

DESARROLLO WEB EN ENTORNO CLIENTE TEMA3: JS OBJECTS

Índice

- Definición de objetos
- Propiedades de objetos
- Métodos de objetos
- Acceso a objetos
- Constructores de objetos
- Prototypes de objetos
- Object a partir de
ECMAScript 5

1. Definiciones de objetos

3

- Definir y crear un único objeto, utilizando **un objeto literal** `{}` y palabra: `valor`
- Utilizar la palabra clave **New Object()**
- Con un **constructor** de objetos: se crea un “tipo de objetos”

- **function** person(first, last, age, eye) {
 - `this.firstName = first;`
 - `this.lastName = last;`
 - `this.age = age;`
 - `this.eyeColor = eye;`
 - `}`

`var myFather = new person("John", "Doe", 50, "blue");`

- Con **Object.Create()** del **ECMAScript 5**

1. Definiciones de objetos literal }

4

- Definir y crear un único objeto, utilizando **un objeto literal }** y propiedad: valor

Ejemplo

```
var person = {  
  firstName: "John",  
  lastName : "Doe",  
  id       : 5566,  
  fullName : function() {  
    return this.firstName + " " + this.lastName;  
  }  
};
```

- Propiedades
- Métodos
- **this:** se refiere al dueño de la función.
- En este ejemplo this es el objeto persona que posee la función fullName.
- This.firstName es la propiedad firstName de este objeto

Ej1 y Ej2

1. Definiciones de objetos Con New

5

```
<script>
//1.-person es un objeto creado de forma literal, {} y propiedad:valor
var person = {
    firstName : "John",
    lastName  : "Doe",
    age       : 50,
    eyeColor  : "blue"
};
//2.-alumno es un nuevo Objeto, creado con la palabra clave NEW
var alumno = new Object();
alumno.firstName = "Pepe";
alumno.lastName = "Perez";
alumno.age = 25;
alumno.eyeColor = "blue";
// Uso del objeto de forma literal
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";

// Uso del objeto creado con la palabra clave New.
document.getElementById("demo").innerHTML+=
alumno.firstName + " is " + alumno.age + " years old.";
</script>
```

□ Ej3 literal-objeto_y_New.html

1. Definiciones de objetos con constructor function

6

```
<script>
//Uso del constructor de objetos
//Los ejemplos anteriores son limitados en muchas situaciones. Ellos sólo crean objetos
//A veces nos gusta tener un "tipo de objeto" que se puede utilizar para crear más objetos
//La forma habitual de crear un "tipo de objeto" es utilizar una función de objeto
//ATENCIÓN DEBE LLEVAR LA PALABRA CLAVE FUNCTION

function person(first, last, age, eye) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eye;
}

var myFather = new person("John", "Doe", 50, "blue");
var myMother = new person("Sally", "Rally", 48, "green");

document.getElementById("demo").innerHTML =
    "My father is " + myFather.age + ". My mother is " + myMother.age;
</script>

</body>
</html>
```

□ Ej 4 constructor.html

1. Objetos javascript por referencia

7

- Los objetos javascript se pasan por referencia
- Las variables se pasan por valor
- Ej6 referencia.htm

```
<script>
var person = {firstName:"John", lastName:"Doe", age:50, eyeColor:"blue"}

var x = person;
x.age = 10;
// Los OBJETOS en javascript: se pasan por referencia.
// x contiene la dirección de person, es decir le apunta, es decir es la misma
// lo que loe ha ga a x se lo hago a person y viceversa
document.getElementById("demo").innerHTML =
person.firstName + " is " + person.age + " years old.";

person.firstName="Cristina";

document.getElementById("demo").innerHTML =
x.firstName + " is " + x.age + " years old.";

//Las VARIABLES JavaScript no son mutables. Sólo los objetos de JavaScript. LAS
var z= 5;
var y= 8;
z=y;
document.getElementById("demo").innerHTML+=
" La variable z( guarda una copia de y) is " + z + " <br> La variable y is " + y
</script>

</body>
</html>
```

1. Objetos javascript for...in

8

```
<script>
//USAR FOR...IN PARA RECORRER UN OBJETO
var txt = "";
var person = {fname:"John", lname:"Doe", age:25};
person.nationality = "English";
var x;
for (x in person) {
    txt += person[x] + " ";
}

document.getElementById("demo").innerHTML = txt + person.nationality;
</script>
```


1. Recorrer Objetos javascript

9

- Existen varias formas de recorrer objetos en javascript: [Ver referencia en developer.mozilla.org](https://developer.mozilla.org)

- Bucles for...in

Este método recorre todas las propiedades enumerables de un objeto y su cadena de prototipos

- `Object.Keys(o)`

Este método recorre todas las propiedades enumerables ("claves") propias (no en la cadena de prototipos) de un objeto o.

- `Object.getOwnPropertyName(o)`

Este método devuelve un arreglo que contiene todos los nombres (enumerables o no) de las propiedades de un objeto o.

- Ej7

1. Objetos javascript delete

10

□ Ej8

```
1  <html>
2  <body>
3
4  <p>You can delete object properties.</p>
5
6  <p id="demo"></p>
7
8  <script>
9  var person = {
10     firstname:"John",
11     lastname:"Doe",
12     age:50,
13     eyecolor:"blue"
14 };
15 delete person.age;
16 // delete solo es para propiedades de objeto, no para variables ni funciones
17 //objetos JavaScript heredan las propiedades de su prototipo.
18
19 //La palabra reservada delete no elimina las propiedades heredadas, pero si se
20 //una propiedad de prototipo, que afectará a todos los objetos heredados del pro
21 document.getElementById("demo").innerHTML =
22 person.firstname + " is " + person.age + " years old.";
23 </script>
24
25 </body>
26 </html>
```

1. Objetos javascript method

11

□ Ej9

```
1  <!DOCTYPE html>
2  <html>
3  <body>
4
5  <p>Creating and using an object method.</p>
6
7  <p>A method is actually a function definition stored as a property value.</p>
8
9  <p id="demo"></p>
10
11 <script>
12 var person = {
13     firstName: "John",
14     lastName : "Doe",
15     id       : 5566,
16     fullName : function() {
17         return this.firstName + " " + this.lastName;
18     }
19 };
20
21 document.getElementById("demo").innerHTML = person.fullName();
22 </script>
23 </body>
24 </html>
25
26
```

1. Objetos javascript method-modificar-propiedades

12

- Ej10
- A través de los **métodos** definidos se **pueden cambiar los valores de las propiedades, los valores para el cambio se reciben por parámetro.**

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
function person(firstName,lastName,age,eyeColor) {
  this.firstName = firstName;
  this.lastName = lastName;
  this.age = age;
  this.eyeColor = eyeColor;
  this.changeName = function (name) {
    this.lastName = name;
  }
}
var myMother = new person("Sally","Rally",48,"green");
myMother.changeName("Doe");
document.getElementById("demo").innerHTML =
"My mother's last name is " + myMother.lastName;
</script>

</body>
</html>
```

1. Objetos javascript añadir propiedades y method a instancias

13

□ Ej 1 1

```
<p id="demo"></p>

<script>
function Person(first, last, age, eye) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eye;
}

var myFather = new Person("John", "Doe", 50, "blue");
var myMother = new Person("Sally", "Rally", 48, "green");

myFather.nationality = "English";
//AÑADO UNA PROPIEDAD SOLO A MYFATHER
document.getElementById("demo").innerHTML +=
    "My father is " + myFather.nationality;

//AÑADO UN MÉTODO SOLO A MYFATHER
myFather.name = function() {
    return this.firstName + " " + this.lastName;
};
document.getElementById("demo").innerHTML +=
    "My father is FULL " + myFather.name();
//CUIDADO
document.getElementById("demo").innerHTML +=
    "My mother is " + myMother.nationality + myMother.name();
</script>

</body>
</html>
```

1. Objetos javascript añadir propiedades y method al **prototipo**

14

□ Ej 12

```
<!DOCTYPE html>
<html>
<body>

<p id="demo"></p>

<script>
function Person(first, last, age, eye) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eye;
    this.nationality="English";
    this.name = function() {
        return this.firstName + " " + this.lastName
    };
}

var myFather = new Person("John", "Doe", 50, "blue");
document.getElementById("demo").innerHTML =
    "My father is " + myFather.name();

var myMother = new Person("PEpa", "Doe", 50, "blue");
document.getElementById("demo").innerHTML +=
    "My mother is " + myMother.name();
</script>
```

1. Objetos javascript añadir propiedades y method con el método prototype

15

□ Ej 13

```
<body>

<p id="demo"></p>

<script>
function Person(first, last, age, eye) {
    this.firstName = first;
    this.lastName = last;
    this.age = age;
    this.eyeColor = eye;
}
Person.prototype.nationality="English";
Person.prototype.name_nation = function() {
    return this.firstName + " " + this.lastName + " "+this.nationality;
};
Person.prototype.changename = function() {
    this.firstName=prompt("Introduce un nombre nuevo:");
    return this.firstName + " " + this.lastName
};

var myFather = new Person("John", "Doe", 50, "blue");
document.getElementById("demo").innerHTML =
    "My father is " + myFather.changename() + myFather.nationality;

var myMother = new Person("Pepa", "Doe", 50, "blue");
document.getElementById("demo").innerHTML +=
    " My mother is " + myMother.name_nation();
</script>

</body>
```

JavaScript Accessors (Getters and Setters)

16

- Ej 14 y Ej 15
- ECMAScript 5 (2009) introduce Getter and Setters.
- Permiten el acceso a las propiedades de los objetos.
- **Why Using Getters and Setters?**
 - It gives simpler syntax
 - It allows equal syntax for properties and methods
 - It can secure better data quality
 - It is useful for doing things behind-the-scenes

Getters and Setters

17

- Ej 16 y EJ 17
- En el Ej 16, `person.fullName()` se accede como una función.
- En el Ej 17, `person.fullName` se accede como una propiedad y la sintaxis es más simple.

Getters and Setters. Data Quality get

18

□ Ej 18

Get no admite
parámetros
en ECMAScript 5.

<https://developer.mozilla.org/es/docs/Web/JavaScript/Referencia/Funciones/get>

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Getters and Setters</h2>

<p>Getters and setters allow you to get and set properties via methods.<

<p id="demo"></p>

<script>
// Create an object:
var person = {
  firstName: "John",
  lastName : "Doe",
  language : "en",
  get lang() {
    return this.language.toUpperCase();
  }
};
// Display data from the object using a getter:
document.getElementById("demo").innerHTML = person.lang;
</script>

</body>
</html>
```

Getters and Setters. Data Quality set

19

□ Ej19

```
<!DOCTYPE html>
<html>
<body>

<h2>JavaScript Getters and Setters</h2>

<p>Getters and setters allow you to get and set properties via methods.</p>

<p id="demo"></p>

<script>
// Create an object:
var person = {
  firstName: "John",
  lastName : "Doe",
  language : "",
  set lang(lang) {
    this.language = lang.toUpperCase();
  }
};
// Set a property using set:
person.lang = "en";
// Display data from the object:
document.getElementById("demo").innerHTML = person.language;
</script>

</body>
</html>
```

Getters and Setters. Object.defineProperty

20

□ Ej 20

```
<script>
// Define an object
var obj = {counter : 0};

// Define Setters and Getters
Object.defineProperty(obj, "reset", {
  get : function () {this.counter = 0;}
});
Object.defineProperty(obj, "increment", {
  get : function () {this.counter++;}
});
Object.defineProperty(obj, "decrement", {
  get : function () {this.counter--;}
});
Object.defineProperty(obj, "add", {
  set : function (value) {this.counter += value;}
});
Object.defineProperty(obj, "subtract", {
  set : function (value) {this.counter -= value;}
});

// Play with counter:
obj.reset;
obj.add = 5;
obj.subtract = 1;
obj.increment;
obj.decrement;
document.getElementById("demo").innerHTML = obj.counter;
</script>
```

Object.create y call

21

□ DWEC Tema 3 Objetos Write a program-inherit

```
var Person = function (firstName) {
  this.firstName = firstName;
};

Person.prototype.sayHello = function() {
  console.log("Hello, I'm " + this.firstName);
};

var person1 = new Person("Alice");
var person2 = new Person("Bob");

// call the Person sayHello method.
person1.sayHello(); // logs "Hello, I'm Alice"
person2.sayHello(); // logs "Hello, I'm Bob"
// Add a couple of methods to Person.prototype
Person.prototype.walk = function(){
  console.log("I am walking!");
};

Person.prototype.sayHello = function(){
  console.log("Hello, I'm " + this.firstName);
};

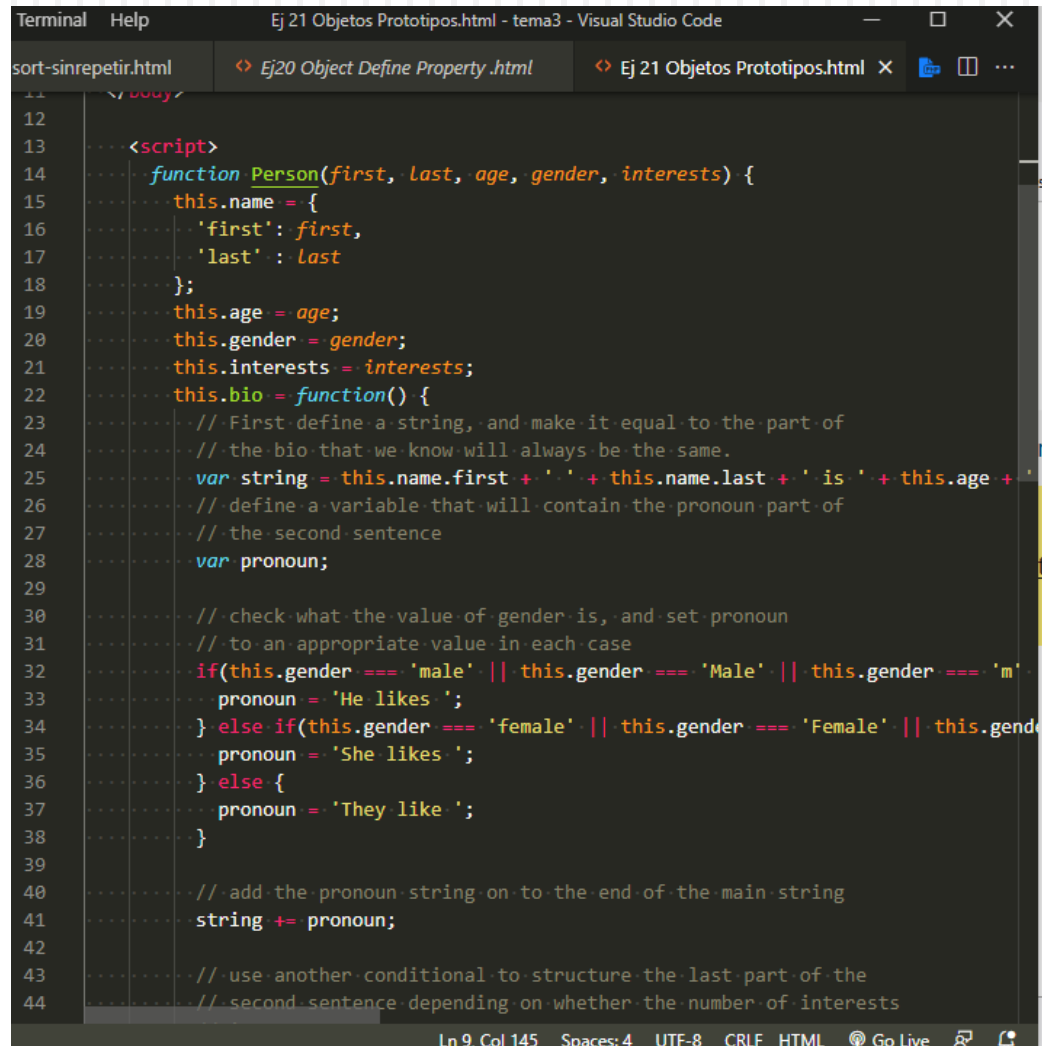
// Define the Student constructor
function Student(firstName, subject) {
  // Call the parent constructor, making sure (using call)
  // that "this" is set correctly during the call
  Person.call(this, firstName);

  // Initialize our Student-specific properties
  this.subject = subject;
}
```

Prototipos de objetos---opcional

22

- Ej 21, ver ejercicio y probar por consola: `person1.xxx`



```
11 <script>
12
13 </script>
14 function Person(first, last, age, gender, interests) {
15   this.name = {
16     'first': first,
17     'last': last
18   };
19   this.age = age;
20   this.gender = gender;
21   this.interests = interests;
22   this.bio = function() {
23     // First define a string, and make it equal to the part of
24     // the bio that we know will always be the same.
25     var string = this.name.first + ' ' + this.name.last + ' is ' + this.age +
26     // define a variable that will contain the pronoun part of
27     // the second sentence
28     var pronoun;
29
30     // check what the value of gender is, and set pronoun
31     // to an appropriate value in each case
32     if(this.gender === 'male' || this.gender === 'Male' || this.gender === 'm' ||
33     pronoun = 'He likes ';
34     } else if(this.gender === 'female' || this.gender === 'Female' || this.gende
35     pronoun = 'She likes ';
36     } else {
37     pronoun = 'They like ';
38     }
39
40     // add the pronoun string on to the end of the main string
41     string += pronoun;
42
43     // use another conditional to structure the last part of the
44     // second sentence depending on whether the number of interests
```

Herencia Prototípica--opcional

23

- Ej 22

Objetos javascript

24

- Ver toda la referencia de:
- [Trabajar con objetos en developer.mozilla.org.](#)
- [Prototipos de objetos en developer.mozilla.org](#)
- [Herencia en javascript en developer.mozilla.org](#)