JS SEMINARIO DE LENGUAJES

CLASE 5

OBJETOS EN JAVASCRIPT

Exploración de objetos literales, clases, propiedades, métodos y prototipos en JavaScript.

AGENDA

1. Objetos 1
2. Clases Y Objetos
3. El Método Contructor
4. Métodos Y Funciones Flecha
5. Mergeando Un Objeto Con Otros
6. Spread Operator ...
7. Herencia

OBJETOS [1]

En JavaScript todo es un objeto. Es una colección de datos relacionados que consta de atributos y métodos. Permiten mantener la información segura y protegida en su propio entorno, "encapsulada"

OBJETOS 2

Creación de objetos. Literal.

```
var nombreObjeto = {
  miembro1Nombre: miembro1Valor,
  miembro2Nombre: miembro2Valor,
  miembro3Nombre: miembro3Valor
  }
```

OBJETOS 3

Creación de objetos. Literal.

```
var auto = {
  marca: "Tesla",
  modelo: "Model S",
  tipo: "sedan",
  descripcion: function(){
    return this.marca + " - " + this.modelo + " - " + this.tipo + "
  }
```

Ver ejemplo

OBJETOS [4]

Creación de objetos. Literal.

¿Y si ahora quiero crear otro objeto auto?

```
var auto2 = {
  marca: "Waymo",
  modelo: "One",
  tipo: "sedan",
  descripcion: function(){
    return this.marca + " - " + this.modelo + " - " + this.tipo + "
  }
```

Evidentemente esto no escala. ¿Qué necesitamos?

CLASES EN JS[1]

Las clases se utilizan para representar cualquier tipo de dato. Por ejemplo autos, personas, clientes, etc.
Entonces una clase representa cada uno de estos elementos: Car, Person, Client, etc.
Un objeto es un elemento particular de la clase, una instancia.
Se introdujo en ECMAScript 2015. Es una mejora sintáctica.

CLASES EN JS²

Por ejemplo, la persona identificada como "Messi" y nombre "Lionel Andrés" representa un objeto particular de la clase Person.

CLASES EN JS[3]

El objeto de clase Person puede ser asociado a una variable p de un programa. De esta manera se crean variables para identificar cada objeto asociado con la clase.

La pregunta al crear una clase es: ¿qué características tiene un objeto de la clase? y ¿qué acciones queremos que realice el objeto de esa clase?

CLASES EN JS[4]

```
class Person {
  // class properties
  lastname = "";
  firstname = "";
  age = 32;

  // class methods
  display() {
    console.log(" The person's lastname = " + this.lastname +
          ", firstname = " + this.firstname);
  }
  }
```

CLASES EN JS[5]

Las características se denominan propiedades de una clase y los métodos es el comportamiento.
En JS una clase se crea con la palabra clave class seguido del nombre de la clase, seguido de { }
Por ejemplo

```
class Person {

}
```

AGREGANDO PROPIEDADES A UNA CLASE

Se agregan en el cuerpo de la clase, sin necesidad de usar let o var.

```
class Person {
  firstname;
  lastname;
  age;
  }

  var p = new Person;
  console.log(p); // ¿Qué retorna? Probarlo en un editor
```

Se declaran las propiedades sin un valor específico.

AGREGANDO MÉTODOS A UNA CLASE[1]

Es posible agregar métodos a una clase y los objetos creados a partir del ella pueden invocarlos directamente.

```
// class methods
display() {
    console.log(" The person's lastname = " + this.lastname +
      ", firstname = " + this.firstname);
}
```

CLASES Y OBJETOS

Una vez creada la clase es posible crear objetos asociados a esta clase utilizando la plalabra clave new seguido del nombre de la clase.

```
var p = new Person;
p.lastname = "Lionel Andrés";  // initialization of the lastname prc
p.firstname = "Messi";    // initialization of the firstname propert
console.log("Variable p = ", p);
p.display();
```

THIS

La palabra clave this se utiliza para referirse al objeto en sí mismo y acceder por ejemplo a las propiedades de la calse.

```
// class methods
display() {
  console.log(" The person's lastname = " + this.lastname +
        ", firstname = " + this.firstname);
```

CLASES Y OBJETOS

Una vez creado el objeto, es posible cambiar el valor de las propiedades del mismo.

```
var p = new Person;
p.lastname = "Lionel Andrés";  // initialization of the lastname prc
p.firstname = "Messi";    // initialization of the firstname propert
console.log("Variable p = ", p);
p.display();
```

EL MÉTODO CONTRUCTOR

Se invoca con cada sentencia new. Es la forma más recomendable de crear una clase. Define una estructura clara y la inicialización de las propiedades es más específica. Puede o no tener parámetros.

```
class Person {
  // class properties
  lastname = "";
  firstname = "";
  age = 32;
  // class methods
  constructor(lastname, firstname, age){
    this.lastname = lastname;
    this.firstname = firstname;
    this.age = age;
  }
  display() {
    console.log(" The person's lastname = " + this.lastname +
          ", firstname = " + this.firstname);
```

```
}
```

MÉTODOS Y FUNCIONES FLECHA

Las funciones flecha no son ideales si necesitas acceder a otras propiedades del objeto usando this. No tienen su propio enlace a this o super.

```
var obj = {
  i: 10,
  b: () => console.log(this.i, this),
  c: function () {
    console.log(this.i, this);
  },
  };

  obj.b(); // imprime indefinido, Window {...} (o el objeto global)
  obj.c(); // imprime 10, Object {...}
```

Ver ejemplo línea 46

MERGEANDO UN OBJETO CON OTROS

Comparten la misma posición de memoria

```
var p = { lastname : "Messi", firstname : "Lionel" };
console.log("p (before modification of p2) =", p);
// p = { lastname : "Messi", firstname : "Lionel" }
var p2 = p;
p2.city = "Mar del Plata";
console.log("p (after modification of p2) =", p);
// p = { lastname : "Messi", firstname : "Lionel" }
// city : "Mar del Plata"}
```

SPREAD OPERATOR ...

La idea es poder crear un objeto a partir de otro.

```
var p2 = { ...p}; // copy the properties of object p into
// object p2
p2.city = "Mar del Plata";
```

HERENCIA

Mencanismo que permite reutilizar código y organizar lógicamente las clases creando nuevas a partir de clases existentes.
Se heredan propiedades y métodos.

- Herencia de Clases: palabra clave extends, crea una clase hija que hereda de la clase padre.
- Herencia funcional: composición de objetos mediante funciones
- Herencia prototípica: cada objeto internamente tiene un enlace a su prototype. Se usa para los objetos como String y Array.

HERENCIA DE CLASES[1]

```
..
class Student extends Person {
  studying() {
    console.log(this.name + " is studying.");
  }
}

let e = new Student("Trinidad", "Estudiante de secundaria");
e.birthday(16);
e.display();
e.studying();
```

Ver ejemplo de herencia de clases

HERENCIA DE CLASES²

super permite agregar una propiedad o atributo a la clase hija. Las propiedades heredadas se inicializan correctamente.

```
class Student extends Person {
  constructor(name, role, career) {
    super(name, role);          //call parent constructor
    this.career = career;       // add a new properties
  }

  studying() {
    console.log(this.name + " is studying" + this.career);
  }
}

let p = new Person ("Paola", "Docente");
p.birthday(49);
p.display();
```

HERENCIA DE CLASES[3]

Los métodos heredados pueden sobreescribir un método de la clase padre cuando la clase hija necesita un comportamiento diferente.

```
1  class Student extends Person {
2    constructor(name, role, career) {
3      super(name, role);           //call parent constructor
4      this.career = career;        // add a new properties
5    }
6
7    studying() {
8      console.log(this.name + " is studying" + this.career);
9    }
10   display(){
11     console.log("The person's name = " + this.name + " is " + th
12   }
```

HERENCIA FUNCIONAL

Enfoque alternativo a la herencia de clases. Se basa en ala combinación de objetos y reutilización de funcionalidad existente.

```
const Auto = {
  init(marca, modelo, tipo) {
    this.marca = marca;
    this.modelo = modelo;
    this.tipo = tipo;
    },
  display() {
    console.log(this.modelo);
  },
  description(){
      console.log(this.marca + " - " + this.modelo + " - " + this
    },
  };

const Hibrido = {
```

DIFERENCIA ENTRE FUNCIÓN Y OBJETO Y MÉTODO

Una función es un conjunto de instrucciones.
Un objeto es una entidad con propiedades y métodos.
A su vez, un método forma parte de un objeto y se invoca asociado al mismo mientras que la función se puede invocar por sí misma.
Además la variable interna this, en un método apunta al objeto y en una función depende si estamos en modo estricto o no.

REFERENCIAS

Clases en JavaScript

Clases en JavaScript. Guía de Referencia

El tutorial de JavaScript moderno

Herencia de Clases. LenguajeJS.com

Herencia de clases en JavaScript