ES6+ y POO

ES6 (Parte II)

{desafío} latam_

Utilizar las nuevas funcionalidades de la especificación ES6+ para la implementación de un algoritmo Javascript que resuelve un problema planteado

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

Te encuentras aquí

{desafío}
latam_

¿Qué aprenderás en esta sesión?

- *Utiliza la nueva nomenclatura para la definición de una clase, atributos y métodos acorde al lenguaje Javascript ES6.*
- *Identifica las nuevas funcionalidades de la especificación ES6 referentes a ejecuciones asincrónicas.*

¿Recuerdas la diferencia entre var, let y const?

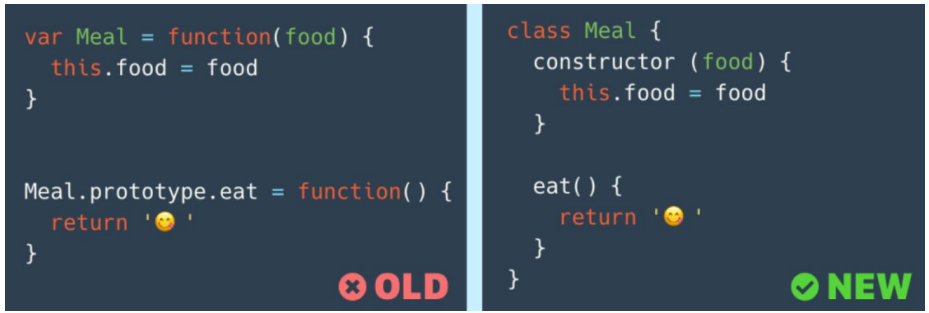
Comenta con tus palabras

{desafío} latam_

/* Clases */

Clases

Las clases (class) son el bloque de construcción fundamental de todos los lenguajes que implementan el Paradigma Orientado a Objetos. A partir de ES6 se decidió agregar la palabra reservada “class” como azúcar sintáctico, es decir, una forma más abreviada y sucinta de definir funciones constructoras y sus prototipos, como observamos en la imagen a continuación. Por lo tanto, ES6 nos provee una nueva forma de construir objetos a través de clases:



Clases

Una clase es una forma de organizar nuestro código, que nos permite abstraer ciertos conceptos y organizarlos en torno a patrones que nos permitan construir objetos. Cuando se habla de azúcar sintáctico, nos referimos a que si bien ES6 implementa clases y otros aspectos del paradigma orientado a objetos, es sólo a nivel de sintaxis, ya que como sabemos JavaScript está orientado a prototipos. Por consiguiente, para declarar una clase bajo la sintaxis de ES6, utilizaremos la palabra reservada `class`, de la siguiente manera:

```
class Cuadrado{
   constructor(lado){
       this.lado = lado;
  }
}
let c1 = new Cuadrado(10);
```

Clases

Observemos que la palabra class, como lo mencionamos, define a una nueva clase llamada `Cuadrado` (por convención, la primera letra se escribe en mayúscula, al igual que en las funciones constructoras con ES5). Pero ahora, en la clase existe la instrucción `constructor` que nos sirve para asignar los valores iniciales a las propiedades o atributos. Luego, al inicializar la variable c1, observamos que para crear un nuevo cuadrado lo seguimos haciendo de la misma forma, es decir, instanciando el objeto sobre una nueva variable.

Ejercicio guiado:
Mi primera clase

Ejercicio guiado

Mi primera clase

Crear un objeto mediante el uso de clases con ES6, bajo el nombre de “Estudiante” y que tenga como atributo el nombre de ese estudiante y la edad. Para luego instanciar el objeto enviando los valores “Juan” y “35”, respectivamente, mostrando por la terminal mediante el uso de Node el objeto. Por lo tanto, para desarrollar el ejercicio planteado se debe:

Paso 1: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crear un archivo con el nombre de script.js. Seguidamente, dentro del archivo script.js creado en el paso anterior, se debe crear nuestra clase, la cual se denominará “Estudiante”.

```
class Estudiante {
}
```

Ejercicio guiado

Mi primera clase

Paso 2: Seguidamente se deben crear los atributos del objeto mediante el constructor, por lo tanto, se debe pasar como parámetros el nombre y la edad, quedando de la siguiente forma nuestra clase:

```
class Estudiante {
   constructor(nombre, edad){
       this.nombre = nombre;
       this.edad = edad;
  }
}
```

Ejercicio guiado

Mi primera clase

Paso 3: Ahora solo queda instanciar el objeto pasando los valores de “Juan” y “35”, luego, mediante un console.log mostrar la nueva instancia creada:

```
class Estudiante {
   constructor(nombre, edad){
       this.nombre = nombre;
       this.edad = edad;
  }
}

let e1 = new Estudiante('Juan',35);
console.log(e1);
```

Ejercicio guiado

Mi primera clase

Paso 4: Finalmente, guarda el archivo y ejecútalo con Node en la terminal mediante el comando: node script.js, seguidamente aparecerá en la línea de la terminal el resultado:

```
Estudiante { nombre: 'Juan', edad: 35 }
```

Ejercicio guiado: Agregar un método a un objeto con clases de ES6

{desafío} latam_

Ejercicio guiado

Agregar un método a un objeto con clases de ES6

Crear una clase para una figura geométrica, como un cuadrado, para así recibir el valor de uno de los lados como atributo y luego retornar el valor del área de la figura geométrica, pero mediante la implementación de un método denominado “calcularArea”. Para esto debemos realizar los siguientes pasos:

Paso 1: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crear un archivo con el de script.js. Seguidamente, dentro del archivo script.js creado en el paso anterior, se debe crear nuestra clase, la cual se denominará “Cuadrado” y el constructor para el atributo “lado”.

```
class Cuadrado{
   constructor(lado){
       this.lado = lado;
  }
}
```

{desafío} latam_

Ejercicio guiado

Agregar un método a un objeto con clases de ES6

Paso 2: Seguidamente, se debe crear el método que permitirá retornar el valor del área del cuadrado, por ende, el nombre del método será “calcularArea”, y a partir de ES6 se incorporan directamente sobre la estructura de la clase, dicho método sigue siendo una función, pero bajo la sintaxis de ES6, por lo tanto, no es necesario utilizar la palabra reservada “function”:

```
class Cuadrado{
   constructor(lado){
       this.lado = lado;
   }
   calcularArea(){
     return this.lado *
this.lado;
   }
}
```

Ejercicio guiado

Agregar un método a un objeto con clases de ES6

Paso 3: Ahora solo queda instanciar el objeto pasando el valor de uno de los lados y luego, mediante un console.log hacer el llamado al método que nos permita calcular el área:

```
let c1 = new Cuadrado(5);
console.log(c1.calcularArea());
```

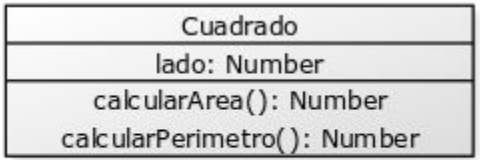
Paso 4: Finalmente, guarda el archivo y ejecútalo con Node en la terminal mediante el comando: `node script.js`, seguidamente aparecerá en la línea de la terminal el resultado:

```
25
```

/* Implementando clases de ES6 a partir de un diagrama UML */

Implementando clases de ES6 a partir de un diagrama UML

En clases anteriores, se pudo aprender un poco sobre los diagramas UML, pero la implementación fue utilizando ES5. Ahora vamos a utilizar los diagramas de clases para realizar un programa en JavaScript bajo la sintaxis de ES6. Por ende, un diagrama de clases representa las clases dentro del sistema, así como los atributos, operaciones y la relación entre clases. Veamos por ejemplo el diagrama de clases del Cuadrado que hemos estado trabajando:



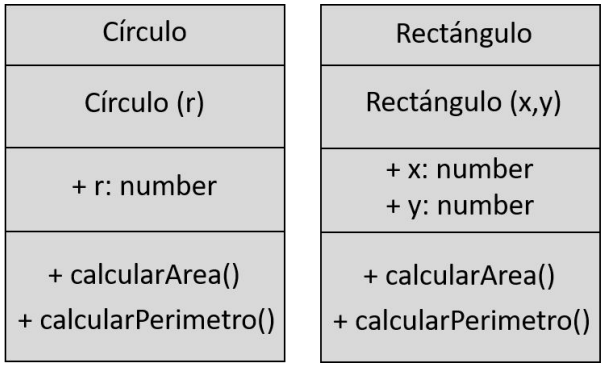
Implementando clases de ES6 a partir de un diagrama UML

Observemos que posee 3 divisiones:

1. La parte superior indica el nombre de la clase. En este caso, Cuadrado.
2. La división central se refiere a los atributos de la clase. Se indica el nombre del atributo y su tipo de dato, separado por dos puntos ":". Para el ejemplo, lado es el atributo y number el tipo de dato.
3. Finalmente, se describen los métodos de la clase y el tipo de dato que retornan. En el cuadrado, calcularArea() y calcularPerimetro() son los métodos de la clase Cuadrado y ambos retornan un dato de tipo numérico.

Implementando clases de ES6 a partir de un diagrama UML

Como veremos a continuación, también se puede describir la visibilidad de los atributos y métodos, es decir, si son públicos (+), privados (-) o protegidos (#) dentro del contexto. A continuación, podemos observar dos figuras geométricas con sus constructores y métodos. Por ende, consideremos el siguiente diagrama de clases UML:



Ejercicio guiado:
Del diagrama al código

Ejercicio guiado

Del diagrama al código

```
class Rectangulo {
    constructor(x, y) {
        this.x = x;
        this.y = y
    }
    calcularArea() {
        return this.x * this.y;
    }
    calcularPerimetro() {
        return (this.x + this.y) * 2;
    }
}
```

Paso 1: Crear una carpeta en tu lugar preferido de trabajo de tu computador, luego crea un archivo script.js. Luego, en el archivo creado, lo primero es trabajar con la clase para el Rectángulo. Es decir, crearemos la estructura de una clase en ES6 con el nombre de la figura geométrica en cuestión. Seguidamente, esta clase tendrá el constructor que recibe dos valores (x, y), estos valores serán los lados del rectángulo. Finalmente, se crean los dos métodos indicados (calcularArea y calcularPerimetro) con sus fórmulas respectivas a la figura geométrica.

Ejercicio guiado

Del diagrama al código

Paso 2: Crear la clase para el círculo, la cual, en el constructor recibirá un solo valor que será el radio. Luego se crean los métodos para el cálculo del área y del perímetro de esta figura geométrica.

```
class Circulo {
    constructor(r) {
        this.r = r;
    }
    calcularArea() {
        return this.r * this.r * Math.PI;
    }
    calcularPerimetro() {
        return this.r * Math.PI * 2;
    }
}
```

Ejercicio guiado

Del diagrama al código

Paso 3: Queda por unir todo el código en uno solo y hacer las respectivas instancias para probar nuestro código. En consecuencia, se crearán dos instancias, una para el rectángulo y una para el círculo. Tanto el círculo como el rectángulo invocarán directamente a sus métodos (calcularArea y calcularPerimetro).

```
let rectangulo1 = new Rectangulo(3,4);
console.log(rectangulo1.calcularArea());
console.log(rectangulo1.calcularPerimetro());

let circulo1 = new Circulo(3);
console.log(circulo1.calcularArea());
console.log(circulo1.calcularPerimetro());
```

Ejercicio guiado

Del diagrama al código

Paso 4: Para ejecutar el código anterior, utiliza la terminal y Node, con el comando: `node script.js.` Obteniendo como resultado:

```
12
14
28.274333882308138
18.84955592153876
```

Ejercicio guiado:
Getters y setters con ES6

{desafío}
latam_

Ejercicio guiado

Getters y setters con ES6

Paso 1: Crear una carpeta en tu lugar de trabajo favorito y dentro de ella crear un archivo con el de script.js. Seguidamente, dentro del archivo script.js creado en el paso anterior, se debe crear nuestra clase, la cual se denominará “Perro” y el constructor para el atributo “raza”, pero esta vez con la notación guion bajo “_”, quedando la propiedad: “this._raza”.

```
class Perro{
   constructor(raza){
       this._raza = raza;
  }
}
```

Ejercicio guiado

Getters y setters con ES6

Paso 2: Crear el método que permitirá retornar el valor del atributo mediante un getters, pero como estamos trabajando bajo lo nomeclatura de ES6, simplemente se agrega la palabra reservada “get” antes del nombre del método:

```
class Perro{
   constructor(raza){
       this._raza = raza;
  }
   get raza(){
     return this._raza;
   }
}
```

Ejercicio guiado

Getters y setters con ES6

Paso 3: Ya que existe el método para obtener la raza del perro, ahora se debe crear el método para poder modificar ese atributo, para ello utilizamos “set”, quien recibirá un nuevo parámetro y modificará el valor actual de la propiedad con el nuevo:

```
class Perro{
   constructor(raza){
      this._raza = raza;
   }
   get raza(){
      return this._raza;
   }
   set raza(nueva_raza){
      this._raza = nueva_raza;
   }
}
```

Ejercicio guiado

Getters y setters con ES6

```
let perro1 = new Perro('Pastor Aleman');
console.log(perro1.raza);
perro1.raza     =     'Pastor     Belga';
console.log(perro1.raza);
```

Paso 4: Ahora solo queda instanciar el objeto pasando el nombre de la raza del perro, luego mediante un console.log hacer el llamado al método raza() para observar el valor del atributo, posteriormente para realizar la modificación de ese atributo, implementamos el método setter, pasando como parámetro el nuevo valor que deseemos agregar al atributo, y luego se invoca nuevamente el método get para ver si el cambio surtió efecto.

Ejercicio guiado

Getters y setters con ES6

Paso 5: Finalmente, guarda el archivo y ejecútalo con Node en la terminal mediante el comando: node script.js, seguidamente aparecerá en la línea de la terminal el resultado:

```
Pastor Aleman
Pastor Belga
```

¿Cómo la implementación de clases a partir de un diagrama UML puede ayudar a mejorar la organización y estructuración del código en proyectos de programación más grandes y complejos?

{desafío} latam_

Resumiendo

- Las clases (class) son el bloque de construcción fundamental de todos los lenguajes que implementan el Paradigma Orientado a Objetos.
- Una clase es una forma de organizar nuestro código, que nos permite abstraer ciertos conceptos y organizarlos en torno a patrones que nos permitan construir objetos.
- Un diagrama de clases representa las clases dentro del sistema, así como los atributos, operaciones y la relación entre clases.

Próxima sesión...

- *Desafío evaluado - Clases en ES6*

{desafío}
latam_

{desafío}
latam_

Academia de
talentos digitales