AAKASH
aakash-n0dev

Handling JSON

JAVASCRIPT

Introduction to JSON

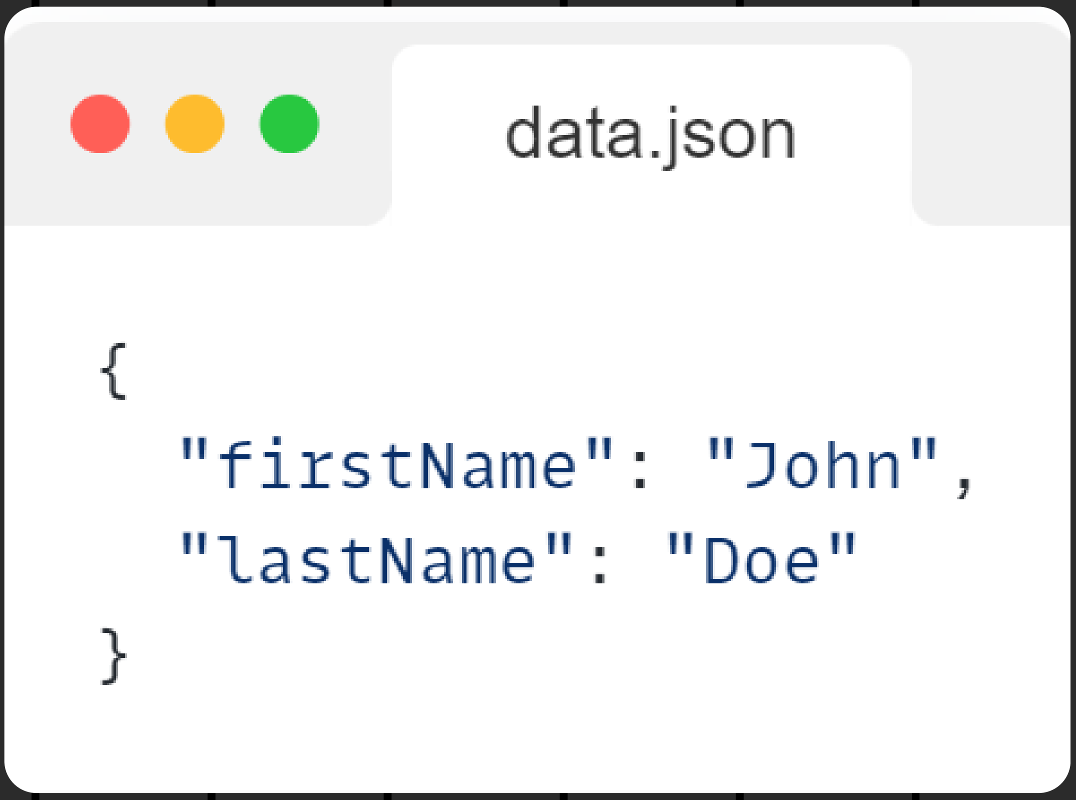
- JSON (JavaScript Object Notation) is a lightweight data interchange format used for storing and transporting data.
- It’s often used when data is sent from a server to a web page.
- JSON is language-independent, self-describing, and easy to understand.

JSON Syntax

- JSON data consists of name/value pairs, similar to JavaScript object properties.
- A name/value pair includes a field name (in double quotes), followed by a colon and the corresponding value.
- Unlike JavaScript, JSON names require double quotes.

JSON Objects

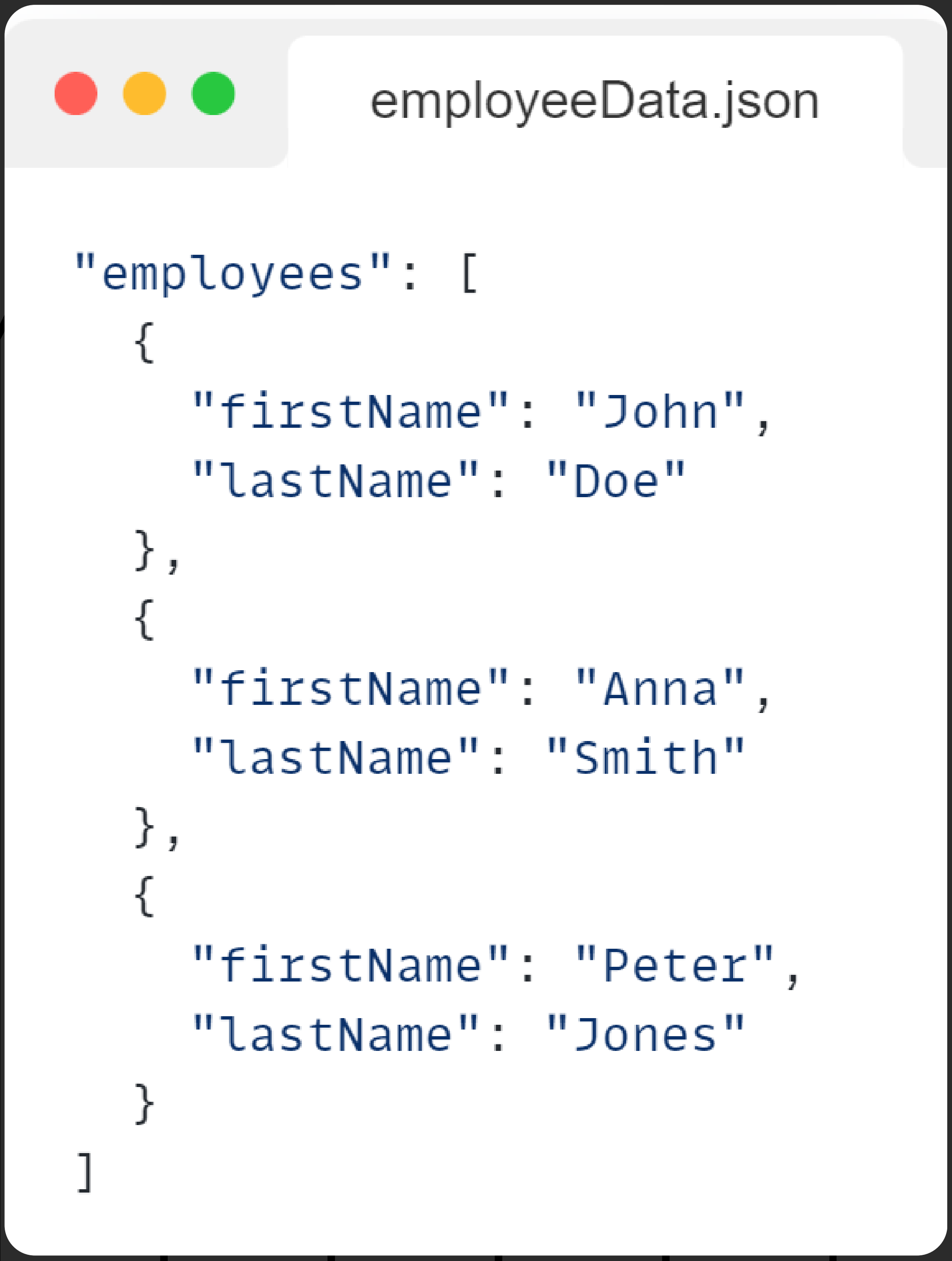
- JSON objects are enclosed in curly braces {}.
- Objects can contain multiple name/value pairs.



```
{
  "firstName": "John",
  "lastName": "Doe"
}
```

JSON Arrays

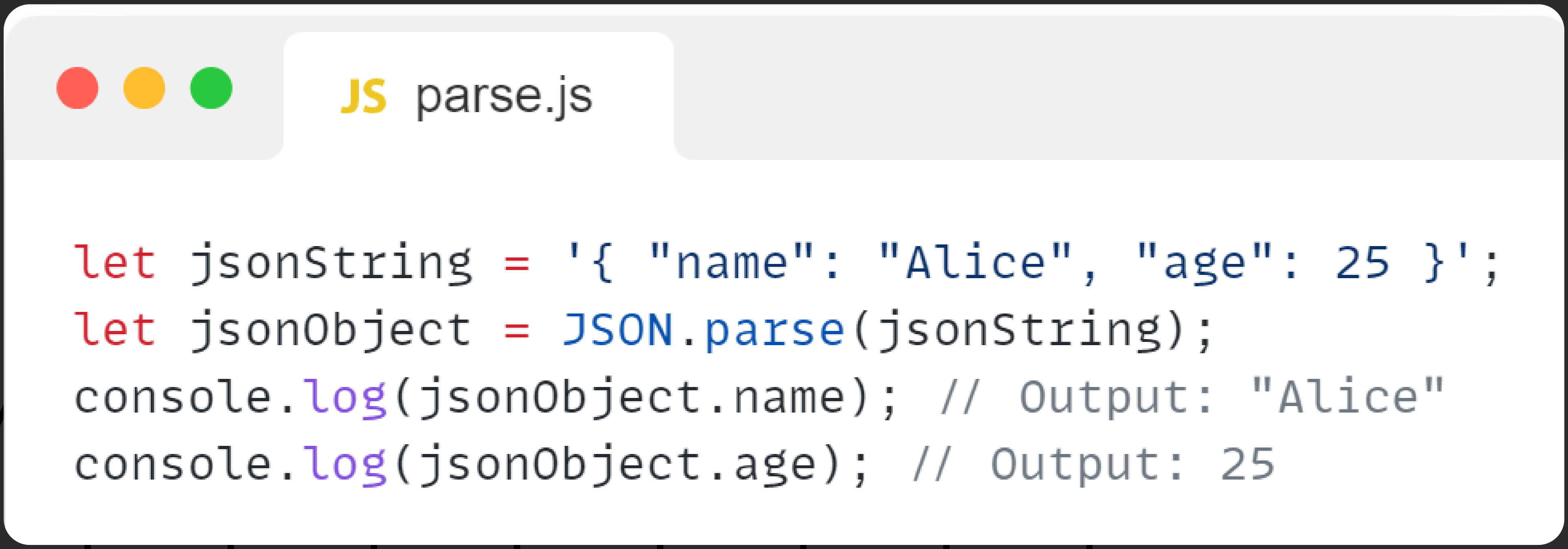
- JSON arrays are enclosed in square brackets [].
- An array can contain objects.



```
"employees": [
  {
    "firstName": "John",
    "lastName": "Doe"
  },
  {
    "firstName": "Anna",
    "lastName": "Smith"
  },
  {
    "firstName": "Peter",
    "lastName": "Jones"
  }
]
```

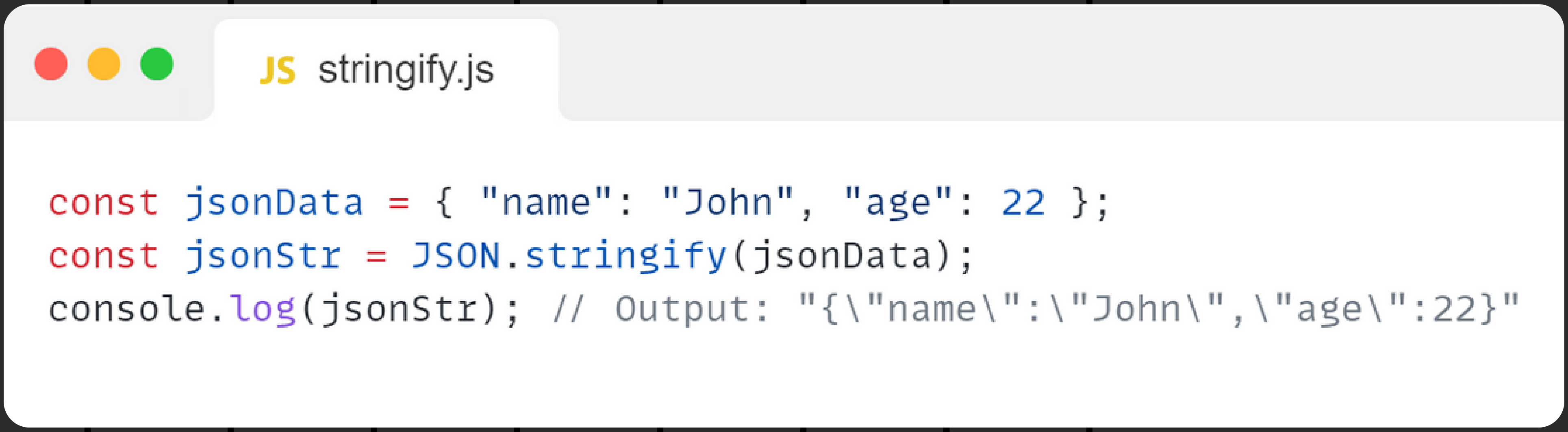
Converting JSON to Js Object & Js Object to JSON

To convert a JSON string to a JavaScript object, use JSON.parse().



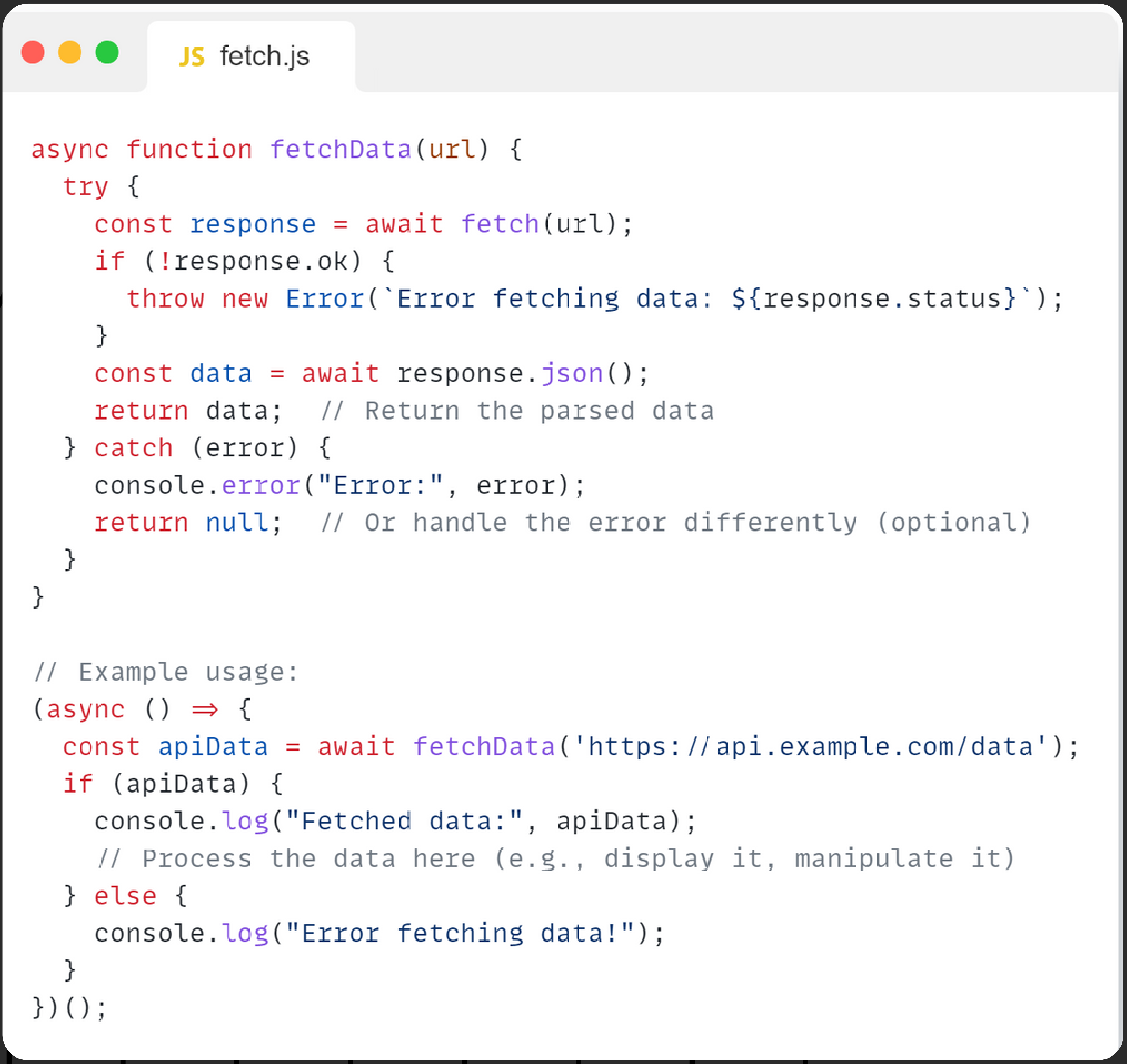
```
let jsonString = '{ "name": "Alice", "age": 25 }';
let jsonObject = JSON.parse(jsonString);
console.log(jsonObject.name); // Output: "Alice"
console.log(jsonObject.age); // Output: 25
```

To convert a JavaScript object to JSON format, use JSON.stringify().



```
const jsonData = { "name": "John", "age": 22 };
const jsonStr = JSON.stringify(jsonData);
console.log(jsonStr); // Output: "{\"name\":\"John\",\"age\":22}"
```

JS code to handle JSON data from an API



```javascript
async function fetchData(url) {
  try {
    const response = await fetch(url);
    if (!response.ok) {
      throw new Error(`Error fetching data: ${response.status}`);
    }
    const data = await response.json();
    return data;  // Return the parsed data
  } catch (error) {
    console.error("Error:", error);
    return null;  // Or handle the error differently (optional)
  }
}

// Example usage:
(async () ⇒ {
  const apiData = await fetchData('https://api.example.com/data');
  if (apiData) {
    console.log("Fetched data:", apiData);
    // Process the data here (e.g., display it, manipulate it)
  } else {
    console.log("Error fetching data!");
  }
})();
```

Did you find it Useful?

Leave a comment!

Alamin CodePapa

@CodePapa360

FOLLOW FOR MORE

Like Comment Repost