Module pattern và Revealing Module pattern

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()
```