Educa2 – Clases y cursos de matemáticas y Programación

Curso de Programación en JavaScript

Preparador: Alexanyer Naranjo

Guía Teórica N°7

Introducción a la Programación Orientada a Objetos (POO)

JavaScript es un lenguaje de programación orientado a prototipos (aunque haremos énfasis en la orientación a objetos). Hasta ahora, hemos utilizado una serie de clases incorporadas para mostrar ejemplos de datos y estructuras de control. Una de las características más poderosas en un lenguaje de programación orientado a objetos es la capacidad de permitir a un programador (solucionador de problemas) crear nuevas clases que modelen los datos necesarios para resolver el problema.

Algunas particularidades de POO en JavaScript son las siguientes:

- Todo es un objeto, incluyendo los tipos y clases.
- No existen métodos ni atributos privados.
- Los atributos pueden ser modificados directamente.
- Permite la sobrecarga de operadores.
- Permite la creación de nuevos tipos de datos.

A continuación, se procede a definir algunos conceptos necesarios para entender la POO:

Objeto: Los ***objetos*** son abstracción de JavaScript para data. Toda la data en un programa JavaScript es representado por objectos o por relaciones entre objectos. (En cierto sentido, y en el código modelo de Von Neumann de una “computadora almacenada del programa” también es un código representado por los objetos.)

Cada objeto tiene una identidad, un tipo y un valor. Una identidad de objecto nunca cambia una vez es creada; usted puede pensar eso como la dirección de objeto en memoria.

El ***tipo*** de un objeto también es inmutable. El tipo de un objeto determina las operaciones que admite el objeto (por ejemplo, “¿tiene una longitud?”) Y también define los valores posibles para los objetos de ese tipo. La función ***typeof()*** devuelve el tipo de un objeto (que es un objeto en sí mismo). El ***valor de algunos objetos puede cambiar***. Se dice que los objetos cuyo valor puede cambiar son **mutables**; los objetos cuyo valor no se puede cambiar una vez que se crean se llaman **immutable**.

Los objetos son la clave para entender la *POO*. Si mira a nuestro alrededor encontrará un sin fin de objetos de la vida real: perro, escritorio, televisor, bicicleta, entre otros.

En JavaScript puede definir una clase con la palabra reservada **class**, de la siguiente forma:

```
class Persona {
    //pass
}
```

En el ejemplo anterior, el nombre de la clase es Persona y dentro del bloque de código usa la sentencia ***//pass***. Aunque no es requerido por el intérprete, los nombres de las clases se escriben por convención capitalizadas y en singular. Las clases pueden (y siempre deberían) tener comentarios.

Atributos: Los atributos o propiedades de los objetos son las características que puede tener un objeto, como el color. Si el objeto es Persona, los atributos podrían ser: id, nombre, apellido, sexo, entre otros.

Los atributos describen el estado de un objeto. Pueden ser de cualquier tipo de dato (incluso un atributo puede ser un Arreglo o un Objeto).

Métodos: Los métodos describen el comportamiento de los objetos de una clase. Estos representan las operaciones que se pueden realizar con los objetos de la clase.

Se definen de la misma forma que las funciones normales, pero deben declararse dentro de la clase.

- La única diferencia sintáctica entre la definición de un método y la definición de una función es que no hay necesidad de utilizar la palabra reservada **function** para la declaración del método de clase.

Clases: Las clases definen las características del objeto.

Con todos los conceptos anteriores explicados, se puede decir que una clase es una plantilla genérica de un objeto. La clase proporciona variables iniciales de estado (donde se guardan los atributos) e implementaciones de comportamiento (métodos) necesarias para crear nuevos objetos, son los modelos sobre los cuáles serán construidos.

Instancias: Ya sabe que una clase es una estructura general del objeto. Por ejemplo, puede decir que la clase Persona necesita tener un id, un nombre, un apellido y un sexo, pero no va a decir cuál es id, nombre, apellido y sexo, es aquí donde entran las instancias.

Una instancia es una copia específica de la clase con todo su contenido.

Métodos constructor(): Es un método especial, el cual se ejecuta al momento de instanciar un objeto. Los argumentos que se utilizan en la definición de constructor() corresponden a los parámetros que se deben ingresar al instanciar un objeto.

Veamos un ejemplo sencillo de la definición de una clase en JavaScript.

```
class Persona {
    //Método constructor
    constructor (nombre, edad, genero) {
        //Atributos de la clase Persona
        this.nombre = nombre
        this.edad = edad
        this.genero = genero
    }
    //Método saludo() de la clase Persona
    saludo () {
        console.log ("Hola a todos, mi nombre es "+this.nombre)
    }
}

//Instancia de la clase Persona. Se utiliza la palabra clave new
let Yo = new Persona('Alexanyer',21,'Masculino')
//Invocación del método saludo() de la clase Persona
Yo.saludo()
```

Herencia: La herencia es una de las premisas y técnicas de la POO la cual permite a los programadores crear una clase general primero y luego más tarde crear clases más especializadas que re-utilicen código de la clase general. La herencia también le permite escribir un código más limpio y legible.

Herencia Simple: La herencia simple se apoya en el uso de clase padre para compartir sus atributos y comportamientos con otras clases derivadas.

Veamos un ejemplo del uso de la Herencia simple.

```
class Persona {
    //Método constructor
    constructor (nombre, edad, genero) {
        //Atributos de la clase Persona
        this.nombre = nombre
        this.edad = edad
        this.genero = genero
    }
    //Método saludo() de la clase Persona
    saludo () {
        console.log ("Hola a todos, mi nombre es "+this.nombre)
    }
}

class Empleado extends Persona {
    //Método constructor de la clase Empleado
    constructor (nombre, edad, genero, puesto, sueldo) {
        //Invocamos el método constructor de la clase padre (Persona)
        super (nombre, edad, genero)
        //Inicializamos los valores de los atributos de la clase Empleado
        this.puesto = puesto
        this.sueldo = sueldo
    }
    //Método saludoEmpleado() de la clase Empleado
    //Observe que hacemos uso del atributo nombre de la clase Persona
    saludoEmpleado () {
        console.log("Soy un empleado y mi nombre es: "+this.nombre)
    }
}
//Instancia de la clase Empleado
Yo = new Empleado('Alexanyer',21,'Masculino','Gerente',5000.00)
//Invocación del método saludo() de la clase Persona
Yo.saludo()
//Invocación del método saludoEmpleado() de la clase Empleado
Yo.saludoEmpleado()
```

Prototipos en JavaScript: Hay que destacar que en JS, todo es trabajado como un Objeto y se encuentran basados en Prototipos.

Un objeto en JavaScript tiene otro objeto, llamado ***prototype*** (prototipo, en español). Cuando pedimos un objeto una propiedad que no tiene, la busca en su prototipo. Así, un prototipo es otro objeto que se utiliza como una fuente de propiedades alternativas.

Prototypo es una propiedad de Objeto (el objeto del que se derivan todos los demás objetos), y esta propiedad es, a su vez, un objeto. Por tanto, el prototipo último de un objeto es **Object.prototype** . Este prototipo padre tiene métodos que incluyen todos los objetos.

Te invitamos a investigar mucho sobre el trabajo de los prototipos en JavaScript a leer los siguientes artículos en sus respectivos enlaces web:

- https://developer.mozilla.org/es/docs/Learn/JavaScript/Objects/Object_prototypes
- https://developer.mozilla.org/es/docs/Web/JavaScript/Herencia_y_la_cadena_de_protipos
- https://www.w3schools.com/js/js_object_prototypes.asp