TechMaster
Học là có việc

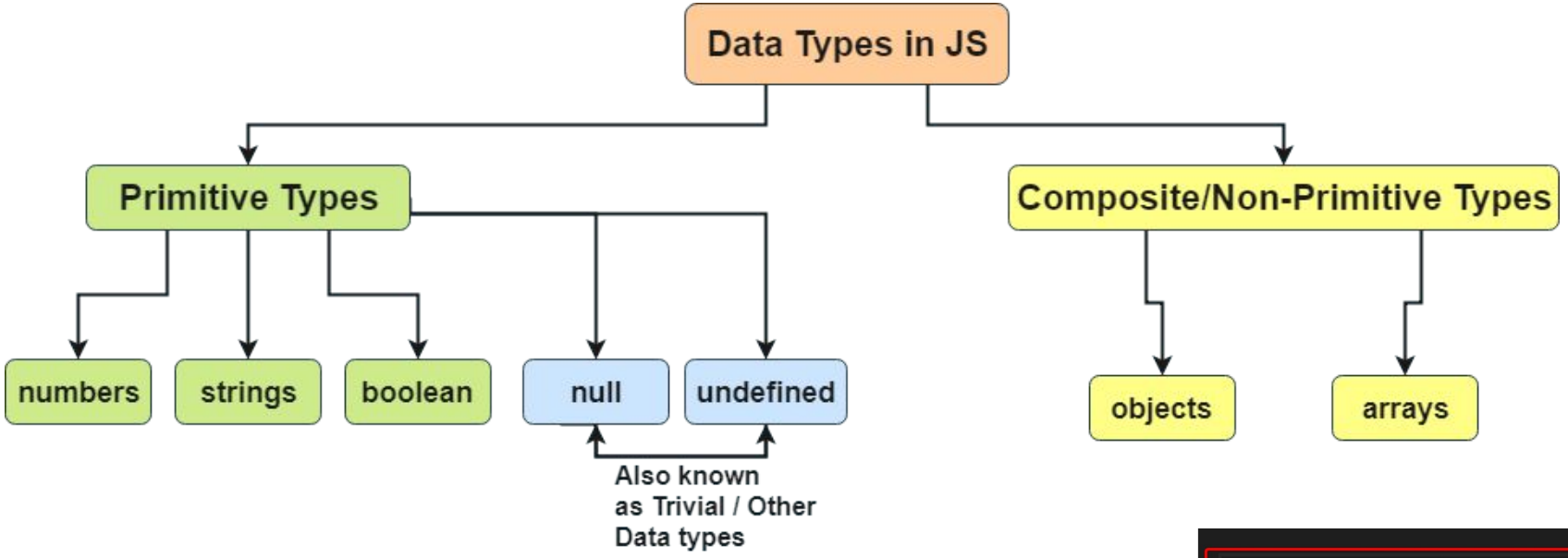
Javascript Basic P2
Object

Trần Huy Hoàng
Email: huyhoang.tran6669@gmail.com
Phone: 0788.719.666

Nội dung

1. Giới thiệu về Object
2. Các phương thức của Objects (Object method & this)
3. Định nghĩa, khởi tạo một Objects (Object constructor)
4. Lab
5. Destructuring Assignment
6. Higher Order Function
7. Js Setter và getter
8. Homeworks

Kiểu dữ liệu trong Javascript



Để kiểm tra kiểu dữ liệu của 1 biến, chúng ta sử dụng toán tử **typeof**

```
let name = "Thuan nguyen";
console.log(typeof name);
```

String

1. Giới thiệu về Object

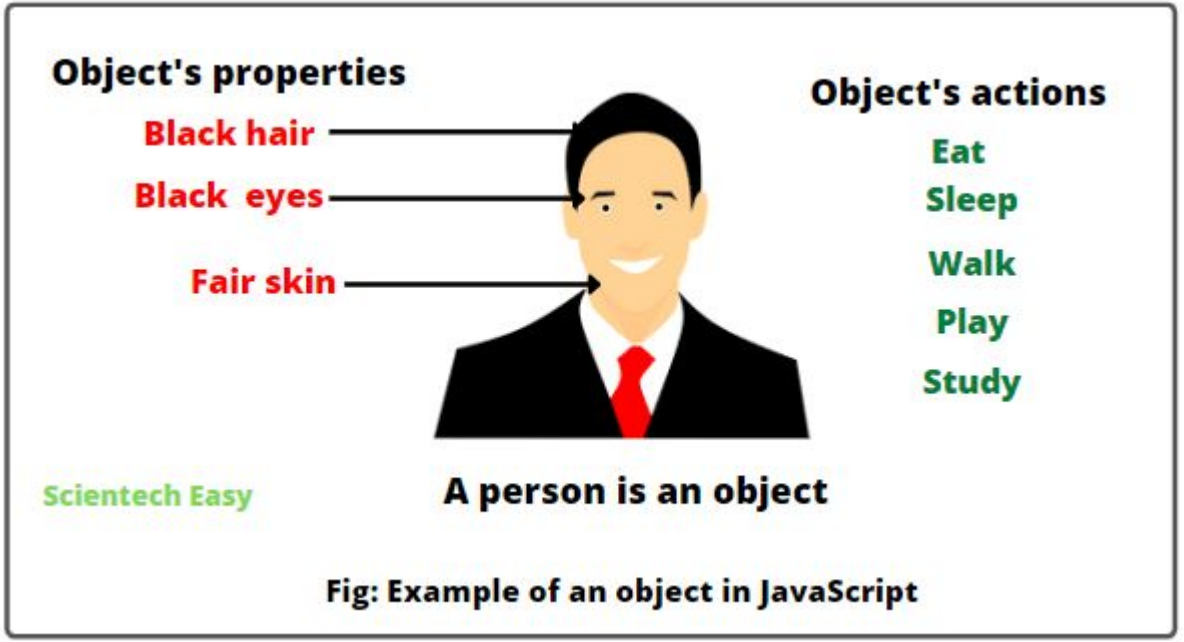
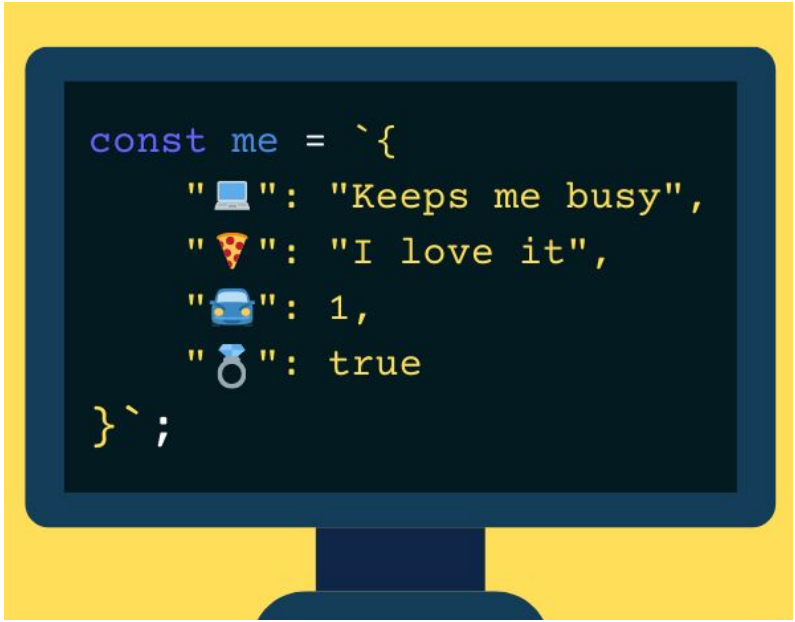
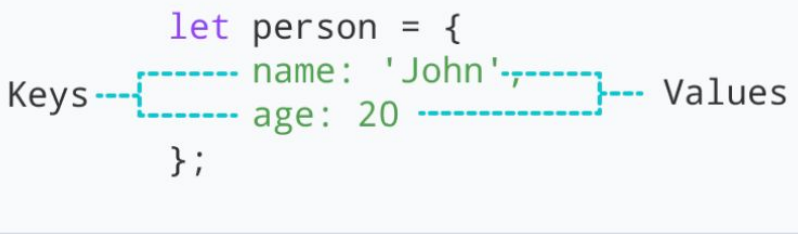


Fig: Example of an object in JavaScript

1. Giới thiệu về Object




```
// object
const student = {
    firstName: 'ram',
    class: 10
};
```



JavaScript object properties

1. Giới thiệu về Object

Về mặt định nghĩa, một đối tượng (một object) là một danh sách các item, mỗi item là một cặp name-value. Trong đó value có thể là: các kiểu dữ liệu cơ bản, function, hay cũng có thể là một object khác (kiểu dữ liệu phức hợp)

```
// object creation
const person = {
    name: 'John',
    age: 20
};
console.log(typeof person); // object
```

```
const object_name = {
    key1: value1,
    key2: value2
}
```

1. Giới thiệu về Object

Cách thức truy cập đến các thuộc tính (properties)

- C1. Using dot Notation
- C2. Using bracket Notation

```
objectName.key
```

```
objectName["propertyName"]
```

1. Giới thiệu về Object

Nested Objects (đối tượng nằm trong đối tượng)

Một Object có thể chứa một hoặc nhiều đối tượng nằm trong nó

```
// nested object
const student = {
    name: 'John',
    age: 20,
    marks: {
        science: 70,
        math: 75
    }
}

// accessing property of student object
console.log(student.marks); // {science: 70, math: 75}

// accessing property of marks object
console.log(student.marks.science); // 70
```

1. Giới thiệu về Object

Object methods

Một Object có thể chứa một hoặc nhiều hàm con nằm trong nó

```
const person = {
    name: 'Sam',
    age: 30,
    // using function as a value
    greet: function() { console.log('hello') }
}

person.greet(); // hello
```

2. Các phương thức của Objects

```
let student = {
    name: "Nguyễn Văn A",
    age: 19,
    email: "nguyenvana@gmail.com"
}
```

1.Lấy danh sách Key của object

```
console.log(Object.keys(student));
```

2.Lấy danh sách Value của object

```
console.log(Object.values(student));
```

3.Kiểm tra một key có nằm trong Object

```
console.log(student.hasOwnProperty("email"));
console.log(student.hasOwnProperty("name"));
```

2. Các phương thức của Objects

```
// creating an object
let student = { };

// adding a property
student.name = 'John';

// adding a method
student.greet = function() {
    console.log('hello');
}

// accessing a method
student.greet(); // hello
```

Khởi tạo method với object

1. Khởi tạo một object rỗng
2. Thêm property vào object
3. Thêm method vào object
4. Truy cập đến method được khai báo

2. Các phương thức của Objects

Từ khóa this

Để truy cập đến các properties của object trong chính object đó chúng ta sử dụng từ khóa **This**

```
const person = {
    name: 'John',
    age: 30,

    // accessing name property by using this.name
    greet: function() { console.log('The name is' + ' ' + this.name); }
};

person.greet();
```

2. Các phương thức của Objects

Thay đổi giá trị của thuộc tính trong object

```
let user = {
    name: "Nguyễn Văn A",
    age: 23,
    email: "abc@gmail.com"
}

// Thay đổi giá trị của thuộc tính name thành "Bùi Hiên"
user.name = "Bùi Hiên"

// Tương tự khi muốn thay đổi giá trị của thuộc tính age, email
user.age = 30
user.email = "hien@techmaster.vn"
```

2. Các phương thức của Objects

Thêm thuộc tính mới cho object

```
let user = {
    name: "Nguyễn Văn A",
    age: 23,
    email: "abc@gmail.com"
}

// Thêm thuộc tính address cho user
user.address = "Hà Nội"
```

2. Các phương thức của Objects

Xóa thuộc tính của object

```
let user = {
    name: "Nguyễn Văn A",
    age: 23,
    email: "abc@gmail.com"
}

// Xóa thuộc tính email của user
delete user.email
```

3. Định nghĩa, khởi tạo một Objects (Object constructor)

Hàm khởi tạo không có tham số

```
// constructor function
function Person () {
    this.name = 'John',
    this.age = 23
}

// create an object
const person = new Person();
```

```
// constructor function
function Person () {
    this.name = 'John',
    this.age = 23,

     this.greet = function () {
        console.log('hello');
    }
}

// create objects
const person1 = new Person();
const person2 = new Person();

// access properties
console.log(person1.name);  // John
console.log(person2.name);  // John
```

3. Định nghĩa, khởi tạo một Objects (Object constructor)

```
// constructor function
function Person (person_name, person_age, person_gender) {

   // assigning  parameter values to the calling object
    this.name = person_name,
    this.age = person_age,
    this.gender = person_gender,

    this.greet = function () {
        return ('Hi' + ' ' + this.name);
    }
}


// creating objects
const person1 = new Person('John', 23, 'male');
const person2 = new Person('Sam', 25, 'female');

// accessing properties
console.log(person1.name); // "John"
console.log(person2.name); // "Sam"
```

Hàm khởi tạo có tham số truyền vào

4. Lab

```
// Danh sách các sản phẩm có trong giỏ hàng
let products = [
    {
        name: "Iphone 13 Pro Max", // Tên sản phẩm
        price: 3000000, // Giá mỗi sản phẩm
        brand: "Apple", // Thương hiệu
        count: 2, // Số lượng sản phẩm trong giỏ hàng
    },
    {
        name: "Samsung Galaxy Z Fold3",
        price: 41000000,
        brand: "Samsung",
        count: 1,
    },
    {
        name: "IPhone 11",
        price: 15500000,
        brand: "Apple",
        count: 1,
    },
    {
        name: "OPPO Find X3 Pro",
        price: 19500000,
        brand: "OPPO",
        count: 3,
    },
]
```

```
// 1. In ra thông tin các sản phẩm trong giỏ hàng theo cấu trúc sau
// Tên - Giá - Thương Hiệu - Số lượng
// Ví dụ : OPPO Find X3 Pro - 19500000 - OPPO - 3

// 2. Tính tổng tiền tất cả sản phẩm trong giỏ hàng
// Tổng tiền mỗi sản phẩm = price * count

// 3. Tìm các sản phẩm của thương hiệu "Apple"

// 4. Tìm các sản phẩm có giá > 20000000

// 5. Tìm các sản phẩm có chữ "pro" trong tên (Không phân biệt hoa thường)

// 6. Thêm 1 sản phẩm bất kỳ vào trong mảng product

// 7. Xóa tất cả sản phẩm của thương hiệu "Samsung" trong giỏ hàng

// 8. Sắp xếp giỏ hàng theo price tăng dần

// 9. Sắp xếp giỏ hàng theo count giảm dần

// 10. Lấy ra 2 sản phẩm bất kỳ trong giỏ hàng
```

5. Destructuring Assignment

Theo định nghĩa chính thức trên MDN, “ **Destructuring Assignment** là một biểu thức Javascript cho phép lấy giá trị (value) từ bên trong (array) hay thuộc tính (properties) trong object và gán cho các biến (variable) mới”.


```
let person = {
    name: "Viet",
    age: 24,
    job: "dev",
};

// Lấy giá trị của object và gán cho biến khác
let name = person.name // name = 'Viet'
let age = person.age // age = 24
let job = person.job // job = 'dev'

// Cũng trường hợp trên nhưng sử dụng destructing
let { name, age, job } = person;
```

5. Destructuring Assignment

```
// Ví dụ với nested object (object lồng nhau)
let person = {
    info: {
        name: 'Viet',
        age: 24,
        job: 'dev'
    }
}

let { info: { name, age, job } } = person // name = 'Viet', age = 24, job='dev'

// Ví dụ với function trả về kết quả là Object
function getPerson() {
    return { name: "Hiên", age: "24" }
}

let { name, age } = getPerson

// Sử dụng rest operater
let obj = { a: 10, b: 20, c: 30, d: 40 }
let { a, b, ...rest } = obj // a - 10; b - 20; rest - {c: 30, d: 40}
```

5. Destructuring Assignment

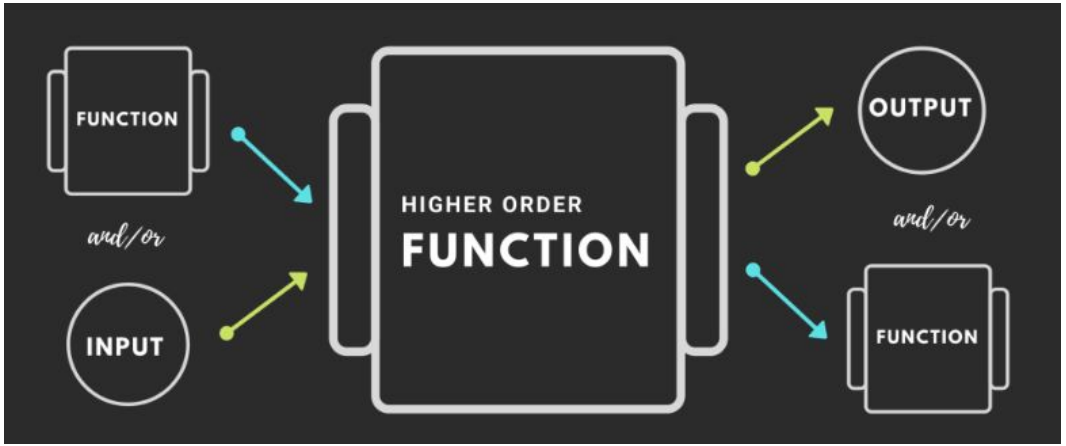
Sử dụng với Array

```
let number = [10, 20]

// Lấy giá trị của array và gán cho biến khác
let a = number[0]
let b = number[1]

// Cũng trường hợp trên nhưng sử dụng destructing
let [a, b] = number

// Gán thiếu phần tử
let number = [10, 20, 30, 40]
let [a, b] = number // a = 10; b = 20

// Gán với rest operator
let number = [10, 20, 30, 40, 50]
let [a, b, ...rest] = number; // a - 10; b - 20; rest - [30,40,50]

// Swap phần tử
let a = 1;
let b = 3;
[a, b] = [b, a];
// console.log(a); // 3
// console.log(b); // 1
```

6. Higher Order Function

Higher Order Function (HOF) là function thỏa mãn một trong các điều kiện sau:

- Nó nhận function khác vào làm tham số
- Nó trả về một function



6. Higher Order Function

```
function funA(a, b) {
    return function funB(c) {
        return a + b + c
    }
}

// Gọi function lần lượt
data = funA(3, 4) // => data ở đây là 1 fucntion được trả về từ funcA
console.log(data);
console.log(data(5));

// Gọi nối 2 lần
console.log(funA(4, 5)(6));
```

6. Higher Order Function

Nhận Function khác làm tham số

```
let arr = [1, 2, 3, 4, 5, 6, 7]
let filterNumber = function (ele) {
    return ele % 2 == 1
}

let newArr = arr.filter(filterNumber)
console.log(newArr);
```

7. Js Setter và getter

getter và **setter** là các hàm hoặc phương thức được dùng để lấy ra hoặc thiết lập giá trị cho các biến.

- Hữu ích khi một số hành động kiểu như là validation
- Đóng gói được thực hiện để ẩn dữ liệu, làm cho thành phần được đóng gói sẽ không thể bị truy cập, ngoại trừ việc thông qua các getter và setter

```
const student = {

    // data property
    firstName: 'Monica',

    // accessor property(getter)
    get getName() {
        return this.firstName;
    }
};

// accessing data property
console.log(student.firstName); // Monica

// accessing getter methods
console.log(student.getName); // Monica

// trying to access as a method
console.log(student.getName()); // error
```

```
const student = {
    firstName: 'Monica',

    //accessor property(setter)
    set changeName(newName) {
        this.firstName = newName;
    }
};

console.log(student.firstName); // Monica

// change(set) object property using a setter
student.changeName = 'Sarah';

console.log(student.firstName); // Sarah
```

7. Js Setter và getter

```
const student = {
    firstName: 'Monica'
}

// getting property
Object.defineProperty(student, "getName", {
    get : function () {
        return this.firstName;
    }
});

// setting property
Object.defineProperty(student, "changeName", {
    set : function (value) {
        this.firstName = value;
    }
});

console.log(student.firstName); // Monica

// changing the property value
student.changeName = 'Sarah';

console.log(student.firstName); // Sarah
```

Chống ghi đè

- Thay đổi các thuộc tính getter và setter trở thành **read-only**
- Các thuộc tính được khai báo thông qua Object.defineProperties, Object.defineProperty và Reflect.defineProperty tự động được cấu hình **writable**: **false**

8. Homeworks

```
const grades = [
 {name: 'John', grade: 8, sex: 'M'},
 {name: 'Sarah', grade: 12, sex: 'F'},
 {name: 'Bob', grade: 16, sex: 'M'},
 {name: 'Johnny', grade: 2, sex: 'M'},
 {name: 'Ethan', grade: 4, sex: 'M'},
 {name: 'Paula', grade: 18, sex: 'F'},
 {name: 'Donald', grade: 5, sex: 'M'},
 {name: 'Jennifer', grade: 13, sex: 'F'},
 {name: 'Courtney', grade: 15, sex: 'F'},
 {name: 'Jane', grade: 9, sex: 'F'}
]
```

1. Viết function tính thứ hạng trung bình của cả lớp
2. Viết function tính thứ hạng trung bình của nam trong lớp
3. Viết function tính thứ hạng trung bình của Nữ trong lớp
4. Viết function tìm học viên Nam có thứ hạng cao nhất trong lớp
5. Viết function tìm học viên Nữ có thứ hạng cao nhất trong lớp
6. Viết function tìm học viên Nam có thứ hạng thấp nhất trong lớp
7. Viết function tìm học viên Nữ có thứ hạng thấp nhất trong lớp
8. Viết function thứ hạng cao nhất của cả lớp
9. Viết function thứ hạng thấp nhất của cả lớp
10. Viết function lấy ra danh sách tất cả học viên Nữ trong lớp
11. Viết function sắp xếp tên học viên theo chiều tăng dần của bảng chữ cái
12. Viết function sắp xếp thứ hạng học viên theo chiều giảm dần
13. Viết function lấy ra học viên Nữ có tên bắt đầu bằng “J”
14. Viết function lấy ra top 5 người có thứ hạng cao nhất trong lớp

The End