APRENDERAPROGRAMAR.COM

# STATIC JAVASCRIPT. PROPIEDADES Y MÉTODOS ESTÁTICOS O "DE CLASE". EJERCICIO. CÓDIGO EJEMPLOS BÁSICOS (CU01148E)

Sección: Cursos
Categoría: Tutorial básico del programador web: JavaScript desde cero

Fecha revisión: 2029

**Resumen:** Entrega nº48 del Tutorial básico "JavaScript desde cero".

Autor: César Krall

## STATIC JAVASCRIPT

A diferencia de otros lenguajes, JavaScript no utiliza la palabra clave static para definir variables o métodos estáticos, aunque sí provee de la posibilidad de definir propiedades y métodos únicos asociados a un tipo definido equivalentes a las propiedades y métodos estáticos.


Recordar que si escribimos una definición de tipo de objeto basada en function y dentro de ella un método o propiedad antecedida de la palabra clave this, cada vez que se crea una instancia con new se generan copias de las propiedades y métodos, cosa que ya hemos visto que puede resultar ineficiente. Hemos visto como alternativa, definir propiedades y métodos en el prototipo que son heredados por todas las instancias de ese tipo de objeto. De este modo, todos los objetos tienen acceso a esa propiedad o método por herencia prototípica: cuando se invoca la propiedad o método sobre el objeto y no se encuentra, se procede a la búsqueda en el objeto prototipo (el "padre" del objeto).

La ventaja de usar prototype es que estos propiedades y métodos sólo existen una vez en memoria y no generan duplicados para cada objeto, y así es más eficiente el código. Además, las propiedades y métodos pueden transmitirse a lo largo de una cadena de herencia y ser accesibles desde cualquier objeto.

## SIMULAR PROPIEDADES Y MÉTODOS ESTÁTICOS

Tenemos aún otra manera de generar propiedades y métodos: declararlos como propiedades y métodos asociados al objeto que define el tipo (lo que llamaríamos "la clase"), de modo que sólo serán accesibles invocando al nombre del tipo de objeto, pero no a través de las instancias. Esto se asemeja mucho a lo que en otros lenguajes se denomina propiedades y métodos estáticos, de ahí que por analogía muchas veces se aluda a este tipo de propiedades y métodos como estáticos.

Para definir métodos y propiedades que simulan ser estáticos podemos hacerlo fuera de la función constructora con esta sintaxis:

```
function nombreObjeto (par1, par2, ..., parN) {
    this.propiedad1 = valorPropiedad1;
    this.propiedad2 = valorPropiedad2;
    ... }
nombreObjeto.nombrePropiedadEstática1 = valorPropiedadEstática1;
nombreObjeto.nombrePropiedadEstática1 = valorPropiedadEstática2;
...
nombreObjeto.métodoEstático1 = function (par1, par2, ...) { ... }
nombreObjeto.métodoEstático2 = function (par1, par2, ...) { ... }
...
```

También es sintácticamente posible incluir la propiedad o método estático dentro de la función constructora con esta sintaxis, que es análoga a la anterior con la diferencia de que la declaración se realiza dentro de la función:

```
function nombreObjeto (par1, par2, ..., parN) {
    this.propiedad1 = valorPropiedad1;
    this.propiedad2 = valorPropiedad2;
    ...
nombreObjeto.nombrePropiedadEstática1 = valorPropiedadEstática1;
nombreObjeto.nombrePropiedadEstática1 = valorPropiedadEstática2;
    ...
  nombreObjeto.métodoEstático1 = function (par1, par2, ...) { ... }
  nombreObjeto.métodoEstático2 = function (par1, par2, ...) { ... }
    ...
    }
```

Una propiedad o método estático no se duplica en cada objeto, sino que existe una única vez en memoria.

Escribe este código y comprueba sus resultados.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">

function Taxi (tipoMotor, numeroPasajeros, carga, velocidad) {
if (Taxi.numeroObjetos) {Taxi.numeroObjetos++;}
else {Taxi.numeroObjetos = 1; }
this.tipoMotor = tipoMotor; this.numeroPasajeros = numeroPasajeros;
this.carga = carga; this.velocidad = velocidad;
alert('Creado objeto número '+Taxi.numeroObjetos);
}

function ejemploObjetos() {
var taxi1 = new Taxi(1, 4, 300, 90);
var taxi2 = new Taxi(2, 5, 250, 100);
var taxi3 = new Taxi(1, 6, 400, 80);
alert('El número de objetos Taxi creados hasta el momento es ' + Taxi.numeroObjetos);
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemploObjetos()"> Probar </div>
</body>
</html>
```

La expresión << if (Taxi.numeroObjetos) >> devuelve true si numeroObjetos es una propiedad existente de Taxi y false en caso contrario. Si la propiedad estática no existe cuando se invoca la creación de un objeto con new, se crea con la sentencia Taxi.numeroObjetos = 1;

El resultado esperado es:

Creado objeto número 1, Creado objeto número 2, Creado objeto número 3. El número de objetos Taxi creados hasta el momento es 3.

Si volvemos a ejecutar el script el contador se sigue incrementando (4, 5, 6, si volvemos a ejecutar 7, 8, 9, etc.), excepto si volvemos a recargar la página web en nuestro navegador. Cuando tiene lugar una recarga, todo se inicializa, con lo cual volveríamos a empezar la cuenta del número de objetos por 1.

A continuación un ejemplo de declaración de una propiedad y método estático fuera de la función constructora. Escribe este código y comprueba sus resultados.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">
<html><head><title>Ejemplo aprenderaprogramar.com</title><meta charset="utf-8">
<script type="text/javascript">

function TaxiRenault (tipoMotor, numeroPasajeros, carga, velocidad) {
this.tipoMotor = tipoMotor; this.numeroPasajeros = numeroPasajeros;
this.carga = carga; this.velocidad = velocidad;
}

TaxiRenault.fabricante = 'Renault, S.A.'; //Propiedad estática
TaxiRenault.mostrarMensaje = function () {alert('Soy un taxi Renault');} //Método estático

function ejemploObjetos() {
var taxi1 = new TaxiRenault(1, 4, 300, 90);
alert ('La velocidad del taxi 1 es ' + taxi1.velocidad);
TaxiRenault.mostrarMensaje(); //Invocamos el nombre del tipo de objeto
alert ('La propiedad estática fabricante vale ' + TaxiRenault.fabricante);
alert ('Si intentamos obtener la propiedad fabricante para una instancia obtenemos: ' + taxi1.fabricante); //undefined
taxi1.mostrarMensaje(); //ERROR las instancias no tienen acceso a los métodos estáticos
}
</script>
</head>
<body><div id="cabecera"><h2>Cursos aprenderaprogramar.com</h2><h3>Ejemplos JavaScript</h3></div>
<div style="color:blue;" id ="pulsador" onclick="ejemploObjetos()"> Probar </div>
</body>
</html>
```

El resultado de ejecución esperado es:

La velocidad del taxi 1 es 90
Soy un taxi Renault
La propiedad estática fabricante vale Renault, S.A.
Si intentamos obtener la propiedad fabricante para una instancia obtenemos: undefined
... (error, taxi1.mostrarMensaje(); no se ejecuta).

De este ejemplo obtenemos la siguiente conclusión:

No se puede acceder a una propiedad o método estático desde una instancia, hay que hacerlo invocando directamente sobre el nombre del objeto que define el tipo (lo que llamaríamos sobre "la clase").

Modifica el código anterior y escribe:

TaxiRenault.mostrarMensaje = **function** () {alert('Soy un taxi Renault con carga ' + **this**.carga);}

El resultado será Soy un taxi Renault con carga undefined ¿Por qué? Porque this hace referencia al objeto dentro del cual se encuentra la invocación y en este caso estamos trabajando con un método estático que no conoce las propiedades de un objeto en particular.

**EJERCICIO**

Define un tipo de objeto Meteorito cuyas propiedades de instancia (específicas de cada objeto) sean diametro, temperatura y nombre. La temperatura será un valor numérico que suponemos está en grados centígrados. Como propiedad estática del tipo meteorito define definicionSegunDiccionario (que debe contener la definición de meteorito) y como métodos estáticos obtenerRadio (que debe devolver el radio a partir de un parámetro diametro) y obtenerTemperaturaFarenheit (que debe devolver el valor de temperatura expresado en grados Farenheit a partir de un parámetro temperatura). Crea un objeto de tipo Meteorito con un diámetro, temperatura y nombre y comprueba que puedes acceder a las propiedades y métodos estáticos mostrando por pantalla la definición de meteorito, la superficie del objeto creado y la temperatura Farenheit del objeto creado.

Para comprobar si tus respuestas y código son correctos puedes consultar en los foros aprenderaprogramar.com.

**Próxima entrega:** CU01149E

**Acceso al curso completo** en aprenderaprogramar.com -- > Cursos, o en la dirección siguiente: http://aprenderaprogramar.com/index.php?option=com_content&view=category&id=78&Itemid=206