perplexity

# non-primitive data types in js

In JavaScript, **non-primitive data types** are also known as **reference types**. These data types do not contain the actual value directly; instead, they store a reference (address or pointer) to the location in memory where the value is kept. This means that when you assign or copy a non-primitive variable, you are copying the reference, not the value itself.

## Main Non-Primitive (Reference) Data Types in JavaScript

### 1. Object

The core reference type in JavaScript. Objects are used to store collections of key–value pairs.

```
let person = { name: "Alice", age: 25 };
```

- Keys (properties) can be strings or symbols.
- Values can be any data type (including other objects and functions).

### 2. Array

A special kind of object for ordered collections of values (indexed numerically).

```
let arr = [1, 2, 3, "hello"];
```

- Arrays are objects under the hood: `typeof arr // "object"`

### 3. Function

Functions in JavaScript are first-class objects, so they are reference types.

```
function greet() {
  console.log("Hello!");
}
```

- Functions can be passed as arguments, returned from other functions, and have properties of their own.

### 4. Date

A built-in object for dealing with dates and times.

```
let today = new Date();
```

### 5. RegExp (Regular Expressions)

Used for pattern matching in strings.

```
let regex = /abc/i;
```

### 6. Other Built-In Objects

- **Map** and **Set** (ES6): Structures for storing unique items and key–value pairs with special properties.
- **WeakMap** and **WeakSet**: Similar to Map/Set but hold "weak" references, which do not prevent garbage collection.

## Key Features of Non-Primitive Types

- **Stored by reference** (not by value)
- **Mutable**: Their contents (properties, elements) can be changed
- **Assignment/copying** copies the reference, not the object itself

## Difference from Primitives

| Feature | Primitive Types | Non-Primitive (Reference) Types |
|---|---|---|
| Examples | Number, String, etc. | Object, Array, Function, etc. |
| Stored by | Value | Reference (pointer) |
| Mutability | Immutable | Mutable |
| `typeof` | "number", etc. | "object" (functions: "function") |
| Copies via | Value copy | Reference copy |

## Summary

- The only true **non-primitive type in JS is** `object`, but Arrays, Functions, Dates, Regexps, Maps, and Sets are all specialized objects.
- **Anything not a primitive (number, string, boolean, null, undefined, symbol, bigint) is an object/reference type.**

> Modern JavaScript programming often involves working with these objects—creating, modifying, and sharing references between parts of your program. Understanding the difference between primitive and non-primitive types is crucial for writing robust, bug-free code.