1. Concepto básico de JSON

JSON es un formato de texto que representa datos estructurados como objetos, listas, valores de texto, números, etc. Es muy similar a los objetos en JavaScript, pero no incluye métodos o funciones. Aquí tienes un ejemplo simple de cómo se ve un archivo JSON:

```json
{
  "name": "Juan",
  "age": 30,
  "isStudent": true,
  "skills": ["JavaScript", "HTML", "CSS"],
  "address": {
    "street": "Calle Falsa 123",
    "city": "Madrid",
    "zip": "28001"
  }
}
```

2. Crear y Guardar un archivo JSON

Para crear un archivo JSON, simplemente escribe los datos en formato JSON y guarda el archivo con la extensión .json. Por ejemplo, crea un archivo llamado data.json:

```json
{
  "name": "Juan",
  "age": 30,
  "isStudent": true,
  "skills": ["JavaScript", "HTML", "CSS"]
}
```

Puedes guardar este archivo en el mismo directorio de tu proyecto para acceder a él desde JavaScript.

3. Cargar datos JSON en JavaScript (de forma manual)

Supongamos que tienes un objeto JSON en tu archivo JavaScript. Puedes definirlo como una cadena de texto y luego convertirlo en un objeto JavaScript usando JSON.parse(). Aquí tienes un ejemplo básico:

```
const jsonData = `{
  "name": "Juan",
  "age": 30,
  "isStudent": true,
  "skills": ["JavaScript", "HTML", "CSS"]
}`;

// Convertimos la cadena JSON en un objeto JavaScript
const data = JSON.parse(jsonData);

console.log(data.name); // "Juan"
console.log(data.skills[0]); // "JavaScript"
```

4. Usar Fetch API para obtener datos de un archivo JSON

Una vez que tienes un archivo JSON, puedes utilizar fetch() para cargar y procesar los datos en JavaScript. La función fetch() es asíncrona, por lo que usaremos .then() para manejar las respuestas.

```
fetch('data.json')
  .then(response => response.json()) // Convertimos la respuesta en JSON
  .then(data => {
    console.log(data.name); // "Juan"
    console.log(data.skills); // ["JavaScript", "HTML", "CSS"]
  })
  .catch(error => console.error('Error:', error));
```

5. Crear y Manipular un Objeto JSON en JavaScript

También puedes crear un objeto JSON directamente en tu código y convertirlo en una cadena JSON usando JSON.stringify(). Esto es útil si quieres preparar datos para enviar a un servidor.

```
const user = {
  name: "Ana",
  age: 25,
  skills: ["Python", "React", "Node.js"]
};

// Convertir el objeto JavaScript a JSON
const jsonString = JSON.stringify(user);
console.log(jsonString);
```

6. Actualizar datos en un objeto JSON

Una vez que tienes un objeto JSON en JavaScript, puedes actualizarlo fácilmente como lo harías con cualquier objeto JavaScript.

```
const user = {
  name: "Ana",
  age: 25,
  skills: ["Python", "React", "Node.js"]
};

// Cambiamos la edad y añadimos una nueva habilidad
user.age = 26;
user.skills.push("JavaScript");

console.log(user);
```

7. Escribir en un archivo JSON (con Node.js)

Si trabajas en un entorno de servidor (por ejemplo, con Node.js), puedes escribir en un archivo JSON utilizando el módulo fs (file system). Esto no se puede hacer directamente en el navegador por razones de seguridad, pero es útil en entornos de desarrollo backend.

```
const fs = require('fs');

const user = {
  name: "Ana",
  age: 26,
  skills: ["Python", "React", "Node.js", "JavaScript"]
};

// Convertimos el objeto a JSON y lo guardamos en un archivo
fs.writeFileSync('user.json', JSON.stringify(user, null, 2), 'utf-8');
```

En el ejemplo anterior, JSON.stringify(user, null, 2) convierte el objeto en una cadena JSON con formato, y fs.writeFileSync() escribe esa cadena en un archivo llamado user.json.

8. Enviar datos JSON a un servidor con fetch

Supongamos que quieres enviar datos JSON a un servidor (por ejemplo, a una API). Puedes hacer esto usando fetch() con el método POST y configurando las cabeceras adecuadamente.

```
const userData = {
  name: "Carlos",
  age: 28,
  skills: ["Angular", "JavaScript", "TypeScript"]
};

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

9. Ejemplo Completo: Cargar, Modificar, y Guardar JSON en el Navegador

Aunque el navegador no permite escribir directamente en archivos locales, puedes simular esto almacenando los datos temporalmente en localStorage o en una variable.

```
// Simulamos cargar datos JSON desde un archivo
const data = {
  name: "Laura",
  age: 32,
  skills: ["Vue.js", "CSS", "HTML"]
};

// Modificamos los datos
data.age = 33;
data.skills.push("JavaScript");

// Guardamos el JSON en localStorage (solo en el navegador)
localStorage.setItem('userData', JSON.stringify(data));

// Para cargar los datos de nuevo
const loadedData = JSON.parse(localStorage.getItem('userData'));
console.log(loadedData);
```

Ejemplo Complejo de JSON de Datos de Usuarios


```json
{
  "users": [
    {
      "nombre": "Carlos",
      "primerApellido": "Martínez",
      "segundoApellido": "García",
      "direccion": {
        "calle": "Calle Mayor",
        "numero": "123",
        "piso": "3ºB",
        "codigoPostal": "28001",
        "localidad": "Madrid",
        "provincia": "Madrid",
        "pais": "España"
      },
      "telefono": "+34 600 123 456",
      "email": "carlos.martinez@example.com",
      "imagenPerfil": "/images/users/carlos_martinez.png",
      "descripcion": "<p>Desarrollador web con más de 5 años de experiencia en <strong>JavaScript</strong>
      "hobbies": ["Fútbol", "Ciclismo", "Lectura", "Viajar"],
      "experienciaLaboral": [
        {
          "empresa": "Tech Solutions",
          "puesto": "Desarrollador Full Stack",
          "fechaInicio": "2018-05-01",
          "fechaFin": "2021-08-15",
          "descripcion": "Desarrollo de aplicaciones web y APIs utilizando Node.js y React."
        },
        {
          "empresa": "Innovatech",
          "puesto": "Ingeniero de Software",
          "fechaInicio": "2021-09-01",
          "fechaFin": "presente",
          "descripcion": "Lidero proyectos de aplicaciones móviles y web con un equipo de desarrolladores.
        }
      ],
      "redesSociales": {
        "linkedin": "https://www.linkedin.com/in/carlosmartinez",
        "github": "https://github.com/carlosmartinez",
        "twitter": "https://twitter.com/carlosm_dev"
      }
    },
    {
      "nombre": "Ana",
      "primerApellido": "López",
      "segundoApellido": "Fernández",
      "direccion": {
```

Explicación de la Estructura

1. **Raíz (users)**: El objeto principal tiene una propiedad users, que es un array de objetos. Cada objeto dentro del array representa un usuario con su propia información.
2. **Datos Básicos**: Dentro de cada usuario, están las propiedades de nombre, primerApellido, segundoApellido, telefono, email, y imagenPerfil.
3. **Dirección Anidada**: La dirección (direccion) es un objeto anidado que contiene propiedades como calle, numero, piso, codigoPostal, localidad, provincia, y pais. Esto facilita el acceso y la modificación de la dirección completa o de partes específicas.
4. **Descripción con Etiquetas HTML**: La propiedad descripcion incluye una descripción con etiquetas HTML. Esto permite formatear el texto, utilizando etiquetas como , , y <p> para darle estilo.
5. **Hobbies (Array)**: La propiedad hobbies es un array de cadenas, que almacena varios intereses de cada usuario.
6. **Experiencia Laboral (Array de Objetos)**: La experiencia laboral (experienciaLaboral) es un array que contiene objetos. Cada objeto tiene detalles sobre una experiencia específica, como empresa, puesto, fechaInicio, fechaFin, y descripcion.
7. **Redes Sociales (Objeto)**: Las redes sociales (redesSociales) están organizadas en un objeto, donde cada propiedad representa una red social específica y su valor es la URL del perfil del usuario en esa red.

Cómo Usar Este JSON en JavaScript

Una vez tengas este JSON en un archivo (por ejemplo, users.json), puedes cargarlo en JavaScript de la siguiente manera:

```
fetch('users.json')
    .then(response => response.json())
    .then(data => {
    // Accedemos al primer usuario
    const user1 = data.users[0];

    console.log("Nombre completo:", `${user1.nombre} ${user1.primerApellido} ${user1.segundoApellido}`);
    console.log("Teléfono:", user1.telefono);
    console.log("Ciudad:", user1.direccion.localidad);
    console.log("Hobbies:", user1.hobbies.join(', '));
    console.log("Experiencia Laboral más reciente:", user1.experienciaLaboral[0].empresa);

    // Mostrar una descripción con HTML en una página
    document.getElementById("descripcion").innerHTML = user1.descripcion;

    // Acceso a redes sociales
    console.log("LinkedIn:", user1.redesSociales.linkedin);
    console.log("GitHub:", user1.redesSociales.github);
})
.catch(error => console.error('Error al cargar los datos:', error));
```

Enunciado del Ejercicio: Creación de una Página de Perfil Completa con JSON y JavaScript

Objetivo

Desarrollar una página web de "Perfiles de Usuario" que cargue información desde uno o más archivos JSON y muestre los datos en formato visual utilizando HTML, CSS y JavaScript vanilla. La página debe mostrar perfiles individuales de usuarios, con secciones que incluyan datos personales, una lista de hobbies, una sección de experiencia laboral y enlaces a redes sociales.