ejercicio guiado

Creando objetos con métodos

Los métodos sirven para que los objetos realicen acciones. Veamos un ejercicio sencillo de cómo crear un método y utilizarlo. En este caso, se debe crear un objeto denominado “vaca” y luego crear un método “sonido” que permita mostrar por la consola del navegador web el sonido que emite la vaca. Para esto, sigamos los siguientes pasos:

Paso 1: En la consola de tu navegador web, crea el objeto con el nombre de “vaca” como se muestra a continuación:

```
var vaca = new Object();
```

Ejercicio guiado

Creando objetos con métodos

Paso 2: Agregar el método al objeto vaca creado anteriormente, el método debe llevar el nombre del sonido y deberá mostrar por la consola del navegador web el sonido que hace una vaca “moo”.

```
vaca.sonido = function () {
     console.log("moo");
}
```

Paso 3: Utilizar el método realizando el llamado al objeto, indicando el método que deseamos implementar, en este caso, “sonido”. Recuerda que esto se puede lograr mediante la convención del punto, es decir, agregar el objeto y seguidamente el punto con el nombre del método que deseamos invocar. Finalmente, después de ejecutar el método requerido, podremos observar el texto "moo" en la consola del navegador web.

```
vaca.sonido();
```

Ejercicio guiado

Creando objetos con métodos

Ahora bien, también es posible agregar un método a un objeto utilizando la notación literal.

```
perro = {
   hablar: function () {
     console.log('guau');
   }
};
perro.hablar();
```

Como se puede observar en el ejemplo anterior, mediante la notación literal se puede agregar cualquier método a un objeto, y posteriormente utilizarlo realizando el llamado convencional mediante el punto, para indicar el método al cual queremos acceder.

Ejercicio guiado: Métodos que utilizan propiedades (operador this)

{desafío} latam_

Ejercicio guiado

Métodos que utilizan propiedades (operador this)

Es muy frecuente cuando trabajamos con objetos, que un método deba modificar o utilizar uno de los estados o atributos de un objeto, para lograr esto tendremos que utilizar la instrucción this. Para ilustrar esto, en el siguiente ejercicio se solicita crear un objeto con el nombre de “persona” en la consola del navegador web, luego, agregar la propiedad o atributo “nombre” con el valor de “Camila”.

Paso 1: Crear la variable con el nombre de “persona” y luego se debe asignar un nuevo objeto mediante la instrucción “new Object()”, el cual permitirá inicializar un nuevo objeto en la variable indicada, como se muestra a continuación:

```
var persona = new Object();
persona.nombre = "Camila";
```

Ejercicio guiado

Métodos que utilizan propiedades (operador this)

Paso 2: Agregar un método al objeto creado anteriormente, en este caso se debe crear el método “saludar”, y agrega una función que permita mostrar un saludo por la consola del navegador web, indicando el valor asignado a la propiedad “nombre”, aquí es donde se debe implementar el operador this:

```
persona.saludar = function(){
  console.log("Hola soy " +
this.nombre);
};
```

Paso 3: Realizar el llamado al método “saludar” y observa el resultado en la consola del navegador web:

```
persona.saludar();
```

{desafío} latam_

Ejercicio guiado

Métodos que utilizan propiedades (operador this)

El resultado en la consola del navegador web será:

```
Hola soy Camila
```

Todo bien hasta el momento, pero qué sucedería si en el código anterior no utilizamos la palabra reservada this, y ejecutamos el código nuevamente en la consola del navegador web ¿Qué pasaría?

```
Uncaught ReferenceError: nombre is not defined
```

¿Qué fue lo que sucedió? Si observamos el error que obtuvimos veremos que nos muestra el mensaje "nombre is not defined", esto se debe a que si no especificamos con this que nombre es una propiedad, dentro del objeto buscará una variable llamada nombre. Por consiguiente, para referirnos a una propiedad del objeto, tenemos que utilizar this.

/* La función constructora */

La función constructora

Existen mejores formas para crear objetos, una muy utilizada es la función constructora, la cual, es una función normal y corriente que se utiliza para definir una especie de plantilla para nuestros objetos personalizados, la ventaja de esta forma es que nos permite crear múltiples objetos del mismo tipo.

Ejercicio guiado: Creación de objetos a partir de una función constructora

{desafío}
latam_

Ejercicio guiado

Creación de objetos a partir de una función constructora

Paso 1: Crear en la consola del navegador web una función constructora denominada “Estudiante”, utiliza el parámetro “nombre” y asigna el parámetro a una propiedad que llevará el mismo indicativo, pero recuerda que se debe implementar la palabra reservada “this”.

```
function Estudiante(nombre) { // Función constructora
  this.nombre = nombre;
};
```

Ejercicio guiado

Creación de objetos a partir de una función constructora

Paso 2: Instanciar el objeto con nuevos valores, en este caso con distintos nombres, para ello se debe implementar la instrucción “new” seguidamente del nombre de la función constructora, y como argumento el valor que deseamos que almacene el objeto. Quedando el código de la siguiente manera:

```
var c1 = new Estudiante('Javiera');
var c2 = new Estudiante('Francisco');
var c3 = new Estudiante('Diana');
```

Paso 3: Mostrar en la consola del navegador web el valor del nombre de cada uno de los objetos creados:

```
console.log(c1);
console.log(c2);
console.log(c3);
```

Ejercicio guiado

Creación de objetos a partir de una función constructora

Si observamos ahora en el inspector de elementos veremos lo siguiente:

```
Object { nombre: "Javiera" }
Object { nombre: "Francisco" }
Object { nombre: "Diana" }
```

Esta forma de crear objetos nos será útil cuando queremos construir múltiples objetos.

Ahora bien, para agregar un método a un objeto en específico ocuparemos la misma fórmula de antes, es decir, a la instancia creada se le asigna el nombre del método que deseamos crear separados por un punto, seguidamente se le agrega una función anónima con el código correspondiente que necesite el método en sí.

Ejercicio guiado

Creación de objetos a partir de una función constructora

Por ejemplo, si a la función constructora creada anteriormente y a la instancia denominada “c1”, le creamos un método denominado “saludar”, el cual, muestre por consola el siguiente mensaje: “Hola soy Javiera”. Se debería entonces implementar el siguiente código:

```
c1.saludar = function () {
  console.log("Hola soy " +
this.nombre);
};
```

Luego, podemos hacer el llamado al método saludar, agregando los dos paréntesis al final del nombre del método para que pueda ejecutar la función, como se muestra a continuación:

```
c1.saludar();
```

Ejercicio guiado

Creación de objetos a partir de una función constructora

Al ejecutar todo el código anterior en nuestra consola del navegador web, el resultado será:

```
Hola soy Javiera
```

Ahora, implementemos el método saludar con la misma funcionalidad realizada anteriormente, esta vez para la instancia creada en la variable “c2”. Por consiguiente, utilizando la variable “c2”, luego con el punto más el nombre del método y la función que este almacenará, se muestra por consola el siguiente mensaje: “Hola soy” más el valor de la variable denominada “nombre”. Quedando el código:

```
c2.saludar = function(){
  console.log("Hola soy " + this.nombre);
}
```

Ejercicio guiado

Creación de objetos a partir de una función constructora

Posteriormente, realizamos el llamado a esa función dentro del objeto y automáticamente podremos observar el resultado en la consola del navegador:

```
c2.saludar();
// resultado
Hola soy Francisco
```

Un detalle importante que siempre es bueno recordar, por convención los nombres que utilizaremos para las funciones constructoras serán con la primera letra mayúscula, de esta forma las diferenciaremos de otros tipos de funciones.

/* Prototipos */

Prototipos

- Permiten agregar una propiedad o diversas funcionalidades a múltiples objetos.
- Mecanismo mediante el cual los objetos en JavaScript heredan características entre sí.
- Los objetos en JavaScript se construyen en base a prototipos.
- Los métodos y propiedades para un objeto son definidos en la propiedad prototype, la cual reside en la función constructora del objeto.

¿Qué ventajas tiene la función constructora en la creación de objetos personalizados?

¿Por qué es importante tener precaución al modificar masivamente las propiedades de todos los objetos?

Resumiendo

- En la Programación Orientada a Objetos (POO) se busca describir algo del mundo real de manera abstracta (es decir, en el código).
- Todo es un objeto, y cualquier acción que necesitemos realizar en los datos (lógica, modificaciones, entre otros) se escriben como métodos de un objeto.
- La función constructora, la cual, es una función normal y corriente que se utiliza para definir una especie de plantilla para nuestros objetos personalizados.
- Los objetos en JavaScript se construyen en base a prototipos.

Próxima sesión...

- *Reconoce los pilares de la programación orientada a objetos para el desarrollo de una pieza de software.*

{desafío}
latam_

Academia de
talentos digitales