

## **UD 07. CLASES Y OBJETOS EN JAVASCRIPT**

**Desarrollo web en entorno cliente  
CFGS DAW**

Sergio García Barea  
[sergio.garcia@ceedcv.es](mailto:sergio.garcia@ceedcv.es)  
2019/2020

Versión:191011.1322


## Licencia



**Reconocimiento - NoComercial - CompartirIgual (by-nc-sa):** No se permite un uso comercial de la obra original ni de las posibles obras derivadas, la distribución de las cuales se debe hacer con una licencia igual a la que regula la obra original.

## Nomenclatura

A lo largo de este tema se utilizarán distintos símbolos para distinguir elementos importantes dentro del contenido. Estos símbolos son:

 Importante

 Atención

 Interesante

## ÍNDICE DE CONTENIDO

<b>1. Introducción: Clases y objetos.....</b>	<b>3</b>
<b>2. Definiendo clases.....</b>	<b>3</b>
2.1 Más ejemplos.....	4
<b>3. Utilizando Class.....</b>	<b>7</b>
<b>4. Pasando variables, arrays y objetos a cadenas: JSON.....</b>	<b>7</b>
<b>5. Material adicional.....</b>	<b>8</b>
<b>6. Bibliografía.....</b>	<b>8</b>

## UD07. CLASES Y OBJETOS EN JAVASCRIPT

### 1. INTRODUCCIÓN: CLASES Y OBJETOS

Javascript es un lenguaje que permite el uso de objetos. Muchas veces el término clase y objeto se confunden. Una definición podría ser que una clase define “como es un objeto” y que un objeto es la plasmación efectiva de ese objeto. A partir de una clase se pueden crear si se desea muchos objetos.

Para entenderlo un ejemplo: supongamos tenemos la clase “casa”. Esa clase define que atributos tiene una casa. Un ejemplo de esos atributos podría ser: dirección, número habitaciones, metros cuadrados.

Ahora bien, cada objeto es una casa existente. Podemos tener por ejemplo dos objetos que surgen a partir de la clase “casa”. Uno sería una casa con dirección “Avenida del puerto 1, Valencia”, 3 habitaciones y 100m<sup>2</sup> y otro una casa con dirección “Calle Colón 1, Valencia”, 5 habitaciones y 200m<sup>2</sup>.

La clase definiría como podrían ser los objetos y los objetos en si son clases contextualizadas en algo concreto.

### 2. DEFINIENDO CLASES

La forma de definir clases en Javascript es ligeramente distinta a la utilizada en otros lenguajes de programación. Aquí la forma de definir una clase y su constructor asociado es simplemente definir una función.

```
function coche(marca, modelo, anyo, matricula) {  
    // Definimos una funcion para la clase  
    function mostrarCoche() {  
        var resultado = "Marca " + this.marca+ " modelo " +this.modelo;  
        alert(resultado);  
    }  
    // Definimos e inicializamos los atributos  
    this.marca=marca;  
    this.modelo=modelo;  
    this.anyo=anyo;  
    this.matricula=matricula;  
    // Asociamos la funcion definido antes al objeto  
    this.mostrarCoche=mostrarCoche;  
}
```

Esta clase la utilizaríamos como en este ejemplo:

### Ejemplo:

```
// Construimos los objetos a partir de la clase
var coche1=new coche("Seat","Ibiza",2000,"1234ABC");
var coche2=new coche("Ford","Mondeo",1999,"2235CNF");
// Ejecutamos el metodo en ambos objetos
coche1.mostrarCoche();
coche2.mostrarCoche();

// Mostramos la matricula del segundo objeto
alert(coche2.matricula);
```

## 2.1 Más ejemplos

Vamos a crear una clase Hotel, que tenga un array de elementos que serán de la clase habitación. Estos tendrán métodos para consulta, liberación y ocupación de habitaciones.

NOTA: Se aporta este ejemplo en formato editable como complemento al tema.

```
// Definimos la clase habitacion

function habitacion(id,m2,libre){
    // Atributos clase habitacion
    this.id=id;
    this.m2=m2;
    this.libre=libre;

    // Funciones clase habitacion

    function consultar(){
        if(this.libre==true){
            alert("Habitacion "+this.id+ " esta libre");
        } else {
            alert("Habitacion "+this.id+" esta ocupada");
        }
    }
    this.consultar=consultar;

    // Marca la habitacion como ocupada
```

```
function ocupar(){
    this.libre=false;
}
this.ocupar=ocupar;

// MARca la habitacion como libre
function liberar(){
    this.libre=true;
}
this.liberar=liberar;

// Codigo inicializacion (no hay en este ejemplo)
}

// Definimos la clase Hotel
function hotel(nombre,nhab) {
    // Atributos del hotel
    this.nombre=nombre;
    this.nhab=nhab;
    // Aqui guardaremos Array de habitaciones
    this.arrayHabs=new Array();

    // Definicion y asignacion de metodos de la clase

    // Marca la habitacion recibida como parametro como ocupada
    function ocuparHab(n){
        this.arrayHabs[n].ocupar();
    }
    this.ocuparHab=ocuparHab;

    // Marca la habitacion recibida como parametro como libre
    function liberarHab(n){
        this.arrayHabs[n].liberar();
    }
    this.liberarHab=liberarHab;
```

```
function consultarHab(n){
    this.arrayHabs[n].consultar();
}

this.consultarHab=consultarHab;

// Codigo inicializador del hotel
for(i=0;i<nhab;i++){
    // Creamos habitaciones con id i, 30 m2 y libres
    this.arrayHabs[i]=new habitacion(i,30,true);
}
}

// Creo un hotel
var miHotel=new hotel("CEED House",20);

//Ocupo 2 hab
miHotel.ocuparHab(1);
miHotel.ocuparHab(2);

// Consulta
miHotel.consultarHab(1);
miHotel.consultarHab(2);

// Libero la primera
miHotel.liberarHab(1);

// Consulta
miHotel.consultarHab(1);
miHotel.consultarHab(2);
```

### 3. UTILIZANDO CLASS

Con la versión de Javascript ES6 (ECMAScript 2015) podemos utilizar “class” para la definición de objetos.

La palabra reservada “class” es un tipo de función, pero en lugar de usar la palabra clave “function”, se utiliza la palabra “class” y las propiedades son asignadas usando el método “constructor()”.

Un ejemplo de uso:

```
class TarjetaFelicitacion {  
    // Constructor que se inicializa con un mensaje asociado  
    // a una tarjeta de felicitación  
    constructor(mensaje) {  
        this.mensajeTarjeta = mensaje;  
    }  
    // Recibe el nombre del destinatario y devuelve un mensaje personalizado  
    getMensaje(nombre) {  
        return x + ", tengo este mensaje para ti " + this.mensajeTarjeta ;  
    }  
}  
  
// Instanciamos la tarjeta con un mensaje  
miTarjeta = new TarjetaFelicitacion("Feliz cumple!");  
// Obtenemos el mensaje generado y lo metemos en el HTML de miDiv  
document.getElementById("miDiv").innerHTML = miTarjeta.present("Carlos");
```

Más información en [https://www.w3schools.com/js/js\\_classes.asp](https://www.w3schools.com/js/js_classes.asp)

### 4. PASANDO VARIABLES, ARRAYS Y OBJETOS A CADENAS: JSON

JSON es una notación para convertir variables, arrays y objetos en cadenas de texto y así poder facilitar la comunicación entre distintos programas (enviándose el contenido de un objeto como cadena de texto).

Más información sobre el formato en [http://www.w3schools.com/js/js\\_json.asp](http://www.w3schools.com/js/js_json.asp)

En este tema solo explicaremos su uso, con un fin introductorio. Será utilizado más adelante en los temas.

✈ JSON es uno de los formatos más utilizados para intercambiar información entre programas (cliente, servidor, API...).

En Javascript podemos usar JSON de la siguiente forma:

Para convertir un objeto a texto siguiendo el formato JSON:

```
textoJSON=JSON.stringify(objeto);
```

Para convertir una cadena de texto en JSON a un objeto usamos:

```
var objeto = JSON.parse(textoJSON);
```

### Ejemplo:

// Suponemos el objeto coche definido anteriormente en el apartado 2

```
// Construimos los objetos a partir de la clase
var coche1=new coche("Seat","Ibiza",2000,"1234ABC");
textoJSON=JSON.stringify(coche1);
alert(textoJSON);
// Reconstruimos el objeto y usamos un metodo para probarlo
var cocheReconstruido=JSON.parse(textoJSON);
cocheReconstruido.mostrarCoche();
```

## 5. MATERIAL ADICIONAL

[1] Curso de Javascript en Udacity <https://www.udacity.com/course/javascript-basics--ud804>

[2] Trabajando con objetos en Javascript  
[https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Trabajando\\_con\\_objetos](https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Trabajando_con_objetos)

## 6. BIBLIOGRAFÍA

[1] Referencia Javascript

<http://www.w3schools.com/jsref/>