40. 实现一个 JSON.stringify

1. 核心概念 (Core Concept)

`JSON.stringify()` 是 JavaScript 内置的用来将 JavaScript 值（对象、数组、基本类型等）转换成 JSON 格式字符串的方法。JSON (JavaScript Object Notation) 是一种轻量级的数据交换格式，易于人阅读和编写，同时也易于机器解析和生成。

2. 为什么需要它？ (The "Why")

- **数据序列化：** 将复杂的 JavaScript 数据结构转换成字符串格式，便于在网络上传输（如发送 HTTP 请求体）或存储（如浏览器 localStorage）。
- **跨语言数据交换：** JSON 是一种独立于语言的数据格式，使得不同编程语言之间能够方便地交换数据。
- **简便易用：** `JSON.stringify()` 提供了标准、高效的方式来完成这一转换过程，无需手动拼接字符串。

3. API 与用法 (API & Usage)

`JSON.stringify()` 方法的语法如下：

```
JSON.stringify(value[, replacer[, space]])
```

- `value`: 将要转换成 JSON 字符串的 JavaScript 值。
- `replacer` (可选): 一个函数或者一个数组。
  - 如果是一个函数，它会接收两个参数 (key, value)，用于转换或过滤属性值。
  - 如果是一个数组，它指定了要包含在 JSON 字符串中的属性名（或索引）的列表。
- `space` (可选): 一个字符串或者一个数字，用于控制输出字符串的缩进和格式化。
  - 如果是一个数字，表示每一级的缩进空格数（最大 10）。
  - 如果是一个字符串（如 '\t' 或 ' '），用作缩进字符。

核心转换规则:

- 基本类型：`string`，`number`，`boolean` 被转换为对应的 JSON 基本类型。`null` 转换为 `null`。
- `undefined`, Functions, Symbols：这些值在对象中会被忽略，在数组中会被转换成 `null`。
- `BigInt`： 默认会抛出 `TypeError`。
- 对象 (`Object`):
  - 只包含可枚举的自身属性。
  - 属性名必须是字符串。

    - 属性值如果包含循环引用，会抛出 `TypeError`。
    - 如果对象有 `toJSON()` 方法，则会调用该方法，并使用其返回值进行序列化。
- 数组 (`Array`):
    - 元素会被转换为对应的 JSON 值。
    - `undefined`, Functions, Symbols 元素会被转换为 `null`。

代码示例 (来自 MDN Web Docs):

基本用法:

```
const obj = {
  a: 1,
  b: [2, 3],
  c: 'hello',
  d: true,
  e: null,
  f: undefined, // 会被忽略
  g: function() {} // 会被忽略
};

console.log(JSON.stringify(obj));
// Output: {"a":1,"b":[2,3],"c":"hello","d":true,"e":null}
```

使用 replacer 数组:

```
const obj = {
  a: 1,
  b: 2,
  c: 3
};

console.log(JSON.stringify(obj, ['a', 'c']));
// Output: {"a":1,"c":3}
```

使用 replacer 函数:

```
const obj = {
  a: 1,
  b: 2,
  c: 'sensitive data'
};

console.log(JSON.stringify(obj, (key, value) => {
  if (key === 'c') {
    return 'redacted';
  }
```

```
  return value;
}));
// Output: {"a":1,"b":2,"c":"redacted"}
```

使用 space 进行格式化:

```
const obj = {
  a: 1,
  b: 2,
  c: {
    d: 3
  }
};

console.log(JSON.stringify(obj, null, 2));
/* Output:
{
  "a": 1,
  "b": 2,
  "c": {
    "d": 3
  }
}
*/
```

4. 关键注意事项 (Key Considerations)

- **循环引用 (Circular References):** `JSON.stringify()` 无法处理包含循环引用的对象，会抛出 `TypeError`。
- **不可序列化的值:** `undefined`,函数, Symbols, `BigInt` （默认情况下）等值会被特殊处理或忽略，需要注意。
- **`toJSON()` 方法:** 对象原型链上的 `toJSON()` 方法会影响序列化结果，它是重要的自定义序列化机制。
- **数据丢失:** 并非所有 JavaScript 数据结构都能完美序列化为 JSON，例如函数、Symbol、Date 对象会转换为字符串等，丢失了原始类型信息。

5. 参考资料 (References)

- **MDN Web Docs - JSON.stringify()**: https://developer.mozilla.org/zh-CN/docs/Web/JavaScript/Reference/Global_Objects/JSON/stringify
- **ECMA-404 The JSON Data Interchange Format**: https://www.ecma-international.org/publications-and-standards/standards/ecma-404/

- **ECMAScript 2023 Language Specification - 25.5.2 The JSON.stringify (value, replacer, space) Arguments:** https://tc39.es/ecma262/#sec-json.stringify (详细的算法步骤和规则)