let's make this "shared box vs separate boxes" thing click with a tiny console experiment you can run right now.

🧪 Experiment 1 — `var` (shared box)

Js Copy

```
for (var i = 1; i <= 3; i++) {
  setTimeout(() => console.log("var i:", i), 1000);
}
```

What you'll see after 1 second:

Code Copy

```
var i: 4
var i: 4
var i: 4
```

Why?

- `var` is **function-scoped**, so there's only **one** `i` for the whole loop.
- The loop finishes before the first `setTimeout` runs, so `i` ends up as `4` (loop exit value).
- All callbacks look into the same box and see `4`.

🧪 Experiment 2 — `let` (separate boxes)

Js Copy

```
for (let i = 1; i <= 3; i++) {
  setTimeout(() => console.log("let i:", i), 1000);
}
```

What you'll see after 1 second:

Code Copy

```
let i: 1
let i: 2
let i: 3
```

Why?

- `let` is **block-scoped**, so each loop iteration creates a **new** `i`.
- Each callback closes over its own box, keeping the value from that iteration.

🔍 How this connects to your original code

In your recursive example:

- `let item` → each `setTimeout` callback gets its own `item` box, so it logs the correct word.
- `var item` → all callbacks share one `item` box, so they all log the last value assigned before the loop paused.