Objetos. Prototipos

- C.F.G.S. DAW
- 6 horas semanales
- Mes aprox. de impartición: Dic

Carmelo José Jaén Díaz

- iPasen - cjaedia071@g.educaand.es

Índice

OBJETIVO

- Profundizar en el concepto de objeto.
- Conocer y manejar funciones relativas al lenguaje sobre arrays, strings, números.

GLOSARIO

Backtracking. Estrategia utilizada en algoritmos que resuelven problemas que tienen ciertas restricciones. Este término fue creado por primera vez por el matemático D. H. Lehmer en la década de los cincuenta.

BOM (Browser Object Model). Convención específica implementada por los navegadores para que JavaScript pudiese hacer uso de sus métodos y propiedades de forma uniforme.

Expresión regular. Secuencia de caracteres que forman un patrón determinado, generalmente un patrón de búsqueda.

NaN. Propiedad que indica Not a Number (valor no numérico).

Objeto window. Aquel que soportan todos los navegadores y que representa la ventana del navegador. Se estudiará en profundidad en capítulos posteriores.

URI (Uniform Resource Identifier). Cadena de caracteres que identifica un recurso en una red de forma unívoca. Una URI puede ser una URL, una URN o ambas.

GLOSARIO

URN. Localizador de recursos en la web que funciona de forma parecida a una URL, pero su principal diferencia es que no indica exactamente dónde se encuentra dicho objeto.

INTRODUCCIÓN

Los prototipos son un modo que nos permite que los **objetos de Javascript hereden características entre sí.**

En esta lección vamos a aprender a **diferenciar un objeto y un prototipo** y a ver cómo **añadir y borrar propiedades y métodos a un objeto y a un prototipo.**

Pregunta introductoria: ¿Hasta qué punto es práctico crear las propiedades con prototipo?

Pongamos que tienes un tipo de objeto, el que sea, que tiene tres atributos, pero en otra parte del programa necesitas trabajar con un cuarto atributo. Puedes usar siempre cuatro y en una parte del programa no utilizar ese cuarto atributo, o bien incluirlo solamente allí donde lo necesitas. De esta manera estarías creando objetos “a medida” en las partes del programa que necesitaras. Quizá no es lo habitual, pero al menos tienes la opción de hacerlo allí donde quieres.

PROTOTIPOS

Todos los objetos tienen un prototipo, que a su vez es un objeto.

Es decir, hasta ahora hemos estudiado que los objetos tienen propiedades y métodos, pues adicionalmente tienen una propiedad denominada *prototype* que contiene un objeto y este objeto tiene las propiedades y los métodos asociados al objeto que estamos definiendo.

A continuación, se emplea un ejemplo para facilitar la comprensión de este nuevo concepto.

SINTAXIS DE CREACIÓN DE UN PROTOTIPO USANDO LA FUNCIÓN CONSTRUCTOR

En primer lugar vamos a crear un objeto mediante su constructor (en las lecciones previas lo hemos hecho mediante su declaración simple). Se recomienda empezar por mayúscula a la hora de nombrar objetos, es similar a la definición de clases en POO.

```
function Docente (nom, ape, ini){
        this.nombre = nom;
        this.apellido = ape;
        this.inicio = ini;

        this.nombreCompleto = function (){
            return this.nombre + " " + this.apellido;
        }
    }
```

RECORDATORIO Y SINGULARIDAD
Añadir una propiedad a un objeto

Antes hemos creado un prototipo de un objeto empleando su constructor, si ahora queremos crear objetos de ese tipo **Docente**, haría lo siguiente:

```
let docenteSAI = new Docente ("Carmelo", "Jaén", "2020");
let docenteEE = new Docente ("José", "Díaz", "2022");
```

Si recordamos, para añadir una propiedad a un objeto escribiríamos:

```
docenteSAI.especialidad= "Sistemas y Aplicaciones Informáticas";
//De esta manera añadiríamos la propiedad especialidad a docenteSAI
//pero no a docenteEE
```

RECORDATORIO Y SINGULARIDAD
Añadir una propiedad a un objeto

Si intentamos acceder a esa propiedad:

```
alert(docenteSAI.especialidad); //Devuelve "Sistemas y Aplicaciones Informáticas"
```

Pero si probamos:

```
alert(docenteEE.especialidad); //Devuelve "Undefined"
```

Lo que hemos hecho ha sido añadir una propiedad a un objeto (`docenteSAI`) no a un prototipo (`Docente`). Por eso, el objeto `docenteEE` no adquiere esa nueva propiedad.

RECORDATORIO Y SINGULARIDAD
Añadir un método a un objeto

Si recordamos, para añadir un método a un objeto escribiríamos:

```
docenteSAI.presentacion = function(){
        return "Hola, me llamo "+this.nombre+ “ y soy el profesor de ”
+this.especialidad;
      }

//De esta manera añadiríamos el método presentacion a docenteSAI
//pero no a docenteEE
```

RECORDATORIO Y SINGULARIDAD
Añadir un método a un objeto

Si intentamos acceder a ese método:

```
alert(docenteSAI.presentacion());
//Devuelve "Hola me llamo Carmelo y soy el profesor de Sistemas y Aplicaciones Informáticas"
```

Pero si probamos:

```
alert(docenteEE.presentacion()); //Devuelve un error por consola "docenteEE.presentacion is not a function"
```

Lo que hemos hecho ha sido añadir un método a un objeto (`docenteSAI`) no a un prototipo (`Docente`). Por eso, el objeto `docenteEE` no adquiere ese nuevo método.

AÑADIR UNA PROPIEDAD A UN PROTOTIPO

Para añadir una propiedad a un prototipo tenemos dos opciones:

- Añadiendo directamente sobre la definición del prototipo.
- Mediante la sintaxis: `<Nombre_objeto>.prototype.<propiedad> =...`

```
Docente.prototype.tutor = true;
```

Al hacer esto, es lo mismo que si la hubiéramos definido en la creación del prototipo (objeto `Docente`).

AÑADIR UNA PROPIEDAD A UN PROTOTIPO

Ahora podría hacer:

```
alert(docenteSAI.tutor);
//Devuelve true
```

Y:

```
alert(docenteEE.tutor);
//También devuelve true
```

AÑADIR UN MÉTODO A UN PROTOTIPO

Para añadir un método a un prototipo tenemos dos opciones:

- Añadiendo directamente sobre la definición del prototipo.
- Mediante la sintaxis: `<Nombre_objeto>.prototype.<metodo> =...`

```
Docente.prototype.tieneTutoria = function(){
      return "¿Es el docente " +this.nombre+ " tutor? " +this.tutor;
   }
```

Al hacer esto, es lo mismo que si lo hubiéramos definido en la creación del prototipo (objeto `Docente`).

EJEMPLO

Por último, si creamos un nuevo objeto del tipo `Docente`:

```
let docenteMates = new Docente("Donato", "Ramón", "1912");
```

Tendrá asociados una propiedad `tutor` (con valor `true`) y un método `tieneTutoria()`, por tanto si hacemos:

```
alert (docenteMates.nombre);
//Devuelve “Donato”
alert (docenteMates.tieneTutoria());
//Devuelve “¿Es el docente Donato tutor? true”
```

PROTOTIPOS

Por tanto, si queremos añadir una propiedad o un método a todos los objetos.

1. Tenemos que crear el prototipo del objeto mediante su constructor
2. Añadiremos las propiedades o métodos haciendo uso de la palabra reservada `prototype`.

Podemos decir, que en la versión ES5, este era el concepto de “herencia”.

Veremos ahora las novedades de la ES6, con la introducción del concepto de “clases” típico de la POO.