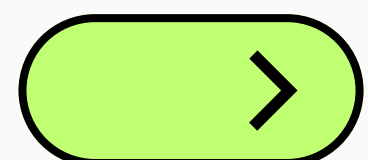


FLORIAN BÊME

# STRATEGY PATTERN

CHANGE TA STRATÉGIE SANS CHANGER TON  
CODE!



01

# C'est quoi ?

Un design pattern qui te permet de changer de comportement sans réécrire ton code. Tu crées plusieurs "recettes" différentes et tu choisis celle que tu veux appliquer.



02

# Le problème

Ton code accumule des if/else pour gérer différentes méthodes...

- ✗ Fonction géante avec des dizaines de if/else
- ✗ Code illisible et difficile à comprendre
- ✗ Ajouter une option = modifier tout le code



03

# La solution : Strategy

Une stratégie = une classe !

- ✓ Chaque algorithme isolé dans sa classe
- ✓ Changement de stratégie à la volée
- ✓ Ajout facile sans modifier l'existant



04

# Implémentation

```
// SANS STRATEGY ✖  
function pay(amount, method) {  
  if (method === 'card') {  
    console.log(`Carte: ${amount}€`);  
  } else if (method === 'paypal') {  
    console.log(`PayPal: ${amount}€`);  
  } else if (method === 'crypto') {  
    console.log(`Crypto: ${amount}€`);  
  }  
  // Ajouter Apple Pay = modifier cette fonction ⚠  
}
```

```
// AVEC STRATEGY ✔  
class CardStrategy {  
  pay(amt) { console.log(`Carte: ${amt}€`); }  
}  
class PayPalStrategy {  
  pay(amt) { console.log(`PayPal: ${amt}€`); }  
}  
const processor = new PaymentProcessor();  
processor.setStrategy(new CardStrategy());  
processor.pay(100);
```