- No raw pointers (or C++ like references)

- No way to mutate values within functions

- `inout` parameters allow pointer-like semantics safely

- Use cases:

- Multiple return types without tuples

- Swap

- Modifying properties

- Reassigning references

83

inout params

```swift
struct Point { var x: Double, y: Double }

func movePoint (_ point: inout Point, dx: Double, dy: Double)
{
    point.x += dx  // Direct memory modification
    point.y += dy  // No copy needed
}

var p = Point (x: 1, y: 2)
movePoint (&p, dx: 5, dy: 3)  // Efficient in-place mutation
```

`inout` params

- No raw pointers (or C++ like references)
- No way to mutate values within functions
- `inout` parameters allow pointer-like semantics safely
- Use cases:
  - Multiple return types without tuples
  - Swap
  - Modifying properties
  - Reassigning references

```
struct Point { var x: Double, y: Double }

func movePoint (_ point: inout Point, dx: Double, dy: Double)
{
    point.x += dx  // Direct memory modification
    point.y += dy  // No copy needed
}

var p = Point (x: 1, y: 2)
movePoint (&p, dx: 5, dy: 3)  // Efficient in-place mutation
```

Cheat `inout`