TEXT CLASS REVIEW

TEMAS A TRATAR EN EL CUE:

- ¿Qué es una clase?
- El método constructor.
- Accesadores y mutadores.
- Métodos personalizados.
- Instanciación de una clase.

¿QUÉ ES UNA CLASE?

Es una forma de organizar código de forma entendible, con el objetivo de simplificar el funcionamiento de nuestro programa. Las clases son «conceptos abstractos», de los que se pueden crear objetos de programación, cada uno con sus características concretas.

Son consideradas "funciones especiales", como las expresiones y declaraciones de funciones. Su sintaxis tiene dos componentes: expresiones de clases y declaraciones de clases.

Las clases de ES6 son plantillas para crear objetos JavaScript, y encapsulan datos con código para trabajar con esos datos.

DEFINICIÓN DE UNA CLASE POR MEDIO DE UNA DECLARACIÓN

Una manera de definir una clase es mediante una declaración de clase. Para declararla, se utiliza la palabra reservada “class”, y un nombre para la clase, por ejemplo: "UserInfo".

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

DEFINICIÓN DE UNA CLASE POR MEDIO DE UNA EXPRESIÓN

Una expresión de clase es otra manera de definir una clase. Estas expresiones pueden ser nombradas, o anónimas. El nombre dado a la expresión de clase es local dentro del cuerpo de esta. Por ejemplo:

```
1  //Clase Anonima
2  let UserInfo = class{
3          constructor(nombre, edad) {
4                  this.nombre = nombre;
5                  this.edad = edad;
6          }
7  };
8  //Salida de UserInfo
9  console.log(UserInfo.name);
10
11
12 //Clase Nombrada
13 let UserInfo = class UserInfo2{
14         constructor(nombre, edad) {
15                 this.nombre = nombre;
16                 this.edad = edad;
17         }
18 };
19 //Salida del  UserInfo2
20 console.log(UserInfo.name);
```

EL MÉTODO CONSTRUCTOR

El contenido de una clase es la parte que se encuentra entre las llaves{}. En este lugar es donde se definen los miembros de clase, como los métodos o constructores.

El método constructor es un método especial para crear e inicializar un objeto creado con una clase, y en ésta, solo puede haber un método especial con su nombre. Si una clase contiene más de una ocurrencia del método constructor, se arrojará un error de sintaxis.

Un constructor permite proporcionar cualquier inicialización personalizada que deba realizarse, antes de que se pueda llamar a cualquier otro método en un objeto instanciado. Éste puede usar la palabra reservada “super” para llamar al constructor de una superclase.

En caso de no proporcionar uno propio, se entregará el predeterminado. Si su clase es una clase base, el constructor predeterminado está vacío:

```
constructor() {}
```

Si su clase es una clase derivada, el constructor predeterminado llama al constructor principal, y pasa los argumentos proporcionados:

```
constructor(...args) {
        super(...args);
}
```

ACCESADORES Y MUTADORES

Accesadores (getters): en ocasiones, se requiere tener acceso a una propiedad que devuelve un valor calculado dinámicamente, o mostrar el valor de alguna variable interna sin necesidad de métodos explícitos. En ES6, esto se logra con el uso de un getter. Para utilizarlo, se deben tener en cuenta las siguientes consideraciones:

- Puede tener un identificador que sea un número, o una cadena.
- No debe tener parámetros.
- No deben existir múltiples getters para una misma propiedad.

La idea básica de éste, es que agrega sintaxis para definir propiedades de acceso como métodos, en lugar de propiedades de datos simples. Un getter se define por la palabra clave get, seguida de una función con el nombre de la propiedad, que no toma argumentos, y devuelve su valor.

Mutadores (setters): en JavaScript, éstos se utilizan para ejecutar una función donde se requiera cambiar una propiedad. Generalmente, son ocupados en conjunto con getters, para crear un tipo de pseudo-propiedad. No es posible tener simultáneamente un setter en una propiedad que ya tiene un valor. Para hacer uso de ellos, se debe considerar los siguientes aspectos:

- Puede tener un identificador que sea un número, o una cadena.
- No debe tener parámetros.
- No deben existir en un objeto de notación literal con otro set, o con otra entrada de datos con la misma propiedad.

Un setter se define mediante la palabra clave set, seguida de una función con el nombre de la propiedad que toma el nuevo valor de la propiedad como parámetro.

JavaScript ES6 tiene sus propios métodos de acceso get y set. Veamos cómo funcionan:

```
class UserInfo{
        constructor(nombre, edad) {
                this.nombre = nombre;
                this.edad = edad;
        }
        get nombre() {
                return this._nombre;
        }
        set nombre(nuevoNombre) {
                 nuevoNombre =  nuevoNombre.trim();
                if (nuevoNombre == ""){
                        throw "nuevoNombre no puede estar vacía";
                }
                this._nombre = nuevoNombre;

        }
}
```

Al observar el código anterior, notaremos que la propiedad nombre ha cambiado a _nombre, esto ocurre para evitar la colisión con el getter y el setter.

El accesador o getter, usa la palabra clave get, seguida del nombre del método, que en este caso es nombre.

```
get nombre() {
 return this._nombre;
}
```

Para llamar al getter, se utiliza la siguiente sintaxis:

```
let  nombre = userInfo.nombre;
```

Cuando Javascript ve el acceso a la propiedad nombre de la clase UserInfo, éste comprueba si la clase tiene alguna propiedad nombre. De lo contrario, verifica si tiene algún método que se vincule a la propiedad del nombre. En este ejemplo, el método nombre() se vincula a la propiedad nombre,

a través de la palabra clave get. Una vez que JavaScript encuentra el método getter, lo ejecuta, y devuelve un valor.

El setter utiliza la palabra clave set, seguida por el nombre del método:

```
set nombre(nuevoNombre) {
  nuevoNombre =  nuevoNombre.trim();
  if (nuevoNombre == ""){
   throw "nuevoNombre no puede estar vacía";
  }
  this._nombre = nuevoNombre;
 }
}
```

JavaScript llamará al setter **nombre()**, cuando asigne un valor a la propiedad nombre. Se verá así:

```
userInfo.nombre = 'María Zerpa';
```

MÉTODOS PERSONALIZADOS

Las funciones son objetos completamente tipificados, los cuales pueden manipularse, extenderse, y transmitirse como datos. Una estructura de función normal en nodejs, se define de la siguiente manera:

```
function nombre_de_la_funcion() {
 // cuerpo de la función
 // retorno opcional
}
```

Toda función devuelve un valor. En ausencia de una sentencia de retorno explícita, la función retorna el valor undefined.

Por ejemplo:

```
function mi_dato() {
        return 111;
}
console.log(mi_dato()); //devuelve 111
```

```
5
6 function mi_valor() {
7 }
8 console.log(mi_valor()); //devuelve undefined
```

INSTANCIACIÓN DE UNA CLASE

Por otra parte, una instancia es la instanciación de una clase; es decir, uno de sus miembros. Por ejemplo: Manuel podría ser una instancia de la clase Empleado, representando a un individuo en particular como un empleado. Una instancia tiene exactamente las mismas propiedades de su clase padre (ni más, ni menos).

Un lenguaje basado en prototipos, tal como lo es JavaScript, no hace esta distinción: simplemente tiene objetos.