DESARROLLO DE APLICACIONES WEB

12-11-2023

ROBERTO RODRÍGUEZ JIMÉNEZ

roberto.rodjim.1@educa.jcyl.es

Desarrollo Web en Entorno Cliente

Estructuras definidas por el usuario en JavaScript

Tarea 04

Contenido

[Tarea online DWEC02 2](#_Toc153196685)

[¿Qué contenidos o resultados de aprendizaje trabajaremos? 2](#_Toc153196686)

[Resultados de Aprendizaje 2](#_Toc153196687)

[Contenidos 2](#_Toc153196688)

[1.- Descripción de la tarea 3](#_Toc153196689)

[**Caso práctico** 3](file:///C:\xampp\htdocs\daw\dwec\rodriguez_jimenez_roberto_DWEC04_Tarea\rodriguez_jimenez_roberto_DWEC04_Tarea.docx#_Toc153196690)

[¿Qué te pedimos que hagas? 3](#_Toc153196691)

[2.- Información de interés 5](#_Toc153196692)

[Recursos necesarios 5](#_Toc153196693)

[Consejos y recomendaciones 5](#_Toc153196694)

[3.- Evaluación de la tarea 5](#_Toc153196695)

[Criterios de evaluación implicados 5](#_Toc153196696)

[¿Cómo valoramos y puntuamos tu tarea? 5](#_Toc153196697)

[RESOLUCIÓN 6](#_Toc153196698)

[Estructura de la página 6](#_Toc153196699)

[La clase Edificio 7](#_Toc153196700)

[El constructor 7](#_Toc153196701)

[Crear un objeto que nos permita instanciar edificios. 8](#_Toc153196702)

# Tarea online DWEC02

**Título de la tarea**: Tarea online para DWEC04  
**Unidad**: DWEC04  
**Ciclo formativo y módulo**: Desarrollo de Aplicaciones Web - Desarrollo web en entorno cliente  
**Curso académico**: 2020/2021

## ¿Qué contenidos o resultados de aprendizaje trabajaremos?

### Resultados de Aprendizaje

* RA4. Programa código para clientes Web analizando y utilizando estructuras definidas por el usuario.

### Contenidos

**Estructuras definidas por el usuario en JavaScript.**

1. Arrays.
   1. Trabajando con Arrays.
2. Funciones.
   1. Trabajando con funciones.
3. Objetos.
   1. Trabajando con objetos.

## 1.- Descripción de la tarea

### **Caso práctico**

Antonio ha avanzado mucho en su estudio de JavaScript, viendo los principales objetos con los que puede trabajar, sus métodos y propiedades.

Ha llegado el momento, de ver cómo puede organizar los datos que genera en su aplicación de JavaScript, de forma que le permita gestionarlos más eficientemente.

También verá cómo crear funciones, el alcance de las variables y cómo utilizarlas en su código. Y para terminar Juan, le va a explicar un poco más en detalle, cómo podrá crear nuevos objetos y asignarles propiedades y métodos en JavaScript.

Antonio ha comenzado a analizar más a fondo parte del trabajo que tiene que realizar, y comenta con su directora Ada, la posibilidad de hacer algunos bocetos de las innovaciones y mejoras que quiere aportar al proyecto. Ada está muy contenta con los progresos de Antonio y está deseando ver los bocetos.

### ¿Qué te pedimos que hagas?

Queremos hacer una aplicación en JavaScript para gestionar edificios con la información de la situación del edificio y de los propietarios de cada piso. Para ello queremos almacenar la siguiente información de cada edificio:

* calle.
* número.
* código postal.
* plantas del edificio (dentro de cada planta tendremos un número de puertas y para cada puerta almacenaremos el nombre del propietario).

Se pide:

* Crear un objeto que nos permita instanciar edificios. Cada vez que instanciemos un edificio le pasaremos la calle, número y código postal como parámetros. Se pide además crear los siguientes métodos para el objeto Edificio:
  + agregarPlantasYPuertas(numplantas, puertas) // Se le pasa el número de plantas que queremos crear en el piso y el número de puertas por planta. Cada vez que se llame a este método, añadirá el número de plantas y puertas indicadas en los parámetros, a las que ya están creadas en el edificio.
  + modificarNumero(numero) // Se le pasa el nuevo número del edificio para que lo actualice.
  + modificarCalle(calle) // Se le pasa el nuevo nombre de la calle para que lo actualice.
  + modificarCodigoPostal(codigo) // Se le pasa el nuevo número de código postal del edificio.
  + imprimeCalle // Devuelve el nombre de la calle del edificio.
  + imprimeNumero // Devuelve el número del edificio.
  + imprimeCodigoPostal // Devuelve el código postal del edificio.
  + agregarPropietario(nombre,planta,puerta) // Se le pasa un nombre de propietario, un número de planta y un número de puerta y lo asignará como propietario de ese piso.
  + imprimePlantas // Recorrerá el edificio e imprimirá todos los propietarios de cada puerta.
* Cada vez que se crea un edificio que muestre automáticamente un mensaje del estilo:
  + construido nuevo edificio en calle: xxxxxx, nº: xx, CP: xxxxx.
* Cada vez que se añada un propietario a un piso de un edificio que muestre un mensaje del estilo:
  + xxxxxxxx es ahora el propietario de la puerta x de la planta x.

Aquí se muestra un **ejemplo de lo que tendría que mostrar la aplicación**:

Trabajando con objetos Javascript:

Instanciamos 3 objetos edificioA, edificioB y edificioC con estos datos:

* Construido nuevo edificio en calle: Garcia Prieto, nº: 58, CP: 15706.
* Construido nuevo edificio en calle: Camino Caneiro, nº: 29, CP: 32004.
* Construido nuevo edificio en calle: San Clemente, nº: s/n, CP: 15705.
* El código postal del edificio A es: 15706.
* La calle del edificio C es: San Clemente.
* El edificio B está situado en la calle Camino Caneiro número 29.

Agregamos 2 plantas y 3 puertas por planta al edificio A...

Agregamos 4 propietarios al edificio A...

* Jose Antonio Lopez es ahora el propietario de la puerta 1 de la planta 1.
* Luisa Martinez es ahora el propietario de la puerta 2 de la planta 1.
* Marta Castellón es ahora el propietario de la puerta 3 de la planta 1.
* Antonio Pereira es ahora el propietario de la puerta 2 de la planta 2.

**Listado de propietarios del edificio calle García Prieto número 58**

* Propietario del piso 1 de la planta 1: Jose Antonio Lopez.
* Propietario del piso 2 de la planta 1: Luisa Martinez.
* Propietario del piso 3 de la planta 1: Marta Castellón.
* Propietario del piso 1 de la planta 2:
* Propietario del piso 2 de la planta 2: Antonio Pereira.
* Propietario del piso 3 de la planta 2:

Agregamos 1 planta más al edificio A...

Agregamos 1 propietario más al edificio A planta 3, puerta 2...

* Pedro Meijide es ahora el propietario de la puerta 2 de la planta 3.

**Listado de propietarios del edificio calle García Prieto número 58**

* Propietario del piso 1 de la planta 1: Jose Antonio Lopez.
* Propietario del piso 2 de la planta 1: Luisa Martinez.
* Propietario del piso 3 de la planta 1: Marta Castellón.
* Propietario del piso 1 de la planta 2:
* Propietario del piso 2 de la planta 2:
* Propietario del piso 1 de la planta 3:
* Propietario del piso 2 de la planta 3: Pedro Meijide.

## 2.- Información de interés

### Recursos necesarios

Editor web para teclear el código de la aplicación y un navegador web para ejecutar y probar la aplicación.

### Consejos y recomendaciones

Intenta crear todos los métodos dentro del objeto Edificio y utiliza los métodos programados para imprimir el valor de las propiedades que se piden en el enunciado.

## 3.- Evaluación de la tarea

### Criterios de evaluación implicados

* a) Se han clasificado y utilizado las funciones predefinidas del lenguaje.
* b) Se han creado y utilizado funciones definidas por el usuario.
* c) Se han reconocido las características del lenguaje relativas a la creación y uso de arrays.
* d) Se han creado y utilizado arrays.
* e) Se han reconocido las características de orientación a objetos del lenguaje.
* f) Se ha creado código para definir la estructura de objetos.
* g) Se han creado métodos y propiedades.
* h) Se ha creado código que haga uso de objetos definidos por el usuario.
* i) Se ha depurado y documentado el código.

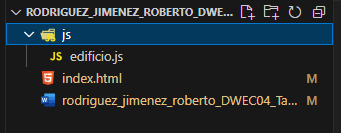
### ¿Cómo valoramos y puntuamos tu tarea?

|  |  |
| --- | --- |
| **Rúbrica de la tarea** | |
| Creación del objeto Edificio. | 1,5 puntos |
| Implmementación del método agregarPlantasYPuertas(numplantas, puertas) | 2 puntos |
| Implementación de los métodos: modificarNumero(numero), modificarCalle(calle), modificarCodigoPostal(codigo), imprimeCalle(), imprimeNumero(), imprimeCodigoPostal() | 0,5 puntos por método |
| Implementación del método agregarPropietario (nombre, planta, puerta) | 1 punto |
| Implementación del método imprimePlantas() | 1,5 puntos |
| Claridad y presentación del código del ejercicio, comentarios en el código y su indentación. | 1 punto |

## 

## RESOLUCIÓN

### Estructura de la página

El código html se limita a una lista que se va llenando con las tareas que se van ejecutando.  
El código JavaScript se ha separado en una hoja externa para hacerlo más legible. El archivo de encuentra en *js/edificio.js*.  
Se ha eliminado la hoja de estilo externa porque apenas se aplican estilos.

Sistema de archivos de la tarea

<!DOCTYPE html>

<html lang="es">

  <head>

    <meta charset="UTF-8" />

    <meta name="viewport" content="width=device-width, initial-scale=1.0" />

    <title>DWEC04 Tarea</title>

  </head>

  <body>

    <header style="text-align: center;">

      <h1>Gestión de edificios</h1>

    </header>

    <main>

      <ul id="tareas" style="list-style: none;"></ul>

    </main>

    <script src="js/edificio.js"></script>

  </body>

</html>

*Código de index.html*

class Edificio {

  constructor(calle, numero, cp = 1000) {}

  agregarPlantasYPuertas(plantas = 1, puertas = 1) {}

  modificarNumero(numero) {}

  modificarCalle(calle) {}

  modificarCodigoPostal(codigo) {}

  imprimeCalle() {}

  imprimeNumero() {}

  imprimeCodigoPostal() {}

  agregarPropietario(nombre, planta, puerta) {}

  imprimePlantas() {}

  mostrarMensajeDeEdificioConstruido(edificio) {}

  mostrarMensajeDePropietario(planta, puerta) {}

}

*Listado de funciones de edificio.js*

### La clase Edificio

Representa un edificio que tiene cuatro propiedades:

* La calle en la que está situado, de tipo *string.*
* El número del edificio, de tipo *string*.
  + El número del edificio se declara *string* dada la posibilidad de que el edificio no tenga número (s/n) o de que exista otro con el mismo número en la misma calle, por ejemplo n.º1 y n.º1 bis.
* El código postal es de tipo *number*.
* Las plantas son una colección puertas a las que se les asigna un propietario (*string*).
  + La estructura de las plantas es plantas[planta][puerta] = “propietario”.
  + Las plantas comienzan en 0, pues es lógico que la planta 1 sea la primera. De este modo, la planta 0 es la planta baja.

La clase no valida ni formatea datos, se limita a dar respuesta al enunciado del ejercicio.

Los nombres de las funciones de la clase se ajustan al enunciado del ejercicio.

#### El constructor

*constructor(calle, número, cp)*

Recibe como parámetros la calle, el número y el código postal. Podría recibir, también, las plantas y el número de puertas, pero se deja para hacerlo por separado.  
Los tres parámetros son obligatorios.

Después de asignar los valores a sus propiedades, se muestra el mensaje de que se ha construido el edificio. Para generar el mensaje se ha creado una función que lo imprime, liberando al constructor de esta tarea.

  constructor(calle, numero, cp) {

    this.\_calle = calle;

    this.\_numero = numero;

    this.\_cp = cp;

    this.\_plantas = [];

    this.mostrarMensajeDeEdificioConstruido(this);

  }

Constructor de la clase Edificio

 mostrarMensajeDeEdificioConstruido(edificio) {

    document.write(`<li>Construido un nuevo edificio en la calle ${edificio.\_calle}, n.º ${edificio.\_numero}, CP: ${edificio.\_cp}</li>`);

  }

Función que muestra el mensaje al crear un edificio

### Crear un objeto que nos permita instanciar edificios.

Cada vez que instanciemos un edificio le pasaremos la calle, número y código postal como parámetros.