Sesión 12 – Semana 4

JSON-Server y Async Await

WWW.MAKAIA.ORG

Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia

Contenido

1. Async/Await
2. JSON-Server
    1. Archivos `JSON`
    2. `JSON-SERVER`

Async/Await

¿Qué son?

Una forma de **azúcar sintáctico** para gestionar las promesas de una forma más sencilla.

Una forma de gestionar o consumir promesas, abandonando el modelo de encadenamiento de `.then()`

Async

Palabra clave que se coloca previamente a `function`, para definirla como una función asíncrona.

```
async function funcion_asincrona() {
  return 42;
}
```

```
const funcion_asincrona = async () => 42;
```

La función devuelve una `PROMISE` que ha sido cumplida, con el valor devuelto en la función (en este caso, 42). De hecho, podríamos utilizar un `.then()` para consumir la promesa:

```
funcion_asincrona().then(value => {
  console.log("El resultado devuelto es: ", value);
});
```

Await

Palabra clave que permite manejar promesas. Lo que hace **`await`** es esperar a que se resuelva la promesa, mientras permite continuar ejecutando otras tareas que puedan realizarse:

```
const funcion_asincrona = async () => 42;

const value = funcion_asincrona();             // Promise { <fulfilled>: 42 }
const asyncValue = await funcion_asincrona();  // 42
```

Con `await` se deja de utilizar `.then()`, para esperar la resolución de la promesa,

Formato de archivo JSON

¿Qué son?

JSON es la abreviatura de la notación de objetos de JavaScript (**JavaScript Object Notation**).

Formato ligero de datos basado en texto que sigue la sintaxis de objeto de JavaScript.

Su contenido puede ser simplemente ARRAY , NUMBER , STRING , BOOLEAN o incluso NULL , sin embargo, lo más habitual es que parta siendo OBJECT o ARRAY.

Formato de archivo JSON

Características

```
{
}
```

Estructura mínima: Objeto vacío

```
{
  "name": "Manz",
  "life": 3,
  "totalLife": 6
  "power": 10,
  "dead": false,
  "props": ["invisibility", "coding", "happymood"],
  "senses": {
    "vision": 50,
    "audition": 75,
    "taste": 40,
    "touch": 80
  }
}
```

- Las propiedades del objeto deben estar entrecomilladas con «comillas dobles»
- Los textos `STRING` deben estar entrecomillados con «comillas dobles»
- Sólo se puede almacenar tipos como `ARRAY` , `NUMBER` , `STRING` , `BOOLEAN` o `null`.
- Tipos de datos como `FUNCTION`, `DATE`, `REGEXP` u otros, no es posible almacenarlos en un JSON.
- Tampoco es posible añadir comentarios en un JSON. Hay formatos derivados que sí lo permiten como JSON5.

Formato de archivo JSON

¿Cómo utilizar?

En JavaScript existen una serie de **métodos** que nos facilitan la tarea de pasar de **`OBJECT`** a **JSON** y viceversa, pudiendo trabajar con contenido de tipo (*que contenga un JSON*) y objetos Javascript según interese.

Método	Descripción
OBJECT `JSON.parse(str)`	Convierte el texto STRING `str` (si es un JSON válido) a un objeto y lo devuelve.
STRING `JSON.stringify(obj)`	Convierte un objeto Javascript OBJECT `obj` a su representación JSON y la devuelve.
STRING `JSON.stringify(obj, props)`	Idem al anterior, pero filtra y mantiene solo las propiedades del ARRAY `props`.
STRING `JSON.stringify(obj, props, spaces)`	Idem al anterior, pero indenta el JSON a NUMBER `spaces` espacios.

Formato de archivo JSON

Ejemplo

```
const user = {
  name: "Manz",
  life: 99,
  power: 10,
};

JSON.stringify(user, ["life"])            // '{"life":99}'
JSON.stringify(user, ["name", "power"])   // '{"name":"Manz","power":10}'
JSON.stringify(user, [])                  // '{}' (filtra todo)
JSON.stringify(user, null)                // '{"name":"Manz","life":99,"power":10}'
```

JSON-SERVER

¿Qué es?

Es un paquete de funcionalidades o librería para entornos de desarrollo y prueba que permite crear API REST (Transferencia de estado representacional, por las siglas en español) de manera rápida y sencilla, es decir, crear interfaces (o aplicaciones) que permiten realizar transferencia o interacción con datos mediante peticiones HTTPS que modifican el estado de esos datos.

Resúmen: con el JSON-SERVER nos permite obtener una “full fake REST API” con cero codificación en menos de 30 seg.

JSON-SERVER

Instalación

1. Abrir una nueva consola o terminal del VSC
2. En la consola, escribir los siguientes comandos

Instalación global

```
npm install -g json-server
```

Instalación local

```
npm i json-server
```

JSON-SERVER

Instalación

3. En la carpeta abierta en el VSC, crear un archivo `db.json` con la siguiente data:

```
{
  "posts": [
    { "id": 1, "title": "json-server", "author": "typicode" }
  ],
  "comments": [
    { "id": 1, "body": "some comment", "postId": 1 }
  ],
  "profile": { "name": "typicode" }
}
```

JSON-SERVER

Instalación

4. Luego, regresar a la terminal del VSC y asegurarse que la ruta indicada en la terminal, corresponda a la carpeta donde se encuentra el archivo JSON.
5. Ejecutar el siguiente comando:

```
json-server --watch db.json
```

JSON-SERVER

Instalación

6. El servidor enciende en el puerto 3000 en **http://localhost:3000/posts/1**

```
$ json-server --watch db.json

  \{^_^}/ hi!

  Loading db.json
  Done

  Resources
  https://localhost:3000/employees

  Home
  https://localhost:3000

  Type s + enter at any time to create a snapshot of the database
  Watching...
```

JSON-SERVER

Instalación

7. Si deseamos utilizar un puerto diferente, en lugar del comando del punto 5 se debe ejecutar el siguiente:

```
json-server --watch db.json --port 3004
```


Carrera 43 A # 34 - 155. Almacentro. Torre Norte. Oficina 701
Medellín (Antioquia), Colombia

MAKAIA
WWW.MAKAIA.ORG

Fuente

- https://lenguajejs.com/javascript/objetos/json/
- https://www.npmjs.com/package/json-server
- https://lenguajejs.com/javascript/asincronia/async-await/

MAKAIA

WWW.MAKAIA.ORG

Info: comunicaciones@makaia.org

Corporación MAKAIA
Medellín, Colombia
Carrera 43A – 34-155. Almacentro
Torre Norte, Oficina 701
Teléfono: (+574) 448 03 74
Móvil: (+57) 320 761 01 76

@makaiaorg