**Nhóm 5 – Tuần 2**

1. **Sự khác nhau giữa: Let, Var trong ES6**

* Trong ES6 có cung cấp một từ khóa nữa dùng để khởi  tạo biến đó là từ khóa let, từ khóa này khác với từ khóa var ở chỗ phạm vi hoạt động. Với từ khóa var nếu ban khai báo biến bên trong hàm thì đó là biến cục bộ, còn nếu bạn khai báo bên ngoài hàm thì nó sẽ là một biến toàn cục. Còn với từ khóa let thì phạm vi hoạt động của nó nhỏ hơn, nó chỉ tồn tại bên trong khối đang khai báo và ta gọi đây là phạm vi **block scoped**.
* Từ khóa let dùng để khởi tạo một biến nhưng biến đó chỉ có tác dụng bên trong khối đang khai báo (block-scoped).

**Cú pháp**: let var\_name = var\_value;

* Ở hình dưới đây mình khai báo biến domain và biến này chỉ hoạt động trong phạm vi của lệnh if.

if (true)

{

let domain = ‘freetuts.net’;

}

**// Lỗi, vì biến domain không tồn tại, nó chỉ tồn tại trong khối if bên trên**

console.log(domain);

* **Khi nào nên sử dụng Let để khai báo:**

Với phạm vi hoạt động hẹp như vậy thì let thường dùng để khai báo các biến mang tính chất tạm thời, nghĩa là nó chỉ sống trong một phạm vi hoạt động của khối đó thôi, không sử dụng qua vị trí khác.

**Ví dụ**: Viết chương trình hoán đổi giá trị của hai biến a và b nếu giá trị của a nhỏ hơn giá trị của b.

**Với bài toán này thì thông thường ta phải khai báo thêm một biến tạm như sau:**

var a = 12;

var b = 20;

if (a < b)

{

    var tmp = a;

    a = b;

    b = tmp;

}

console.log("a: " + a);

console.log("b: " + b);

console.log("tmp: " + tmp);

**Sử dụng [firebug](https://freetuts.net/su-dung-firebug-de-kiem-tra-loi-javascript-380.html" \o "firebug) bạn sẽ thấy kết quả như hình sau:**

a: 20

b: 12

tmp: 12

**Quay lại bài toán ta sử dụng từ khóa let như sau:**

var a = 12;

var b = 20;

if (a < b)

{

    let tmp = a;

    a = b;

    b = tmp;

}

console.log("a: " + a);

console.log("b: " + b);

console.log("tmp: " + tmp); // đoạn này lỗi

* Đoạn code này chạy bình thường, tuy nhiến nếu bạn cố ý console.log(tmp) như ví dụ trên thì sẽ xuất hiện lỗi biến tmp không tồn tại vì biến tmp chỉ tồn tại bên trong lệnh if.

1. **Arrow Function**

**- Cú pháp căn bản:**

Cú pháp căn bản nhất của arrow function như sau:

var functionName = (var1, var2) => {

    // Nội dung function

};

**Ví dụ**: Viết arrow function in ra câu chào và so sánh với cách tạo function thông thường.

**Arrow function**:

var hello = (name, message) => {

    console.log("Chào " + name + ", bạn là " + message);

};

hello('Cường', 'admin freetuts.net');

**Normal function**:

function hello(name, message)

{

    console.log("Chào " + name + ", bạn là " + message);

}

hello('Cường', 'admin freetuts.net');

### **- Nội dung là một câu lệnh đơn:**

+ Trường hợp trong thân của hàm chỉ có một lệnh duy nhất thì bạn có thể sử dụng theo ví dụ dưới đây.

**var** hello = (name, message) => console.log("Chào " + name + ", bạn là " + message);

+ Nghĩa là bạn có thể bỏ đi cặp dấu {}, điều này tuân thủ theo nguyên tắc "*nếu bên thân cặp {} chỉ là một câu lệnh thì bạn có thể bỏ cặp {}*".

### **- Trường hợp một tham số:**

Trường hợp truyền vào chỉ một tham số thì bạn có thể bỏ cặp ().

**var** hello = message => {

    console.log(message);

};

hello('Chào mừng bạn đến với freetuts.net');

### **- Trường hợp không có tham số:**

Trường hợp không có tham số truyền vào thì bạn sử dụng cặp () rỗng, xem ví dụ sau:

**var** hello = () => {

    console.log('Chào mừng bạn đến với freetuts.net');

};

hello();

1. **Kiểu dữ liệu Object trong CS6**

### **3.1 Khởi tạo đối tượng**

****Cách 1****: Sử dụng từ khóa new Object()

|  |  |
| --- | --- |
| 1 | **var** Comment = **new** Object(); |
|  |  |

****Cách 2****: Sử dụng từ khóa {}

|  |  |
| --- | --- |
| 1 | **var** Comment = {}; |

## **3.2 Thuộc tính và phương thức đối tượng**

Mỗi đối tượng sẽ có các thuộc tính và phương thức.

### **Thuộc tính**

Thuộc tính là những đặc điểm (*có thể hiểu là biến*) cần lưu trữ trong đối tượng. Ví dụ với đối tượng Comment thì mình cần các thuộc tính sau.

* title
* content
* fullname
* email

Lúc này ta có thể khai báo bằng ba cách.

****Cách 1****: Sử dụng từ khóa new Object()

// Khởi tạo

**var** Comment = **new** Object();

// Thêm thuộc tính

Comment.title = '';

Comment.content = '';

Comment.fullname = '';

Comment.email = '';

****Cách 2****: Sử dụng từ khóa {} và thêm thuộc tính ngay lúc khai báo

// Khởi tạo

**var** Comment = {

    title : "",

    content : "",

    fullname : "",

    email : ""

};

****Cách 3****: Sử dụng từ khóa {} và thêm thuộc tính sau đó

// Khởi tạo

**var** Comment = {};

// Thêm thuộc tính

Comment.title = '';

Comment.content = '';

Comment.fullname = '';

Comment.email = '';

### **Phương thức**

Phương thức là những hành động (*có thể hiểu là hàm*) của đối tượng. Ví dụ trong đối tượng Comment thì mình cần hai phương thức là:

* addComment()
* validateComment()

Lúc này ta sẽ có ba cách khai báo tương tự như cách khai báo thuộc tính.

****Cách 1****: Sử dụng từ khóa new Object()

// Khởi tạo

**var** Comment = **new** Object();

// Thêm phương thức

Comment.addComment = **function**(){

    // do some thing

};

Comment.validateComment = **function**(){

    // do some thing

};

****Cách 2****: Sử dụng từ khóa {} và thêm phương thức ngay lúc khai báo

// Khởi tạo

**var** Comment = {

    addComment : **function**(){

        // do some thing

    },

    validateComment : **function**(){

        // do some thing

    }

};

****Cách 3****: Sử dụng từ khóa {} và thêm phương thức sau đó

// Khởi tạo

**var** Comment = {};

// Thêm phương thức

Comment.addComment = **function**(){

    // do some thing

};

Comment.validateComment = **function**(){

    // do some thing

};