JSON y Fetch API

Bootcamp Desarrollo Web Full Stack

16 de octubre de 2024

Agenda

- Introducción
- JSON (JavaScript Object Notation)
- Fetch API
- Conclusión

Introducción

- JSON (JavaScript Object Notation)
  - Formato ligero de intercambio de datos
- Fetch API
  - Interfaz moderna para peticiones HTTP asíncronas
- Fundamentales para el desarrollo web moderno

¿Qué es JSON?

- Formato de texto sencillo para intercambio de datos
- Independiente del lenguaje de programación
- Basado en dos estructuras:
  - Colección de pares nombre/valor (objeto)
  - Lista ordenada de valores (array)

Sintaxis de JSON

```json
{
  "nombre": "Juan",
  "edad": 30,
  "ciudad": "Madrid",
  "casado": false,
  "hobbies": ["lectura", "natación", "viajes"],
  "trabajo": {
    "puesto": "Desarrollador",
    "empresa": "TechCo"
  }
}
```

Tipos de datos en JSON

- String: `"Hola Mundo"`
- Number: `42` o `3.14`
- Boolean: `true` o `false`
- Array: `[1, 2, 3]`
- Object: `{"nombre": "Juan", .edad": 30}`
- null: `null`

Trabajando con JSON en JavaScript

```
// JSON.parse()
const jsonString = '{"nombre": "Ana", "edad": 25}';
const objeto = JSON.parse(jsonString);
console.log(objeto.nombre); // Ana

// JSON.stringify()
const persona = {
  nombre: "Carlos",
  edad: 35,
  ciudad: "Barcelona"
};
const jsonPersona = JSON.stringify(persona);
console.log(jsonPersona);
// {"nombre":"Carlos","edad":35,"ciudad":"Barcelona"}
```

Introducción a Fetch API

- Interfaz JavaScript para acceder y manipular partes del canal HTTP
- Proporciona el método global `fetch()`
- Basada en promesas
- Simplifica el código para peticiones HTTP
- Alternativa moderna a AJAX

Sintaxis básica de Fetch

```
fetch('https://api.ejemplo.com/datos')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Opciones de Fetch

```
fetch('https://api.ejemplo.com/datos', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify({ nombre: 'Juan', edad: 30 })
})
.then(response => response.json())
.then(data => console.log(data))
.catch(error => console.error('Error:', error));
```

Manejando respuestas

- `response.json()`
- `response.text()`
- `response.blob()`
- `response.formData()`
- `response.arrayBuffer()`

Manejo de errores

```
fetch('https://api.ejemplo.com/datos')
  .then(response => {
    if (!response.ok) {
      throw new Error('Error HTTP: ' + response.status
          );
    }
    return response.json();
  })
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

Conclusión

- JSON: formato ligero y fácil de usar para intercambio de datos
- Fetch API: forma poderosa y flexible de realizar peticiones HTTP
- Ambos son fundamentales para el desarrollo de aplicaciones web modernas
- Facilitan la interacción con APIs

¿Preguntas?