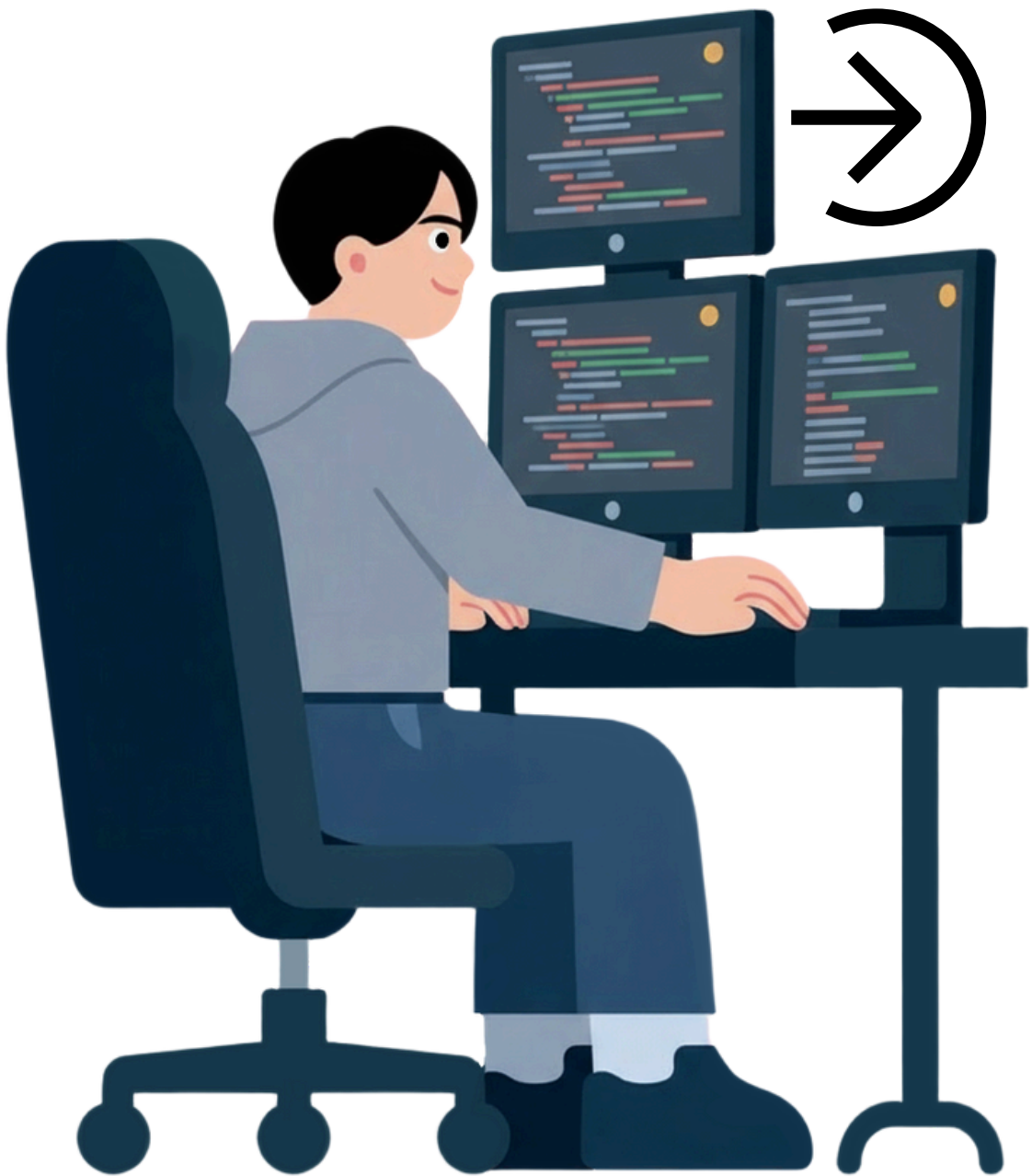
Class Vs Struct

by Shubam Gupta



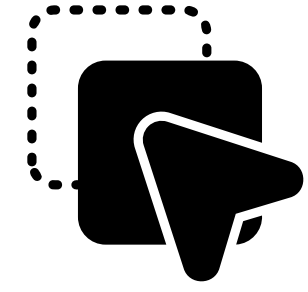
1. What is a Struct?

A Struct is a **value type** in Swift.
When you pass it around, **a copy is created**.

```
struct User {
    var name: String
}
```

🔑 Key Characteristics

- **Value Type**
- **Copied** on assignment
- Stored mostly on **Stack**
- Safer for **concurrency**
- **Fast & lightweight**



2. What is a Class?

A Class is a **reference type**.

When you pass it, **reference is shared**.

```
class User {
    var name: String
    init(name: String) {
        self.name = name
    }
}
```

🔑 Key Characteristics

- **Reference Type**
- Shared **mutable state**
- Stored on **Heap**
- Needs ARC (**Automatic Reference Counting**)
- **Supports inheritance**

3. Memory: Stack vs Heap (Very Important 🚨)

Feature	Struct	Class
Type	Value	Reference
Memory	Stack (mostly)	Heap
Copy	Yes	No
ARC	❌ Not needed	✅ Required
Thread Safety	Safer	Risky

📌 Why this matters?

- Stack → **faster, safer**
- Heap → **flexible but needs memory management**

4. STACK MEMORY - Why is it faster & safer?

How stack works

- Stack follows **LIFO** (Last In, First Out)
- **Memory allocation** & **deallocation** is **automatic**.
- Happens at **compile time.**
- **Allocated** when function starts.
- **Removed instantly** when function ends

Why stack is FAST

✔ No searching for free memory

✔ Just move a pointer up/down

✔ Constant-time operation (O(1))

👉 CPU loves predictable memory access.

5. HEAP MEMORY - Why flexible but needs management?

How heap works

- Shared memory region
- Allocated at runtime
- Objects live beyond function scope.
- Stored on heap
- Reference stored on stack
- Object remains until ARC deallocates it

Why heap is FLEXIBLE

✓ Size can grow dynamically

✓ Objects can live longer

✓ Can be shared across functions, threads

6. Initializers in Struct vs Class

Struct Initialisers

- Swift provides **automatic memberwise initializer**

```
struct Car {
    var model: String
    var year: Int
}

let car = Car(model: "Swift", year: 2024)
```

- Custom initializer is optional

Class Initialisers

- **No default initialiser** if properties are not initialised.
- You must **define init**.

```
class Car {
    var model: String

    init(model: String) {
        self.model = model
    }
}
```

Supports:

- Designated Initialisers.
- Convenience Initialisers.
- Required Initialisers.

"Structs are safer and faster due to value semantics, while classes offer flexibility through reference semantics but require careful memory management."

Thank you
by Shubam Gupta