Perfect. Continuing in **the SAME PDF-style format, depth, and simplicity**.

📘 JavaScript JSON — Internal & Practical View

(Data Exchange ● `stringify` / `parse` ● Memory ● Traps)

🔹 1. What is JSON?

Definition

JSON (JavaScript Object Notation) is a **text-based data format** used to **store and exchange data**.

👉 JSON is **not JavaScript code**

👉 JSON is **just a string format**

Simple Meaning

- JavaScript Object → **in-memory**
- JSON → **string representation**

🔎 **Internal Behavior**

- JSON lives as **plain text**
- Used to send data:
  - Client ↔ Server
  - File storage
  - APIs
- JS must **convert** JSON ↔ Object to use it

✅ **Key Takeaway**

JSON is data format, not a JS object.

⚠️ **Interview Trap**

JSON ≠ JavaScript object.

🔹 2. JSON vs JavaScript Object (VERY IMPORTANT)

Feature	JS Object	JSON
Type	Object	String
Functions	Allowed	❌ Not allowed
Quotes	Optional	Mandatory (`""`)
Trailing commas	Allowed	❌ Not allowed

Feature	JS Object	JSON
Comments	Allowed	❌ Not allowed

```js
// JavaScript Object
const obj = { name: "Anoop" };
```

```json
{
  "name": "Anoop"
}
```

🔎 Internal Behavior

- JSON parser is **strict**
- Any invalid syntax → parse error

✅ Key Takeaway

JSON follows stricter rules than JS.

⚠️ Interview Trap

Single quotes are invalid in JSON.

🔹 3. `JSON.stringify()` — Object → JSON

Syntax

```js
JSON.stringify(value);
```

Example

```js
const user = {
  name: "Anoop",
  age: 22
};

const json = JSON.stringify(user);
```

Output

```
js

json // '{"name":"Anoop","age":22}'
```

🔎 Internal Behavior

- Traverses object
- Converts values to text
- Removes:
  - functions
  - `undefined`
  - symbols

```
js

JSON.stringify({
  a: 1,
  b: undefined,
  c: function () {}
});
// '{"a":1}'
```

✅ Key Takeaway

Only **serializable data** survives stringify.

⚠️ Interview Trap

Functions silently disappear.

🔹 4. `JSON.parse()` — JSON → Object

Syntax

```
js

JSON.parse(jsonString);
```

Example

```
js

const str = '{"name":"Anoop","age":22}';
const obj = JSON.parse(str);
```

🔎 Internal Behavior

- Parses string character by character
- Creates new object in **heap**
- Throws error if JSON is invalid

```
js

JSON.parse("{name: 'Anoop'}"); // ❌ SyntaxError
```

✅ Key Takeaway

`JSON.parse()` always creates a **new object**.

⚠️ Interview Trap

Invalid JSON crashes parsing.

🔹 5. Deep Copy using JSON (Common Trick)

```
js

const obj1 = { a: 1, b: { c: 2 } };

const copy = JSON.parse(JSON.stringify(obj1));
```

🔎 Internal Behavior

- Object → JSON string
- JSON string → new object
- References are broken

⚠️ LIMITATIONS (VERY IMPORTANT)

Lost during copy:

- Functions
- `undefined`
- `Date`
- `Map`, `Set`
- Circular references ❌

```
js

const obj = {};
obj.self = obj;
```

```
JSON.stringify(obj); // ❌ TypeError
```

✅ Key Takeaway

JSON deep copy is **limited and unsafe**.

⚠️ Interview Trap

Circular objects crash stringify.

🔹 6. JSON in APIs (Real World)

Sending Data

```
js

fetch("/api", {
  method: "POST",
  headers: {
    "Content-Type": "application/json"
  },
  body: JSON.stringify({ name: "Anoop" })
});
```

Receiving Data

```
js

const data = await response.json();
```

🔎 Internal Behavior

- Network sends **text**
- Browser parses JSON into object
- Data becomes usable in JS

✅ Key Takeaway

APIs always exchange **strings**, not objects.

🔹 7. Common JSON Errors

❌ Trailing comma

```
json

{
  "a": 1,
```

```
}
```

❌ Comments

```json
{
  // comment
  "a": 1
}
```

🔎 Internal Behavior

- JSON parser fails immediately
- No recovery

⚠️ Interview Trap

JSON is stricter than JS object literals.

🔹 8. `reviver` and `replacer` (Advanced but Useful)

Replacer (during stringify)

```js
JSON.stringify(obj, ["name"]);
```

Reviver (during parse)

```js
JSON.parse(str, (key, value) => {
  if (key === "age") return value + 1;
  return value;
});
```

🔎 Internal Behavior

- Hook into serialization / parsing
- Modify data during conversion

✅ Key Takeaway

Reviver & replacer allow controlled transformation.

🧠 FINAL SUMMARY (EXAM + INTERVIEW)

- JSON is a string format
- Used for data exchange

- `stringify` → object to string
- `parse` → string to object
- JSON is strict
- Deep copy via JSON has limitations
- APIs always send JSON as text

⏭️ NEXT (Same Format)

Reply with **one word**:

- `map-set`
- `spread-rest`
- `array-vs-object`
- `object-methods`
- `storage`

I'll continue in **exact same simplified PDF-style**.