💥 Understanding Scope with `var`, `let`, and `const`

```
{
  var a = 10;
  let b = 20;
  const c = 30;
}

console.log(a); // ✅ 10
console.log(b); // ❌ Uncaught ReferenceError: b is not defined
console.log(c); // ❌ Uncaught ReferenceError: c is not defined
```

🔥 What's Happening?

Keyword	Behavior Inside Block `{}`	Behavior Outside Block `{}`
`var a`	Declared globally (function/global scoped)	Accessible ✅ (prints 10)
`let b`	Block scoped 🚧	Not accessible ❌ (ReferenceError)
`const c`	Block scoped 🚧	Not accessible ❌ (ReferenceError)

📚 Key Points:

- `var` is **function scoped** or **globally scoped**. It does NOT care about blocks `{}` — that's why `a` is accessible outside the block.
- `let` and `const` are **block scoped**. They exist only within the `{}` block where they are defined. Trying to access them outside results in a **ReferenceError**.

🧠 Quick Analogy:

- Think of `var` like a **free bird** 🕊️ — it flies over block boundaries.
- Think of `let` and `const` like **goldfish in a tank** 🐟 — they **stay confined** inside the block they were created in!
