09-02-2021

Dev Javascript

# [Phần 1] Học nhanh các Javascript Design Pattern

**Nội dung bài viết**

Design Pattern là gì
Module pattern
Revealing Module pattern
Singleton pattern
Tham khảo:

Có thể bạn đã từng nghe về **Design Pattern** nhưng bạn chưa hiểu rõ lắm về chúng. Theo mình thì sớm hay muộn thì ai cũng phải tìm hiểu về Design Pattern thôi 😀

Thế design pattern là cái gì thế?

Dễ thôi, đọc tiếp là biết nha 😀

# Design Pattern là gì

> ***Nói một cách đơn giản design pattern là khuôn mẫu thiết kế có sẵn để giải quyết một vấn đề***. *Đây là một thứ khá là vĩ mô nhằm giúp code của bạn trông thống nhất và hiệu quả. Bạn không nghe nhầm đâu,* ***vĩ mô*** *cả đấy* 😆

Thường thì bạn sẽ không phải tự chế ra các design pattern mà là đã có các tiền bối sáng tạo ra nó, họ cảm thấy nếu viết code như thế sẽ mang lại sự hiệu quả và dễ đọc. Bây giờ chúng ta cứ thế mà phang thôi.

Ví dụ :

Bạn code một app **todo list** đơn giản cỡ 200 dòng code, bạn chỉ cần dùng các function gọi đi gọi lại là đủ rồi. Nhưng khi có 2-3 người vào code chung với bạn, mỗi người mỗi phong cách code dẫn đến mất đi sự nhất quán của code ban đầu. Mặc dầu code chạy được nhưng sau này đọc lại bạn sẽ thấy code rất khó bảo trì. Chưa kể nếu bạn đầu thiết kế code không khoa học thì sẽ rất khó mở rộng hay thêm chức năng. Đó là lý do bạn cần thống nhất design pattern ngay từ đầu.

**Design Pattern** có ở mọi ngôn ngữ và mỗi ngôn ngữ lại có nhiều loại design pattern khác nhau. Với lập trình viên Front-End thì chúng ta hay dùng các Framework như React, Angular, Vue. Bạn có để ý thấy là mỗi framework có mỗi cách code khác nhau không? Đó là do chúng áp dụng nhiều design pattern khác nhau.

Ví du:

- Angular sử dụng Depedency Injection, Observer,...
- React thì dùng High Order Component, cũng được biết đến như là Decorator Pattern.

Ở trong bài viết này mình sẽ chia sẽ với anh em những design pattern phổ biến trong Javascript (chứ thật ra JS nó có nhiều lắm, chia sẽ không hết luôn 😢 ), cùng đi tìm hiểu design pattern đầu tiên thôi 🙄

# Module pattern

> ***Module pattern*** *là một design pattern mà theo mình là cực kỳ phổ biến trong Javascript, nó được dùng ở khắp mọi nơi.* ***Module*** *giúp chúng ta tách riêng code thành một vùng độc lập từ đó code chúng ta sẽ clean hơn và dễ tái sử dụng hơn rất nhiều.*

Đặc điểm của Module pattern là sử dụng **IIFE** (immediately-invoked function expression) kết hợp với **closure** và **function scope**.

> *Đọc thêm: Chinh phục High Order Function, Closures, Currying và Callback trong Javascript*

Ví dụ:

```
const map = (function () {
  // Biến này chỉ được truy cập bên trong function
  // Private variable
  const toaDo = [15.9030623, 105.8066925]
```

```
    // Function này chỉ được truy cập bên trong function
    // Private method
    function layToaDo() {
      return toaDo
    }

    return {
      // Function này có thể truy cập từ bên ngoài
      // Public method
      inToaDo: function () {
        console.log(layToaDo())
      }
    }
  })()

  map.inToaDo()
```

Mình nghĩ nhiều bạn cũng sẽ tự hỏi “Mình cần éo gì phải dùng đến IIFE như thế này, cứ khai báo tên function ra là xong có phải nhanh và đỡ rườm rà hơn không?”

Bạn nghĩ đúng rồi đó, nếu khai báo thẳng ra sẽ nhanh hơn **nhưng đó là trường hợp code của bạn**. Nếu bạn muốn đóng gói lại thành 1 file có thể dùng được nhiều nơi hay là tạo một thư viện chẳng hạn, có thể bạn sẽ gặp trường hợp là trùng tên khai báo biến => code hoạt động sai.

Ví dụ bạn tạo được một function trong file **map.js** và muốn dùng function đó ở file khác trong JS.

**map.js**

```
function map() {
  const toaDo = [15.9030623, 105.8066925]
  function layToaDo() {
    return toaDo
  }
  return {
    inToaDo: function () {
      console.log(layToaDo())
    }
  }
}
```

Trong file **app.js** của bạn cũng có một biến tên là `map`

```
const map = {
  vietnam: [15.9030623, 105.8066925]
}
```

Trong file **index.html** bạn import các file js như thế này

```
<script src="map.js"></script>
<script src="app.js"></script>
```

Và bạn đoán thử điều gì sẽ xảy ra. Bạn sẽ gặp lỗi **Uncaught SyntaxError: Identifier 'map' has already been declared,** vì biến `map` đã được khai báo rồi.

Tự dưng đi tạo thư viện mà đem lại rắc rối cho người dùng thư viện đó là mất một số namespace 😀 .

Qua ví dụ trên mình nghĩ bạn cũng thấy điểm ưu việt của module pattern. Đây chỉ là dạng module cơ bản, thế giới module của Javascript có rất rất nhiều kiểu module bên trong để bạn chọn, nếu bạn dùng **webpack** bạn sẽ thấy các tùy chọn module như sau

- Object literal notation
- The module Pattern
- AMD modules

- CommonJS module
- ECMAScript Harmony modules

> *Mình có viết series Webpack siêu tốc, bạn có thể tham khảo qua  nhé*

Sau này còn có sự xuất hiện của ES6 module nữa, nó giải quyết được khá nhiều vấn đề về namespace mà bạn gặp phải khi bạn code theo kiểu "bình thường" không dùng IIFE như trên.

## Revealing Module pattern

> ***Revealing Module Pattern** thực ra chỉ là phiên bản hoàn thiện hơn của Module Pattern truyền thống thôi.*

Nếu như Module Pattern thường thì các biến và phương thức private sẽ được khai báo trong function nhưng ngoài object return, chỉ có các biến và phương thức public mới khai báo trong object return. Nhưng với Revealing Module Pattern thì bạn sẽ khai báo cả private và public ngoài object return luôn, trong object return chỉ quy định lại tên biến và phương thức public trả về thôi.

Ví dụ:

```
const map = (function () {
  const toaDo = [15.9030623, 105.8066925]
  function layToaDo() {
    return toaDo
  }
  function inToaDo() {
    console.log(layToaDo())
  }
  return {
    inToaDo
  }
})()

map.inToaDo()
```

## Singleton pattern

> ***Singleton*** *là một object chỉ khởi tạo một lần duy nhất trong suốt chương trình chạy dù cho bạn có gọi khởi tạo nó bao nhiêu lần đi nữa. Singleton pattern là cách khai báo object đó, đơn*

*giản vậy thôi. Nhờ điều này mà nó giúp cho chương trình chúng ta không bị lãng phí bộ nhớ, chiếm ít ram hơn.*

Singleton pattern dễ thấy nhất trong javascript là object, đây là tính năng mà JS xây dựng sẵn nhằm giảm thiểu việc cấp phát bộ nhớ.

Ví dụ

```
const car = {
  name: 'BMW',
  price: 9000
}

// Khai báo một object bike và gán nó bằng car
// Điều này sẽ không phải tạo mới lại object car
// Mà motor sẽ tham chiếu đến car
const motor = car

// Thay đổi price tại motor cũng làm thay đổi price tại car
// Vì thuộc tính của motor cũng là của car
```

```
motor.price = 10000

console.log(motor.price) // 10000
console.log(car.price) // 10000
```

Bây giờ chúng ta cùng thử tạo một Singleton pattern nhé. Ở đây mình kết hợp với module pattern.

```
const User = (function () {
  let instance
  function init() {
    return {
      name: 'Dư Thanh Được',
      printName() {
        console.log(this.name)
      }
    }
  }
  return {
    getInstance: function () {
      if (!instance) {
```

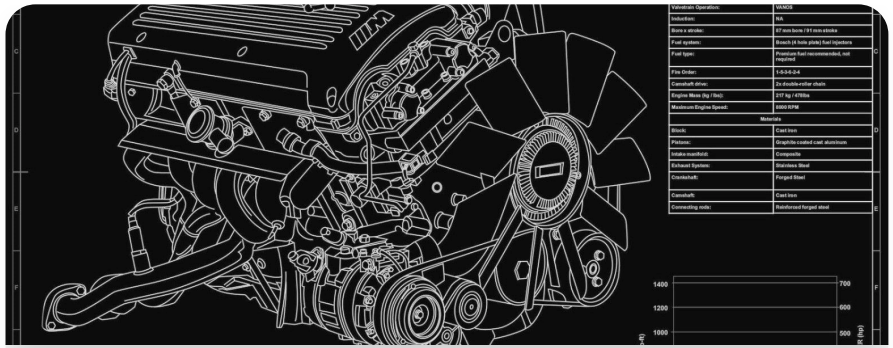
```
            instance = init()
        }
        return instance
    }
  }
})()

// Dù cho gọi getInstance() bao nhiêu lần
// thì cũng chỉ có 1 instance được tạo ra mà thôi
const user1 = User.getInstance()
const user2 = User.getInstance()
console.log(user1 === user2) // true
```

*Cảm ơn mọi người đã đọc đến đây, bài này còn phần 2 nữa nhé*

## Tham khảo:

https://blog.bitsrc.io/understanding-design-patterns-in-javascript-13345223f2dd

https://niithanoi.edu.vn/javascript-pattern.html



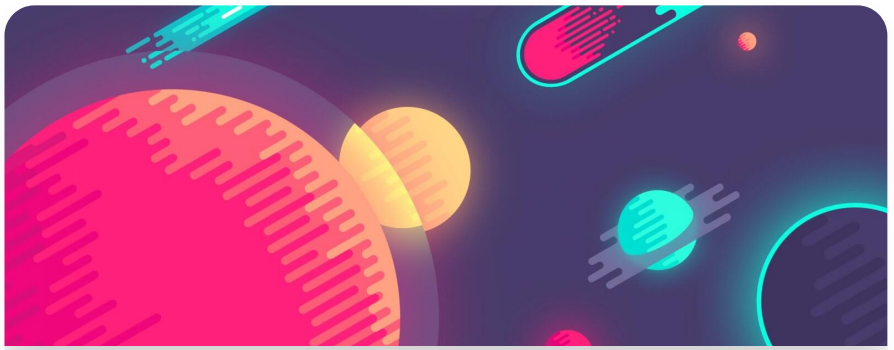
[Phần 2] Học nhanh các Javascript Design Pattern


Cheatsheet học nhanh Destructuring, Rest Parameters...

Front-End Developer ơi, hãy học cho chắc CSS rồi học...


Phần 6 (Phần cuối): JSON và Ajax trong Javascript

Sự thật về Virtual DOM, thực sự có nhanh hơn DOM...



Khóa học Javascript từ căn bản đến nâng cao - Xây...

**Share Article:**

https://xdevclass.com/phan-1-hoc-nhanh-cac-j

## Dư Thanh Được

Chào mọi người, mình là Founder Xdevclass - Lập trình tinh gọn. Thế mạnh UX/UI, Front-End. Thích đọc tin tức về tech & marketing. Thích chia sẻ về lập trình website và phát triển bản thân. Donate Momo: 0768447467

Bắt đầu bình luận nào

### 2 BÌNH LUẬN

**Đức**

Design pattern người ta thưởng phỏng vấn gì vậy a. Em chuẩn bị đi pv ạ

0 Trả lời

**Dư Thanh Được** Author

Reply to Đức

Kể một số Design Pattern trong Javascript?
Em biết gì về Design Pattern abcxyz...?
Xoay quanh vậy thôi à :v

0 Trả lời

## Về XdevClass

XdevClass là trang blog được thành lập bởi Thanh Được chuyên về các chủ đề lập trình và phát triển bản thân. Với phương châm lập trình tinh gọn, blog luôn hướng đến cách viết ngắn gọn, dễ hiểu và thực tế.

## Bài viết gần đây

11-02-2021

**Khóa học Javascript từ căn bản đến nâng cao – Xây dựng 10 project với JS thuần – Kiến thức đầy đủ để bắt đầu với mọi Javascript Framework**

02-03-2021

**Object methods và “this” trong Javascript, có bài tập thực hành**

26-02-2021

**Property flags và descriptors – Bí thuật bên trong Javascript Object**

17-02-2021

**Cuối cùng chúng ta code vì điều gì? Các ngã rẽ của nghề dev**

## Fanpage