Bài 1
Giới thiệu ES6

Module: Advanced Programming with JavaScript

Mục tiêu

- Trình bày được ý nghĩa của ECMAScript
- Khai báo và sử dụng được arrow function
- Sử dụng được destructuring assignment
- Sử dụng được spread syntax (...)
- Sử dụng được từ khoá Class để khai báo lớp
- Sử dụng được cú pháp lambda trong ES6

Thảo luận

ECMAScript là gì?

Ý nghĩa của ECMAScript trong công nghệ JavaScript

Giới thiệu ECMAScript

- ECMAScript là một tiêu chuẩn JavaScript nhằm đảm bảo khả năng tương tác của các trang web trên các trình duyệt web khác nhau.
- Được tiêu chuẩn hóa bởi Ecma International theo tài liệu ECMA-262.
- ECMAScript được sử dụng
  - Viết kịch bản phía máy khách trên World Wide Web
  - Viết các ứng dụng và dịch vụ máy chủ sử dụng Node.js.
- Đã có 11 phiên bản của ECMA-262 được xuất bản

Giới thiệu ES6

- ES6 là chữ viết tắt của ECMAScript 6.
- Tập hợp các kỹ thuật nâng cao của JavaScript
- Phiên bản ES6 được xuất bản vào tháng 6/2015.
- Các thư viện/framework hiện đại như React.js, Angular, VueJS thường sử dụng các tính năng mới trong ES6.

Các tính năng nổi bật của ES6

ES6 giới thiệu một số tính năng mới như, các biến phạm vi khối, vòng lặp mới để lặp qua các mảng và đối tượng và nhiều cải tiến khác để giúp lập trình JavaScript dễ dàng và thú vị hơn:

- Block - Scoped Constructs Let and Const
- Arrow Function
- Rest Parameter
- Destructuring Assignment
- Spread Syntax
- Classes

Từ khóa let & const

Block Scoped

- Block Scoped là phạm vi trong một khối - Chỉ hoạt động trong phạm vi được khai báo bởi cặp {}.
- Trong ES6, chúng ta sử dụng từ khóa let để khai báo cho biến trong cặp {}

Ví dụ :

```
// Cú pháp ES6
for (let i = 0; i < 5; i++) {
   console.log(i); // 0,1,2,3,4
}
console.log(i); // undefined
```

Let & const

- `let` là phạm vi khối `{ }` (block scope)
- `var` có phạm vi trong hàm (function scope) và hành vi hoisting trong JavaScript xảy ra

```
// Cú pháp ES5
for (var i = 0; i < 5; i++) {
 console.log(i); // 0,1,2,3,4
}
console.log(i); // 5

// Cú pháp ES6
for (let i = 0; i < 5; i++) {
 console.log(i); // 0,1,2,3,4
}
console.log(i); // undefined
```

- Từ khóa `const` mới được giới thiệu trong ES6 **định nghĩa các hằng số**
- **Hằng số** ở chế độ **chỉ đọc**

Phân biệt var, let và const

- Các biến được khai báo với từ khóa var có phạm vi trong hàm (function scope)
- Các biến được khai báo với từ khóa let là phạm vi khối { } (block scope)
- Biến const dùng để khai báo một hằng số - là một giá trị không thay đổi được trong suốt quá trình chạy.

Arrow Function

Cú pháp arrow function

- Cung cấp cú pháp ngắn gọn hơn để viết **biểu thức hàm**
- Các Arrow Function được định nghĩa bằng cú pháp mới, ký hiệu suy ra `=>`

```
// Cách 1: sử dụng Anonymous Function
var sum = function(a) {
    return a + 100;
}
console.log(sum(2, 3)); // 5

// Cách 2: Sử dụng Arrow function
var sum = (a) => {
  return a + 100;
}
```

Các cách sử dụng arrow function

- 1 tham số, nếu biểu thức tính toán là đơn giản thì không cần câu lệnh return.

```
param => expression
```

- Nhiều tham số, nếu biểu thức tính toán là đơn giản thì không cần câu lệnh return.

```
(param1, paramN) => expression
```

- Nhiều câu lệnh trong thân hàm, trong trường hợp này cần sử dụng dấu {} và câu lệnh return:

```
param => {
  let a = 1;
  return a + param;
}
```

- Nhiều tham số cần dùng dấu (). Nhiều câu lệnh cần dùng ngoặc {} và câu lệnh return:

```
(param1, paramN) => {
   let a = 1;
   return a + param1 + paramN;
}
```

Rest Parameter

Rest Parameter

- Rest Parameters là tính năng khai báo số lượng tham số không xác định khi khai báo hàm
- Sử dụng 3 dấu chấm (...) trước tên tham số đại diện.
- Tham số này hữu ích trong các tình huống cần truyền các tham số cho một hàm nhưng không biết số lượng chính xác.
- Tham số Rest chỉ có thể là tham số cuối cùng trong danh sách các tham số và chỉ có thể có một tham số Rest.

Ví dụ về rest parameter

```
function sortNumbers(...numbers) {
 return numbers.sort();
}

console.log(sortNumbers(3, 5, 7));
// Kết quả: [3, 5, 7]

console.log(sortNumbers(3, 5, 7, 1, 0));
// Kết quả: [0, 1, 3, 5, 7]
```

Destructuring Assignment

Destructuring Assignment

- Phép gán hủy cấu trúc là một biểu thức giúp dễ dàng trích xuất các giá trị từ mảng hoặc thuộc tính từ các đối tượng, thành các biến riêng biệt
- Có hai loại phép gán hủy cấu trúc:
  - Phép gán hủy cấu trúc mảng
  - Phép gán hủy cấu trúc đối tượng.

Ví dụ

Phép gán hủy cấu trúc mảng

```
let colors = ["Xanh", "Đỏ"];
let [a, b] = colors; // Phép gán hủy cấu trúc mảng
console.log(a); // Xanh
console.log(b); // Đỏ
```

Phép gán hủy cấu trúc đối tượng

```
let school = { name: "CodeGym", age: 4 };
let { name, age } = school; // Phép gán hủy cấu trúc đối tượng
alert(name); // CodeGym
alert(age); // 4
```

Spread Syntax

Spread Syntax

- Cú pháp spread đơn giản được biểu diễn bởi dấu 3 chấm: ...
- Spread Syntax cho phép duyệt qua các phần tử và truyền vào phương thức như các đối số
- Spread Syntax có thể được sử dụng khi tất cả các phần tử từ một đối tượng hoặc mảng cần được đưa vào một danh sách nào đó
- Spread Syntax cũng có thể được sử dụng để chèn các phần tử của một mảng vào một mảng khác mà không cần sử dụng các phương thức mảng như push(), unshift() hay concat(), ...

Ví dụ về Spread Syntax

Spread Syntax với Array

```
const oldArray = [1, 2, 3];
const newArray = [...oldArray, 4, 5];
console.log(newArray);     /* ouput: [1, 2, 3, 4, 5] */
```

Spread Syntax dùng tính toán

```
function sum(x, y, z) {
 return x + y + z;
}
const numbers = [1, 2, 3];
console.log(sum(...numbers));    // expected
output: 6
```

Từ khóa class

Classes

- Trong ES5 trở về trước, các class chưa bao giờ tồn tại trong JavaScript.
- Các class được giới thiệu trong ES6 trông tương tự như các class trong các ngôn ngữ hướng đối tượng khác, chẳng hạn như Java, PHP ...
- ES6 đã hỗ trợ khai báo một đối tượng theo chuẩn OOP, bằng cách sử dụng từ khóa class.
- Các class trong ES6 giúp
  - Tạo các đối tượng
  - Thực hiện kế thừa dễ dàng hơn (từ khóa extends)
- Chi tiết về Class trong ES6 chúng ta sẽ tìm hiểu thêm ở phần tiếp theo của bài học.

Demo

Sử dụng các tính năng của ES6

Thảo luận

Các phương thức xử lý mảng trong JavaScript

Một số phương thức thường dùng

- forEach
- map
- filter
- find
- findIndex
- some
- every
- sort
- reverse
- concat
- reduce

Danh sách phương thức (1)

- **forEach** thực thi một callback function mỗi khi lặp qua các array elements
- **map** tạo một mảng mới bằng cách thực thi một callback function mỗi khi lặp qua các array elements
- **filter** tạo một array mới với các elements passed điều kiện của callback function mỗi khi lặp qua các array elements
- **find** lặp qua array elements và trả về element đầu tiên passed điều kiện của callback function, nếu không tìm thấy element nào thỏa mãn trả về undefined
- **findIndex** lặp qua array elements và trả về index của element đầu tiên passed điều kiện của callback function, nếu không tìm thấy element nào thỏa mãn trả về -1

Danh sách phương thức (2)

- **some** dùng để kiểm tra trả về true nếu ít nhất một element thỏa mãn điều kiện của callback function, ngược lại trả về false nếu không có element nào thỏa mãn điều kiện
- **every** dùng để kiểm tra trả về true nếu tất cả element đều thỏa mãn điều kiện của callback function, ngược lại trả về false nếu có ít nhất một element không thỏa mãn điều kiện
- **sort** thực hiện sắp xếp các elements theo thứ tự tăng dần, ta có thể truyền vào một compare callback function để custom thứ tự sort
- **reverse** đảo ngược thứ tự của array elements

Danh sách phương thức (3)

- **concat** trả về một array mới bằng cách hợp tất cả phần tử của các mảng với nhau
- **reduce** Thực thi 1 hàm callback trên từng phần tử của mảng, theo thứ tự, chuyển giá trị trả về từ phép tính trên phần tử trước đó. Kết quả cuối cùng của tất cả các phần tử của mảng là một giá trị duy nhất

Ví dụ sử dụng reduce

```
const array1 = [1, 2, 3, 4];
const reducer = (previousValue, currentValue) => previousValue + currentValue;

// 1 + 2 + 3 + 4
console.log(array1.reduce(reducer));
// Output: 10

// 5 + 1 + 2 + 3 + 4
console.log(array1.reduce(reducer, 5));
// Output: 15
```

Demo

Thực hành cách sử dụng các array methods

Tóm tắt

Qua bài học này, chúng ta đã tìm hiểu:

- ECMAScript là gì và tổng quan về ES6
- Các tính năng tiêu biểu của ES6
- Giới thiệu về Class trong ES6
- Các phương thức thường dùng để xử lý mảng trong Javascript