## Tema 5.- ¿Qué son los objetos en JavaScript?

Toda la información que proviene del navegador o del documento está organizada en un modelo de objetos, en el que cada objeto tendrá sus propiedades y métodos. Pues bien, JavaScript también ofrece la posibilidad de crear objetos propios, que tendrán sus propiedades y métodos para que un desarrollador pueda definir a su antojo.

Es necesario aclarar que **JavaScript no es un lenguaje orientado a objetos en sentido estricto**. Se considera que, **JavaScript es un lenguaje basado en objetos**. La diferencia entre orientado a objetos y basado en objetos es significante, y tiene que ver sobre todo en cómo los objetos se pueden extender.

**Un objeto en JavaScript es realmente una colección de propiedades.** Las propiedades pueden tener forma de datos, tipos, funciones (métodos) o incluso otros objetos.

**Una función contenida en un objeto se conoce como un método**. Los métodos son como las funciones vistas anteriormente, pero diseñados para ser utilizados en el contexto de un objeto, y por lo tanto, tendrán acceso a las propiedades de ese objeto. Esta conexión entre propiedades y métodos es uno de los ejes centrales de la orientación a objetos.

Los objetos se crean empleando una función especial denominada **constructor**, determinada por el nombre del objeto, exactamente igual que los arrays que hemos tratado anteriormente.

Este es un ejemplo de una función constructor:

```
function Car( )
{
      // propiedades y métodos
}
```

Esta función es la base para crear objetos de tipo Car. Por así decirlo, un constructor es una plantilla que se utiliza para crear objetos.

Por convención, los nombres de los constructores se ponen con la primera letra mayúscula, y cuando creamos un objeto con ese constructor (**instancia de ese objeto**), lo haremos empleando minúsculas al principio, como cualquier otra variable. Por ejemplo: var myCar = new Car( );

La palabra reservada **new se emplea para crear objetos en JavaScript**. Al crear la variable myCar estamos creando una instancia de la clase Car, es decir hemos instanciado el objeto Coche, o dicho de otra manera hemos creado un objeto Coche.

**¿Quieres saber más?**

Más información sobre creación de Objetos en JavaScript:

http://www.desarrolloweb.com/articulos/787.php

http://www.desarrolloweb.com/articulos/788.php

# ¿Cómo funcionan las propiedades de los objetos?

JavaScript ofrece la posibilidad de crear propiedades dentro de los objetos. **Las propiedades para nuestro objeto se crearán dentro del constructor** empleando para ello la palabra reservada **this**.

```
function Coche( )
{
      // Propiedades
      this.model = "Audi A6";
      this.gas = "diesel";
      this.gasAmount = 0; // Cantidad de combustible en el depósito.
}

//A continuación instanciamos dos objetos Car

var martinCar = new Coche( );
var maryCar = new Coche( );
```

La palabra reservada **this**, se utiliza **para hacer referencia al objeto actual**, que en este caso será el objeto que está siendo creado por el constructor, por lo que se usa **this** para crear nuevas propiedades para el objeto.

El único problema con el ejemplo anterior es, que todos los objetos que hagamos del tipo Car serán siempre Audi A6, diésel, y sin litros de combustible en el depósito, tanto el coche de Martin como el de Mary son iguales, ya que los dos son Audi A6, diésel, y sin litros de combustible en el depósito.

Lo ideal sería por lo tanto que en el momento de **instanciar** un objeto de tipo Car, **le pasemos al constructor la información que queramos**, como digamos la marca de coche y el tipo de combustible que utiliza. Para poder hacer esto, se podrá pasar información en el constructor:

```
//Creamos una clase que pueda recibir información mediante su
constructor

function Car(model, gas)
{
      // Propiedades
      this.model = model;
      this.gas = gas;
      this.gasAmount = 0; // Cantidad de combustible inicial por
defecto en el depósito.
}

//volvemos a crear los objetos Car de Martin y Mary, pero ahora
pudiendo definir modelo y tipo de combustible

var martinCar = new Car("Volkswagen Golf","gas");
var maryCar = new Car("Mercedes SLK","diesel");
```

En el anterior ejemplo hemos creado dos coches de diferentes marcas y tipos de gasolina, ahora vamos a ver **cómo se accede a las propiedades** de los objetos para consultar o modificar valores:

```
document.write("<br/>El coche de Martin es un: "+martinCar.model+" a
"+ martinCar.gas);
document.write("<br/>El coche de Silvia es un: "+ maryCar.model+" a
"+ maryCar. gas);
// Imprimirá:
// El coche de Martin es un: Volkswagen Golf a gasolina
// El coche de Silvia es un: Mercedes SLK a diesel


// Ahora modificamos la marca y el combustible en las propiedades:
martinCar.model = "BMW X5";
martinCar.gas = "diesel";
document.write("<br/>El coche de Martin es un: " + martinCar.model +
" a " +
martinCar.gas);
// Imprimirá: El coche de Martin es un: BMW X5 a diesel
```

**Los métodos, que serán funciones que se otorgan a los objetos**. Es posible acceder a las propiedades del objeto desde sus métodos, para poder realizar operaciones con ellas.

Un ejemplo de un método que se podría utilizar en la clase Car:

```
function Car(model, gas)
{
	// Propiedades
	this.model = model;
	this.gas = gas;
	this.gasAmount = 0;
	// Métodos
	this.fillTank = function (litres)
	{
		this. gasAmount += litres

	};
}
```

El método fillTank() lo hemos declarado dentro el objeto Car. La partícula **this** delante del nombre de la función indica que es un método que pertenece al objeto, en caso de no añadir esta partícula, sería un método global común para todos los objetos Car.

Otra forma de declarar el método envejecer será el siguiente:

```
function Car(model, gas)
{
	// Propiedades
	this.model = model;
	this.gas = gas;
	this.gasAmount = 0;


	// Métodos
	Car.prototype.fillTank = function (litres)
	{
		this. gasAmount += litres

	};
}
```

Si bien la primera de las dos formas vistas de declarar un método es la más comúnmente utilizada, la segunda es la forma más correcta.

Hay que señalar que se pueden instanciar tantos objetos como se quiera, **siempre que se le asignen variables diferentes**. También es posible por ejemplo, instanciar múltiples objetos tipo Car diferentes, y almacenarlos en cada una de las posiciones de un array, como en el siguiente ejemplo:

```
<script type="text/javascript">
      var cars = new Array(4);
      cars[0] = new Car("Ferrari Scaglietti", "Diesel");
      cars[1] = new Car("BMW Z4", "Gasolina");
      cars[2] = new Car("Seat Toledo", "Diesel");
      cars[3] = new Car("Fiat 500", "Diesel");
      for(i=0; i<cars.length; i++){
            document.write("Marca: " + cars[i].model +
            " - Cantidad de gasolina: " + cars[i].gas + "<br>");
      }
</script>
```

# Glosario

**Clase**: Define las características del Objeto.

**Objeto**: Una instancia de una Clase.

**Propiedad**: Una característica del Objeto, como el color.

**Método**: Una capacidad del Objeto, como caminar.

**Constructor**: Es un método llamado en el momento de la creación de instancias.