JAVASCRIPT · DATA STRUCTURES

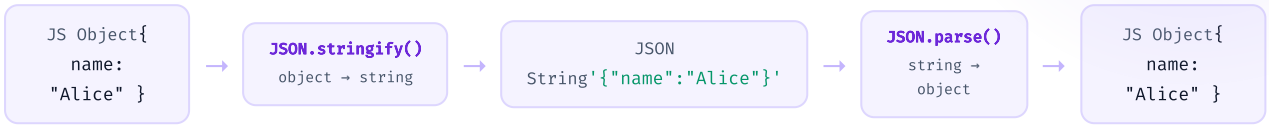
Working with JSON in JavaScript

JS · JSON

JSON.parse() JSON.stringify()

Covers how JS communicates with the outside world — serializing objects and parsing JSON data.

THE BIG PICTURE



WHY JSON EXISTS

JS Objects vs JSON

- JS objects are flexible — **keys without quotes**, can hold functions, Symbols, undefined
- Networks need a **universal format** any language can read — that's JSON
- JSON is strict — **keys must be strings**, no functions or Symbols allowed

```
// JS Object — flexible
const user = { name: "Alice", greet() {} }

// JSON — strict & universal
'{"name": "Alice"}'  // no function
```

QUICK REFERENCE

Two Methods, Two Directions

METHOD	DIRECTION	WHEN TO USE
JSON.parse()	String → Object	Reading API response or storage
JSON.stringify()	Object → String	Sending to API or saving to storage

```
// Real world flow
const raw  = JSON.parse(apiResponse)
const body = JSON.stringify(myObject)
```

JSON METHOD

JSON.parse() — String → Object

Use when you **receive** data from an API or load from storage.

```
const str = '{"name":"Alice","age":25}'
const obj = JSON.parse(str)

obj.name     // "Alice"
obj.age      // 25
typeof obj   // "object"
```

JSON METHOD

JSON.stringify() — Object → String

Use when you **send** data to an API or save to storage.

```
const obj = { name: "Bob", age: 28 }
const str = JSON.stringify(obj)

str         // '{"name":"Bob","age":28}'
typeof str // "string"
```

WATCH OUT

What stringify() Silently Drops

```
const obj = {
  name:  "Alice",            // ✅ kept
  greet: () => "hi",         // ❌ function
  age:   undefined,          // ❌ undefined
  id:    Symbol("id"),       // ❌ symbol
}
JSON.stringify(obj)
// '{"name":"Alice"}'
```

- Dropped **silently** — no error is thrown
- Functions are not a valid JSON value type
- `undefined` has no JSON equivalent
- Symbols are always excluded from JSON
- Always verify output when objects have complex values