Handling JSON data

📦 Handling JSON Data in JavaScript — Explained in Detail

🧾 What is JSON?

JSON (JavaScript Object Notation) is a lightweight **data-interchange format**. It is:

- Easy to read and write for humans.
- Easy to parse and generate for machines.
- Commonly used for sending data between a server and a client.

🔑 JSON Structure Example

```
{
  "name": "Abhi",
  "age": 25,
  "skills": ["JavaScript", "React", "Node.js"],
  "isActive": true
}
```

In JavaScript, this is just an object with key-value pairs.

🔄 Converting Between JSON and JavaScript

✅ `JSON.stringify()` — JS → JSON string

Used to **convert JavaScript objects or arrays into JSON strings**.

```
const user = {
  name: "Abhi",
  age: 25
};
```

```
const jsonString = JSON.stringify(user);
console.log(jsonString); // '{"name":"Abhi","age":25}'
```

> Useful for storing or sending data (e.g., in localStorage, HTTP requests).

✅ `JSON.parse()` — JSON string → JS

Used to **convert a JSON-formatted string into a JavaScript object**.

```
const jsonData = '{"name":"Abhi","age":25}';
const userObj = JSON.parse(jsonData);

console.log(userObj.name); // "Abhi"
```

> Commonly used to parse server responses.

🧪 Example: Sending and Receiving JSON via Fetch

```
// POST request
fetch("https://api.example.com/users", {
  method: "POST",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify({
    name: "Abhi",
    age: 25
  })
})
  .then(response => response.json())
  .then(data => console.log("Created:", data))
  .catch(error => console.error("Error:", error));
```

> 🔍 .json() is used to parse the response body as JSON.

🔗 JSON in Local Storage

```
const settings = { theme: "dark", fontSize: 16 };

// Store in localStorage
localStorage.setItem("settings", JSON.stringify(settings));

// Retrieve from localStorage
const stored = JSON.parse(localStorage.getItem("settings"));
console.log(stored.theme); // "dark"
```

🛑 Common Errors

Issue	Cause
`Unexpected token`	Trying to parse malformed JSON
Circular structure error	Using `JSON.stringify` on objects with circular references
Numbers as strings	JSON doesn't store types like JS

Example:

```
const badJSON = "{ name: 'Abhi' }"; // invalid JSON
JSON.parse(badJSON); // ❌ SyntaxError: Unexpected token n
```

🎯 Best Practices

- Always wrap `JSON.parse()` in try-catch.
- Always stringify objects before sending/storing.
- Avoid functions, `undefined`, and circular references in objects you want to stringify.
- Use tools like https://jsonlint.com to validate JSON.

✅ Summary

Function	Purpose
`JSON.stringify()`	JS → JSON string
`JSON.parse()`	JSON string → JS object
`response.json()`	Convert fetch response to JS object

JSON is essential for working with APIs, localStorage, and modern data handling in JavaScript.

Let me know if you'd like hands-on exercises or a JSON mini-project (like a note-taking app using `localStorage`).