4.A. Estructuras definidas por el usuario en JavaScript

3.- Creación de objetos definidos por el usuario.

3.2.- Definición de métodos.

Las propiedades son solamente la mitad de la ecuación de la Orientación a Objetos en JavaScript. La otra mitad son **los métodos, que serán funciones que se enlazarán a los objetos**, para que dichas funciones puedan acceder a las propiedades de los mismos.

Nos definimos un ejemplo de método, que se podría utilizar en la clase Coche:

```
function rellenarDeposito (litros) {

      // Modificamos el valor de la propiedad cantidad de combustible

      this.cantidad = litros;

}
```

Fíjate que el método `rellenarDeposito`, que estamos programando a nivel global, hace referencia a la propiedad this.cantidad para indicar cuantos litros de combustible le vamos a echar al coche. Lo único que faltaría aquí es realizar la conexión entre el método `rellenarDeposito` y el objeto de tipo Coche (recuerda que los objetos podrán tener propiedades y métodos y hasta este momento sólo hemos definido propiedades dentro del constructor). Sin esta conexión la palabra reservada `this` no tiene sentido en esa función, ya que no sabría cuál es el objeto actual. Veamos cómo realizar la conexión de ese método, con el objeto dentro del constructor:

```
function Coche(marca, combustible) {

      // Propiedades

      this.marca =  marca;

      this.combustible =  combustible;

      this.cantidad = 0;          // Cantidad de combustible inicial por defecto en el depósito.


      // Métodos

      this.rellenarDeposito = rellenarDeposito;

}
```

Aquí se ve de forma ilustrada, que los métodos son en realidad propiedades: se declaran igual que las propiedades, por lo que son enmascarados como propiedades en JavaScript. Hemos creado una nueva propiedad llamada `rellenarDeposito` y se le ha asociado el método `rellenarDeposito`. Es muy importante destacar que el método `rellenarDeposito()` se referencia sin paréntesis dentro del constructor, `this.rellenarDeposito = rellenarDeposito`.

Ejemplo de uso del método anterior:

```
cocheDeMartin.rellenarDeposito(35);

document.write("<br/>El coche de Martin tiene "+cocheDeMartin.cantidad+ " litros de " +
cocheDeMartin.combustible+ " en el depósito.");

// Imprimirá

// El coche de Martin tiene 35 litros de diesel en el depósito.
```

La forma en la que hemos definido el método `rellenarDeposito` a nivel global, no es la mejor práctica en la programación orientada a objetos. **Una mejor aproximación sería definir el contenido de la función `rellenarDeposito` dentro del constructor, ya que de esta forma los métodos al estar programados a nivel local aportan mayor privacidad y seguridad al objeto en general**, por ejemplo:

```
function Coche(marca, combustible) {

     // Propiedades

     this.marca =  marca;

     this.combustible =  combustible;

     this.cantidad = 0;

     // Métodos

     this.rellenarDeposito = function (litros) {

          this.cantidad=litros;

     };

}
```

Una aproximación aún mejor es utilizar la característica del lenguaje JavaScript de ser un lenguaje orientado a prototipos. Un prototipo en JavaScript es una propiedad que tienen todas las funciones para almacenar propiedades que se asignan a un objeto cuando este es creado con el operador `new`. De este modo el prototipo almacena propiedades (y por tanto métodos) que, al pertenecer al prototipo, pertenecen por así decirlo a todos los objetos que comparten ese prototipo.

En nuestro ejemplo, al asignar el método `rellenarDeposito` al prototipo del constructor Coche en lugar de asignarlo al objeto, todos los objetos creados con el constructor, reciben el mismo método `rellenarDeposito` al contrario que los ejemplos anteriores donde cada objeto Coche tiene su propio método `rellenarDeposito`.

Esto tiene dos implicaciones importantes:

- No se duplica código. Esto es, el mismo método se utiliza para todos los objetos con el mismo prototipo
- Permite implementar la herencia de propiedades, si una propiedad no se encuentra en un objeto, ésta se busca en su prototipo y si no se encuentra, se busca en el prototipo de su prototipo y así sucesivamente hasta terminar de recorrer la cadena de prototipos.

```
function Coche(marca, combustible) {

     // Propiedades

     this.marca =  marca;

     this.combustible =  combustible;

     this.cantidad = 0;
}

// Métodos

Coche.prototype.rellenarDeposito = function (litros) {

     this.cantidad=litros;

};
```

Para saber más

Más información sobre el uso de prototipos en JavaScript.

Más información sobre herencia y la cadena de prototipos.

◀ Solución a la tarea para DWEC04

Ir a...

4.B. Objetos JavaScript en W3Schools ▶