

Callbacks

DEV.F
DESARROLLAMOS(PERSONAS);

Elaborado por: César Guerra

www.cesarquerra.mx

dev



OBJETIVOS

1. Entender qué son los Callbacks.
2. Usar Callbacks.
3. Aprender qué es JSON
4. Como manejar JSON



```
function saludar(nombre) {  
  alert('Hola ' + nombre);  
}  
  
function procesarEntradaUsuario(callback) {  
  var nombre = prompt('Por favor ingresa tu nombre.');
```


 callback(nombre);
}

procesarEntradaUsuario(saludar);

¿Que es un Callback? (llamada de vuelta)

- Es una función que recibe como parámetro otra función y la ejecuta.
- La función “callback” por lo regular va a realizar algo con los resultados de la función que la está ejecutando.
- Es una forma de ejecutar código de forma “asíncrona” ya que una función va a llamar a otra.
- Cuando pasamos un callback solo pasamos la definición de la función y no la ejecutamos en el parámetro. Así, la función contenedora elige cuándo ejecutar el callback.


Ejemplo de Callback

```
/* Tenemos 2 funciones que devuelven un valor */
function soyCien() { return 100; }
function SoyDoscientos() { return 200; }

/* Esta función recibe como parametro 2 funciones y las ejecuta */
function sumaDosFunciones(functionOne, functionTwo) {
  const suma = functionOne() + functionTwo();
  return suma; // retornando un nuevo valor, en este caso su suma
}

/* Invocamos a sumaDosFunciones y le pasamos 2 funciones como parámetros */
console.log(sumaDosFunciones(soyCien, SoyDoscientos));
// Resultado → 300
```

Ejemplo setTimeout



```
setTimeout(function() {  
  console.log("He ejecutado la función");  
}, 2000);
```

Le decimos a `setTimeout()` que ejecute la función callback que le hemos pasado por primer parámetro cuando transcurran 2000 milisegundos (es decir, 2 segundos, que le indicamos como segundo parámetro).

Ventajas y desventajas de los callbacks

Ventajas

- Son fáciles de usar.
- Pueden solucionar problemas de flujo de una aplicación
- Ayudan a manejar excepciones.
- Son útiles cuando quieres hacer consultas a una BD o servicio web

Desventajas

- A Veces el concepto es confuso
- Si se usa demasiado se puede caer en algo denominado “callback hell”
- El uso excesivo puede afectar el performance.
- Para programadores novatos no es muy fácil leer y entender qué hacen las funciones callback.

JSON

DEV.F
DESARROLLAMOS(PERSONAS);

dev



JSON

JSON, que significa JavaScript Object Notation, es un formato ligero de intercambio de datos que ha mejorado la forma en que las aplicaciones web modernas se comunican.

Se ha consolidado como la opción predilecta para la transferencia de datos, siendo utilizado en más del 95% de las aplicaciones web contemporáneas.

Su eficiencia y simplicidad lo han posicionado como una alternativa superior a formatos más complejos como XML.

CARACTERÍSTICAS DE JSON



FORMATO DE TEXTO SENCILLO

JSON es un formato de texto simple, lo que facilita su lectura y escritura tanto para humanos como para máquinas.



BASADO EN JAVASCRIPT

Fue desarrollado a partir de la notación de objetos literales de JavaScript, pero es independiente de cualquier lenguaje de programación.



ABIERTO Y ESTANDARIZADO

JSON es un formato abierto y estandarizado, lo que garantiza su compatibilidad y uso universal en diversas plataformas y tecnologías.

XML

```
<empinfo>
  <employees>
    <employee>
      <name>James Kirk</name>
      <age>40</age>
    </employee>
    <employee>
      <name>Jean-Luc Picard</name>
      <age>45</age>
    </employee>
    <employee>
      <name>Wesley Crusher</name>
      <age>27</age>
    </employee>
  </employees>
</empinfo>
```

JSON

```
{ "empinfo" :
  {
    "employees": [
      {
        "name": "James Kirk",
        "age": 40,
      },
      {
        "name": "Jean-Luc Picard",
        "age": 45,
      },
      {
        "name": "Wesley Crusher",
        "age": 27,
      }
    ]
  }
}
```

JSON vs XML

JSON se concibió como una alternativa más eficiente y directa a XML para el intercambio de datos.

Su diseño intuitivo permite una fácil interpretación y procesamiento de la información.

Así mismo al ocupar menos caracteres, es mucho más liviano que XML en tamaño total del archivo.

Estructura de JSON

PARES ATRIBUTO-VALOR

Los datos en JSON se organizan en pares de atributos (claves) y valores, separados por dos puntos.

DELIMITADORES ESPECÍFICOS

Utiliza llaves `{}` para objetos, corchetes `[]` para arrays y comillas dobles `""` para cadenas de texto.

TIPOS DE DATOS SOPORTADOS

Soporta strings, números (enteros y flotantes), booleanos (true/false), arrays y objetos anidados.

Ejemplo de JSON

```
{
  "nombre": "Alice",
  "edad": 30,
  "esEstudiante": true,
  "cursos": ["Matemáticas", "Historia", "Programación"],
  "direccion": {
    "calle": "Calle Falsa 123",
    "ciudad": "Springfield",
    "codigoPostal": "12345"
  }
}
```

Este ejemplo ilustra una estructura JSON típica que representa la información de un usuario. Podemos observar cómo se definen pares de clave-valor, el uso de un array para la lista de cursos y un objeto anidado para la dirección.

JSON.parse()



```
const firstString = `
{
  "name": "César",
  "description": "Developer"
}
`

console.log(JSON.parse(firstString))
//output
//{name: 'César', description: 'Developer'}
```

JSON.stringify()



```
const firstArray = [1, 2, 3, 5, 7, 11]
console.log(JSON.stringify(firstArray))
// Output
// '[1, 2, 3, 5, 7, 11]'
```

Conversion de JSON

JSON.parse() y JSON.stringify()

JSON.parse() y **JSON.stringify()** son métodos esenciales en JavaScript para trabajar con datos en formato JSON.

JSON.parse() convierte una cadena JSON en un objeto de JavaScript, permitiendo su manipulación dentro del código, mientras que **JSON.stringify()** hace lo contrario: toma un objeto de JavaScript y lo transforma en una cadena JSON, facilitando su almacenamiento o transmisión en APIs.

Juntos, permiten el intercambio fluido de datos entre servidores y clientes, manteniendo la integridad del formato JSON en las interacciones web.

Práctica de la Clase

DEV.F
DESARROLLAMOS(PERSONAS);

dev

Gestión de una Biblioteca de Libros

El objetivo es crear una pequeña aplicación de consola que permita realizar las siguientes tareas:

1. **Consultar libros:** Mostrar todos los libros almacenados en formato JSON.
2. **Agregar libros:** Permitir al usuario agregar un libro a la colección.
3. **Actualizar la disponibilidad:** Cambiar el estado de disponibilidad de un libro a 'prestado' o 'disponible'.
4. **Simular un archivo JSON (Opcional):** Aunque no vas a leer/escribir realmente en un archivo, simularás la lectura y escritura de datos usando callbacks, como si interactuaras con un sistema de almacenamiento.

Problema: Gestión de una Biblioteca de Libros

Imagina que eres parte del equipo de desarrollo de una pequeña biblioteca local que ha decidido construir una aplicación para gestionar su inventario de libros. La biblioteca desea almacenar la información sobre los libros, como el título, autor, género y si está disponible o prestado. La biblioteca también quiere ofrecer la opción de consultar el inventario de libros, agregar nuevos libros, y actualizar la disponibilidad de los libros cuando son prestados o devueltos. Para hacer esto, utilizarás JSON para almacenar los datos de los libros y callbacks para manejar las tareas asíncronas, como la lectura y escritura de los datos.

Enlace Campus:

https://edu.devf.la/campus/program/module/desarrollo_avanzado_javascript/callback-json/project/callback-json-project

Código Inicial:

<https://gist.github.com/heladio-devf-mx/b7f9cd1ffe11e2fd9a46cba6ef019ea1>