# Decorator pattern

**Decorator pattern** được sử dụng để mở rộng chức năng của một object mà không phải thay đổi class hoặc constructor function hiện có.

Pattern này có thể được sử dụng để thêm tính năng vào một object mà không thay đổi code sâu bên dưới chúng.

Một ví dụ đơn giản của pattern này là:

```
function Car(name) {
  this.name = name
  // Default values
  this.color = 'White'
}
// Tạo object mới
const tesla = new Car('Tesla Model 3')
// Thêm tính năng mới vào object mà không phải thay đổi constructor function
tesla.setColor = function (color) {
  this.color = color
}
tesla.setPrice = function (price) {
  this.price = price
}
tesla.setColor('black')
tesla.setPrice(49000)
// prints black
console.log(tesla.color)
```

Một ví dụ thực tế hơn của pattern này là:

Bình thường thì chi phí chi trả cho một chiếc xe sẽ khác nhau dựa vào tính năng mà nó có. Nếu không có decorator pattern, chúng ta sẽ tạo nhiều class khác nhau đại diện cho những loại xe mà tính năng của chúng khác nhau, mỗi cái lại có một phương thức tính giá tiền. Ví dụ:

```
class Car() {
}
class CarWithAC() {
}
class CarWithAutoTransmission {
}
class CarWithPowerLocks {
}
class CarWithACandPowerLocks {
}
```

Nhưng với decorator pattern, chúng ta có thể tạo một class cơ bản là `Car` và thêm phương thức tính toán dựa vào decorator function. Ví dụ:

```
class Car {
  constructor() {
    // Default Cost
    this.cost = function () {
      return 20000
    }
  }
}
// Decorator function
function carWithAC(car) {
  car.hasAC = true
  const prevCost = car.cost()
  car.cost = function () {
    return prevCost + 500
  }
}
// Decorator function
function carWithAutoTransmission(car) {
  car.hasAutoTransmission = true
  const prevCost = car.cost()
  car.cost = function () {
    return prevCost + 2000
  }
}
// Decorator function
function carWithPowerLocks(car) {
  car.hasPowerLocks = true
  const prevCost = car.cost()
  car.cost = function () {
    return prevCost + 500
  }
}
```

Đầu tiên, chúng ta tạo class cơ bản là `Car` cho việc tạo `Car` object. Sau đó, chúng ta tạo các decorator function cho các tính năng mà chúng ta muốn thêm vào `Car` object. Tiếp theo chúng ta override cost function để return về giá tiền dựa vào từng đặc điểm của xe.

Quy trình sẽ thực hiện như thế này:

```
const car = new Car()
console.log(car.cost())
carWithAC(car)
carWithAutoTransmission(car)
carWithPowerLocks(car)
```

Cuối cùng, chúng ta có thể tính toán giá tiền của chiếc xe như thế này:

```
// Tính toán chi phí của chiếc xe sau khi thêm n tính năng
console.log(car.cost())
```

Nếu bạn nào đã từng code **React** thì sẽ biết High Order Component, nó chính là Decorator Pattern!