SESSION 12:

ES6

Module 1: Web Application UI Design

Phiên bản: 1.0

NỘI DUNG

1. Tổng quan ES6
2. Biến, tham số mặc định, tham số còn lại
3. Cú pháp Spread, Destructuring
4. Arrow function, Lớp
5. Export và Import

TỔNG QUAN ES6

- **ES6 – ECMAScript 6**
  - Tập hợp các kỹ thuật nâng cao của JavaScript theo chuẩn ECMAScript được phê duyệt vào tháng 6/2015
  - Chuẩn mực của các JavaScript Framework

ES6 – BIẾN - 1

var

- Khai báo theo biến toàn cục (global) hoặc cục bộ (local)
- Có thể gán lại (Re-assignable) và khai báo lại (Re-declarable)
- Không thuộc vùng chết tạm thời (Temporal Dead Zone - TDZ)

let

- Khai báo trong khối lệnh (Block scope)
- Có thể gán lại nhưng không thể khai báo lại
- Phụ thuộc vào vùng chết tạm thời (TDZ)

const

- Khai báo trong khối lệnh
- Không thể gán lại, không thể khai báo lại
- Phụ thuộc vào vùng chết tạm thời

ES6 – BIẾN - 2

- **Khai báo biến với var**
  - Biến toàn cục: được sử dụng trong toàn bộ ứng dụng
  - Biến cục bộ: được sử dụng trong function được khai báo

```
var x = 5;
function variableDemo(){
    console.log('Giá trị x trong hàm: '+x);
}
console.log('Giá trị x ngoài hàm: '+x);
variableDemo();
```

```
function variableDemo(){
    var x = 5;
    console.log('Giá trị x trong hàm: '+x);
}
console.log('Giá trị x ngoài hàm: '+x);
variableDemo();
```

```
PS D:\JSDemo\ES6> node variableDemo.js
Giá trị x ngoài hàm: 5
Giá trị x trong hàm: 5
```

```
D:\JSDemo\ES6\variableDemo.js:5
console.log('Giá trị x ngoài hàm: '+x);
                                   ^

ReferenceError: x is not defined
```

ES6 – BIẾN - 3

- **Khai báo biến với let trong khối lệnh**

```
var x = 5;
if (x===5) {
    var x = 10;
    console.log('Giá trị x trong if là: '+x);
}
console.log('Giá trị x ngoài if là: '+x);
```

```
let x = 5;
if (x===5) {
    let x = 10;
    console.log('Giá trị x trong if là: '+x);
}
console.log('Giá trị x ngoài if là: '+x);
```

```
PS D:\JSDemo\ES6> node variableDemo.js
Giá trị x trong if là: 10
Giá trị x ngoài if là: 10
```

```
PS D:\JSDemo\ES6> node variableDemo.js
Giá trị x trong if là: 10
Giá trị x ngoài if là: 5
```

ES6 – BIẾN - 4

- **Khai báo biến với const – hằng số**

```
const x = 5;
console.log('Giá trị hằng số x là: '+x);
if (x===5) {
    x=10;
    console.log('Giá trị x sau khi thay đổi giá trị là: '+x);
}
```

```
D:\JSDemo\ES6\variableDemo.js:4
    x=10;
      ^

TypeError: Assignment to constant variable.
```

Tham số mặc định – còn lại - 1

- **Default Parameters: Tham số mặc định**
  - Cho phép tham số mang giá trị mặc định nếu tham số không có giá trị hoặc giá trị không xác định
  - 2 cách khai báo tham số mặc định: gán tại vị trí khai báo và gán bên trong function

```
//Tham số mặc định tại vị trí khai báo hàm
function add(num1, num2=1){
    return num1+num2;
}
console.log('Tổng 2 số là: '+add(5,1));
// Sử dụng tham số mặc định
console.log('Tổng 2 số khi sử dụng tham số mặc định: '+add(5));
```

```
PS D:\JSDemo\ES6> node variableDemo.js
Tổng 2 số là: 6
Tổng 2 số khi sử dụng tham số mặc định: 6
```

```
function showName(name) {
    //Tham số mặc định bên trong hàm
    name = name || "Tên mặc định";
    return name;
}
console.log('Tên là:  ' + showName("Nguyễn Duy Quang"));
// Sử dụng tham số mặc định
console.log('Tên là: ' + showName());
```

```
PS D:\JSDemo\ES6> node variableDemo.js
Tên là:  Nguyễn Duy Quang
Tên là: Tên mặc định
```

Tham số mặc định – còn lại - 2

- **Rest Parameters: Tham số còn lại**
  - Tham số đại diện cho những tham số không được khai báo
  - Sử dụng trong function thì khi gọi function sẽ không giới hạn đối số truyền vào
  - Ký hiệu bằng khai báo [...name]

```
function printNumber(num1, num2, ...otherNumbers){
    console.log("num1 = "+num1);
    console.log("num2 = "+num2);
    console.log("Other Numbers: "+otherNumbers);
}
printNumber("one","two","three","four","five");
```

```
PS D:\JSDemo\ES6> node restparameters.js
num1 = one
num2 = two
Other Numbers: three,four,five
```

CÚ PHÁP SPREAD VÀ DESTRUCTURING - 1

- **Spread Syntax – Cú pháp spread**
  - Cho phép lặp lại các phần tử của mảng (array) hay đối tượng (object)
  - Thể hiện dưới dạng [...]

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

```
PS D:\JSDemo\ES6> node spreadSyntax.js
[ [ 1, 2, 3 ], 4, 5 ]
```

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

```
PS D:\JSDemo\ES6> node spreadSyntax.js
[ 1, 2, 3, 4, 5 ]
```

CÚ PHÁP SPREAD VÀ DESTRUCTURING - 2

- **Destructuring – Phá vỡ cấu trúc**
  - Cho phép dễ dàng sử dụng các giá trị phần tử của Array hoặc Object
  - Hữu dụng khi làm việc với fuction có đối số

```
const arr = [1,2,3,4,5];
const [a,b] = arr;
console.log("Giá trị a: "+a);
console.log("Giá trị b: "+b);
console.log("Mảng arr: "+arr);
```

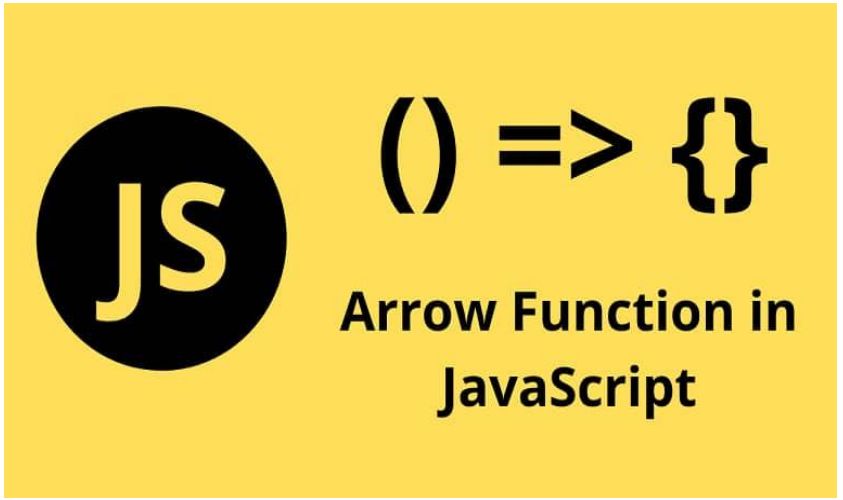
```
Giá trị a: 1
Giá trị b: 2
Mảng arr: 1,2,3,4,5
```

```
const personObject = {
    name: "Nguyễn Duy Quang",
    age: 37,
    address: "Hà Nội"
}
const {name, age} = personObject;
console.log("Name: "+name);
console.log("Age: "+age);
console.log(personObject);
```

```
PS D:\JSDemo\ES6> node destructuring.js
Name: Nguyễn Duy Quang
Age: 37
{ name: 'Nguyễn Duy Quang', age: 37, address: 'Hà Nội' }
```

ARROW FUNCTION

- **Arrow Function**
  - Viết function dưới dạng mũi tên =>
  - Cách viết linh hoạt, tùy biến function hơn



Function	Arrow function

`var functionName = function(val1, val2){`
`  /* Nội dung function */`
`}`
	`var functionName = (val1, val2) => {`
	`  /* Nội dung function */`
	`}`

CLASS – LỚP - 1

- **Class – Lớp**
  - Là một dạng function đặc biệt
  - Có phương thức khởi tạo constructor()
  - Có tính kế thừa (inheritance), kế thừa toàn bộ phương thức của lớp cha
  - Khởi tạo đối tượng từ lớp với từ khóa “new”
  - Gọi phương thức lớp cha với từ khóa “super”
  - Phương thức static
  - Getter và setter cho các thuộc tính

CLASS – LỚP - 2

```
class Person{
    constructor(){
        this.name = "Nguyễn Duy Quang";
        this.age = 37;
    }
    getName(){
        return this.name;
    }

    setAge(age){
        this.age = age;
    }
    getAge(){
        return this.age;
    }
}
var person = new Person();
person.setAge(38);
console.log("Tên người: "+person.getName());
console.log("Tuổi: "+person.getAge());
```

```
class Person{
    constructor(){
        this.name = "Nguyễn Duy Quang";
        this.age = 37;
    }
    getName(){
        return this.name;
    }
    getAge(){
        return this.age;
    }
}
class Student extends Person{
    constructor(name,age,studentId){
        super(name,age);
        this.studentId="SV001";
    }
    getStudentInfo(){
        return "Tên: "+this.getName()+"-Tuổi: "+this.getAge()+"-Mã SV: "+this.studentId;
    }
}
var student = new Student();
console.log(student.getStudentInfo());
```

EXPORT & IMPORT - 1

- Dễ làm việc và quản lý các file javascript
- Export: xuất các function của file
- Import: nhập các function của file được export
- Có 2 dạng export:
  - Theo mặc định (default): import không phụ thuộc vào tên function hay class
  - Theo tên (name): import function hay class theo ý muốn

EXPORT & IMPORT - 2

JS person.js > default

```
1  const person ={
2      name : "Nguyễn Duy Quang"
3  }
4  export default person;
```

```
import person from "./person.js";
console.log(person());
```

```
import otherName from "./person.js";
console.log(otherName());
```

JS person.js > ...

```
1  const person ={
2      name : "Nguyễn Duy Quang"
3  }
4
5  const info = {
6      event : "PR-Marketing"
7  }
8  export {person,info};
```

JS importExport.js

```
1  import {person,info} from "./person.js";
2  import * as otherName from "./person.js";
```

RIKEI Academy®
where the dream begins

KẾT THÚC

HỌC VIỆN ĐÀO TẠO LẬP TRÌNH CHẤT LƯỢNG NHẬT BẢN