追求优雅的代码

用设计模式让代码更优雅

https://github.com/zongzi531

场景下的最佳解决方案

「目的」

复用代码
提升可维护性
提升性能
……

「无处不在」

观察者模式

```
const div = document.createElement('div')

div.addEventListener('click', event => {
  console.log('click 1.')
})

div.addEventListener('click', event => {
  console.log('click 2.')
})

div.click()
```

观察者模式

```
const div = document.createElement('div')

const handleClick = (event: MouseEvent) => {
  console.log('后面某些时间可能会被移除～')
}

div.addEventListener('click', handleClick)

setTimeout(() => {
  div.removeEventListener('click', handleClick)
}, 3000)
```

原型模式

```
class Human {
  public name: string
  constructor(name: string) {
    this.name = name
  }
  getName() {
    return this.name
  }
  sleep() {
    console.log('...')
  }
}

class Zong extends Human {
  constructor(name: string) {
    super(name)
  }
  sayName() {
    console.log(`Hello,  My name is ${this.name}`)
  }
  sleep() {
    console.log('呼～')
  }
}
```

原型模式

```
import { hasOwn } from '@vue/shared'

const obj = {
  hasOwnProperty(key: string) {
    return true
  }
}

console.log(obj.hasOwnProperty('zong')) // 有风险的使用方式
console.log(Object.prototype.hasOwnProperty.call(obj, 'zong')) // 正确的使用方式
console.log(hasOwn(obj, 'zong')) // 正确的使用方式
```

原型模式

```
class Father {
  say() {
    console.log('我是父')
  }
}

class Child extends Father {
  say() {
    // super.say() 加上这个会打印 ?
    console.log('我是子')
  }
}

const child1 = new Child()
child1.say() // ?
```

原型模式

```
class Father {
  say() {
    console.log('我是父')
  }
}

class Child extends Father {
  say() {
    super.say()
    console.log('我是子')
  }
}

const child1 = new Child()
const child2 = new Child()

child1.__proto__.say = () => { console.log('我是谁') }
console.log(child1.say())
console.log(child2.say())
```

Mixin 模式

```
class Zong extends Human {
  constructor(name: string) {
    super(name)
  }
  sayName() {
    console.log(`Hello,  My name is ${this.name}`)
  }
  sleep() {
    console.log('呼～')
  }
}

const mixin = {
  sayHi() {
    console.log('嗨～')
  },
  sleep() {
    console.log('呼～～')
  }
}

Object.assign(Zong.prototype, mixin)

const zong1 = new Zong('1')
```

HOC

Composition API

CRUD 的爱恨情仇

HOF

Mixin

高效开发

专注调试

高安全性

易用易维护

低代码基石

总之，
坚持你的坚持。

提问环节

谢谢！