typeof operator and type checking

🔍 `typeof` Operator and Type Checking in JavaScript

The `typeof` operator in JavaScript is used to **determine the data type** of a variable or value at runtime.

✅ Syntax

```
typeof operand
```

OR

```
typeof(operand)
```

Both are valid. It returns a **string** indicating the type.

🧪 Example Outputs

```
typeof "Abhi";       // "string"
typeof 42;           // "number"
typeof true;         // "boolean"
typeof undefined;    // "undefined"
typeof null;         // "object" ❗ (this is a known JavaScript quirk)
typeof Symbol();     // "symbol"
typeof 10n;          // "bigint"

typeof {};           // "object"
typeof [];           // "object" (Array is an object)
typeof function(){}; // "function"
```

🔍 Detailed Type Output Table

Value	`typeof` Result
`"hello"`	`"string"`
`100`	`"number"`
`true` / `false`	`"boolean"`
`undefined`	`"undefined"`
`null`	`"object"` ❗
`Symbol("id")`	`"symbol"`
`10n`	`"bigint"`
`{}`	`"object"`
`[]`	`"object"`
`function() {}`	`"function"` ✅

❗ Quirks of `typeof`

1. **`typeof null` returns `"object"`**
   - This is a well-known bug in JavaScript that exists for backward compatibility.
   - To properly check for `null`, use:

     ```
     value === null;
     ```

2. **Arrays return `"object"`**
   - To specifically check for arrays:

     ```
     Array.isArray([]);
     ```

3. **`typeof NaN` is `"number"`**
   - `NaN` stands for “Not-a-Number”, but it's still considered a number by `typeof`.

- Use `Number.isNaN(value)` for accurate checking.

🧠 Best Practices for Type Checking

✅ Strings

```
typeof value === "string";
```

✅ Numbers

```
typeof value === "number" && !Number.isNaN(value);
```

✅ Arrays

```
Array.isArray(value);
```

✅ Objects (but not arrays or null)

```
typeof value === "object" && value !== null && !Array.isArray(value);
```

✅ Functions

```
typeof value === "function";
```

🧪 Bonus: Using `instanceof`

While `typeof` works for **primitive types**, use `instanceof` for **complex types**:

```
[] instanceof Array;          // true
{} instanceof Object;         // true
new Date() instanceof Date;   // true
```

🧠 Summary

Use Case	Best Method
Primitive type check	`typeof`
Array check	`Array.isArray()`
Null check	`value === null`
Object (excluding null/arr)	`typeof val === 'object'` + checks
NaN check	`Number.isNaN(value)`

Let me know if you'd like a quick quiz or coding challenge to reinforce this!