Stack vs Heap memory

> in javascript engine two types of memory i studied one is heap memory andwhere objects stored and one is i forgot where variables stored

You're absolutely right, Vipul 👋

In a **JavaScript engine**, memory is mainly divided into **two parts**:

🧠 1. Call Stack (or Stack Memory)

- This is the one you forgot 🙂
- It's where **primitive values** (like numbers, strings, booleans, etc.) and **function execution contexts** are stored.

📌 Details:

- When a function is called, a new **execution context** is created and pushed onto the **call stack**.
- All **local variables** (like `let`, `const`, or `var`) and **primitive values** used inside that function are stored here.
- Once the function finishes execution, its context is **popped** from the stack, and the memory is released.

🧩 Example:

```
js

function greet() {
  let name = "Vipul"; // stored in stack
  console.log(name);
}

greet();
```

Here:

- The variable `name` (a string primitive) is stored in the **call stack**.
- The function `greet()` execution context also lives in the stack while it runs.

🗃️ 2. Heap Memory

- This is where **objects**, **arrays**, and **functions** are stored.
- It's an **unstructured** region of memory — data is stored here and referenced by **pointers**.

🧩 Example:

```js
let user = { name: "Vipul", age: 31 };
```

Here:

- The variable **`user`** (the reference) is stored in the **stack**.
- The actual object **`{ name: "Vipul", age: 31 }`** is stored in the **heap**.
- The stack variable **`user`** just holds a **pointer** (reference) to that heap location.

🧩 Visualization:

```pgsql
Stack Memory (small & fast)       Heap Memory (large & slow)
-----------------------------------------------------------
| user  | ──────────┐              | { name: "Vipul", age: 31 } |
| a = 10 |          │              |                            |
| b = 20 |          │              |                            |
| func() |          │              |                            |
          └──────────┘ (pointer)
```

✅ Summary Table

Type of Memory	Stores	Examples
Stack	Primitive values, execution contexts, function calls	**`let x = 10;`**
Heap	Objects, arrays, functions	**`let obj = {name: "Vipul"};`**

Would you like me to explain **how garbage collection works** for both (stack and heap) in simple terms next?