VIETNAM AUSTRALIA Vocational College

Symbol, Object, Classes, Modules, JSON

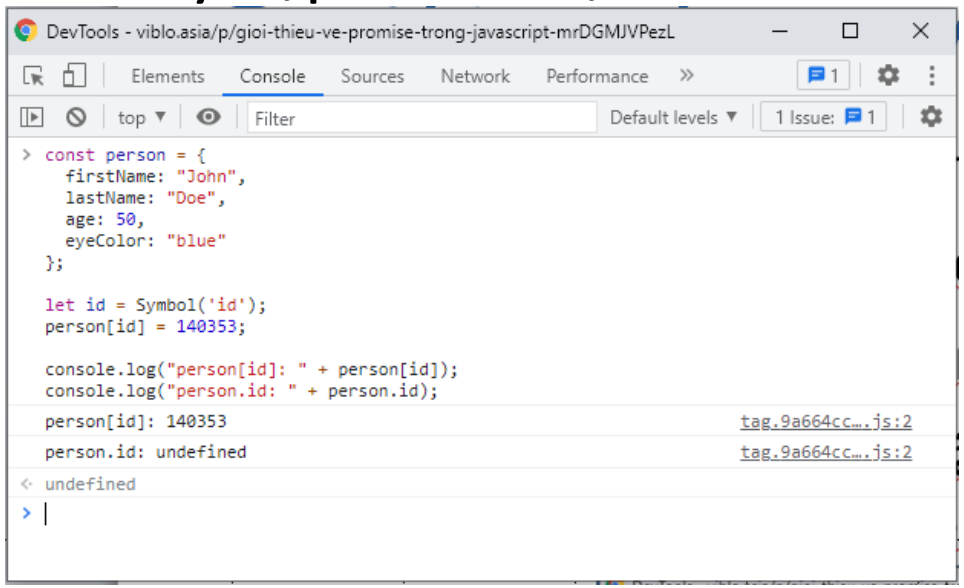
Mentor: Nguyễn Bá Minh Đạo

Nội dung:

1. Kiểu dữ liệu của biến, Symbol, typeof
2. Object, Classes, Constructor, Modules
3. Kiểu dữ liệu JSON, các phương thức của JSON
4. Tham trị, tham chiếu trong JavaScript

Kiểu dữ liệu của biến, Symbol, typeof

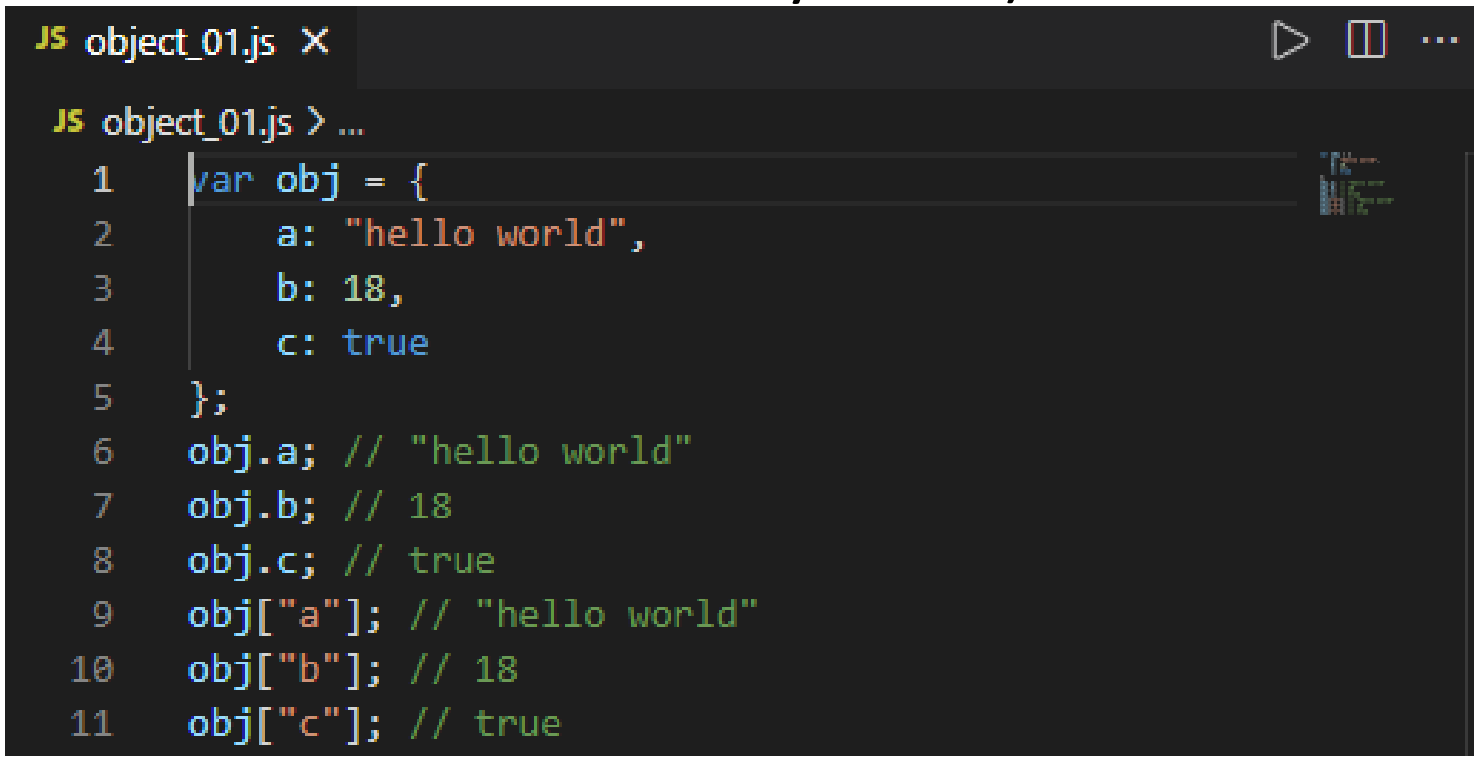
- **Kiểu của biến**: string, number, boolean, null và undefined, object, symbol
- **Kiểu Symbol** mới có trong **ES6:**
  - Là một kiểu dữ liệu nguyên thủy giống string, number, boolean.
  - Đại diện cho một mã định danh ẩn duy nhất mà không có mã nào khác có thể vô tình truy cập vào được.
  - Ví dụ:



Kiểu dữ liệu của biến, Symbol, typeof

- **JavaScript** có một **biểu thức typeof** có thể **kiểm tra giá trị** và **cho** bạn **biết kiểu** của **biểu thức đó** là gì.

  - **Giá trị trả lại từ biểu thức typeof** luôn là **1 trong 6 kiểu** ở **dạng giá trị string** (ES6 là 7 kiểu – thêm kiểm "symbol")



Kiểu dữ liệu của biến, Symbol, typeof

➢ **Biến box giữ mọi kiểu của giá trị, typeof box** sẽ trả về **kiểu** của **giá trị hiện tại trong biến box**, chứ không phải kiểu của biến box.

➢Lưu ý: **typeof null** là **trường hợp đặc biệt**, nó trả về sai kiểu thành **“object”**, khi mong muốn trả về **“null”**.

object_01.js

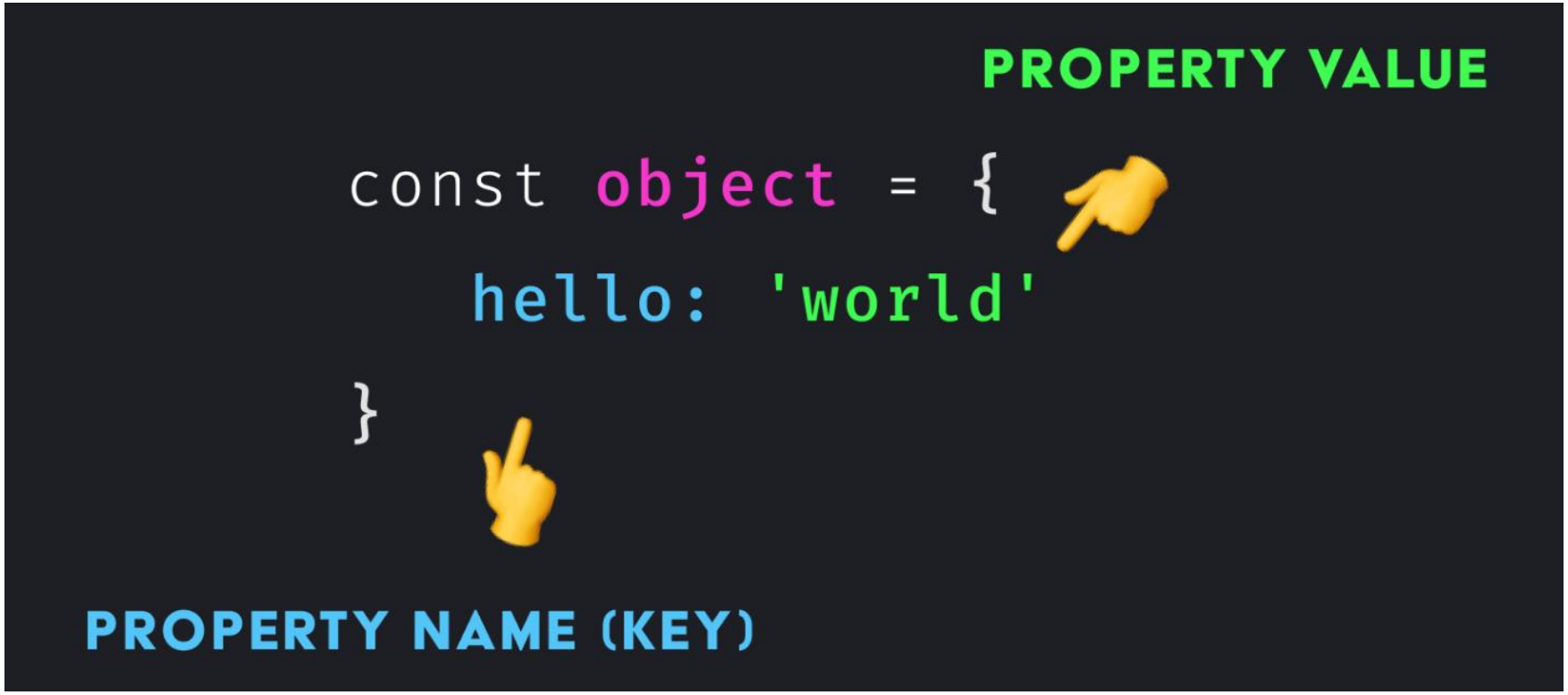
```
var obj = {
    a: "hello world",
    b: 18,
    c: true
};
obj.a; // "hello world"
obj.b; // 18
obj.c; // true
obj["a"]; // "hello world"
obj["b"]; // 18
obj["c"]; // true
```

Object, Classes, Constructor, Modules

- Kiểu **object** đề cập đến **một giá trị phức hợp** mà **có thể lập các thuộc tính**.
- Mỗi **thuộc tính** đều **có thể có giá trị của riêng chúng** với **bất kỳ kiểu nào**.
- Kiểu **object** là **một trong những kiểu hữu dụng nhất** trong JavaScript.

Object, Classes, Constructor, Modules

- **Object** chứa nhiều giá trị **gồm các thuộc tính** và **phương thức**.
- **Object** được **tạo bằng cặp ngoặc nhọn**.
- **Thuộc tính** trong **object gồm key** và **value**.



Object, Classes, Constructor, Modules

❑ **Cú pháp**:

```
var <object-name> = {
    key1: value1,
    key2: value2,
    ...
    keyN: valueN
};
```

❑ **Trong đó**:

➢ **key**: property/method

➢ **method**: function/function expression vai trò property trong **object**.

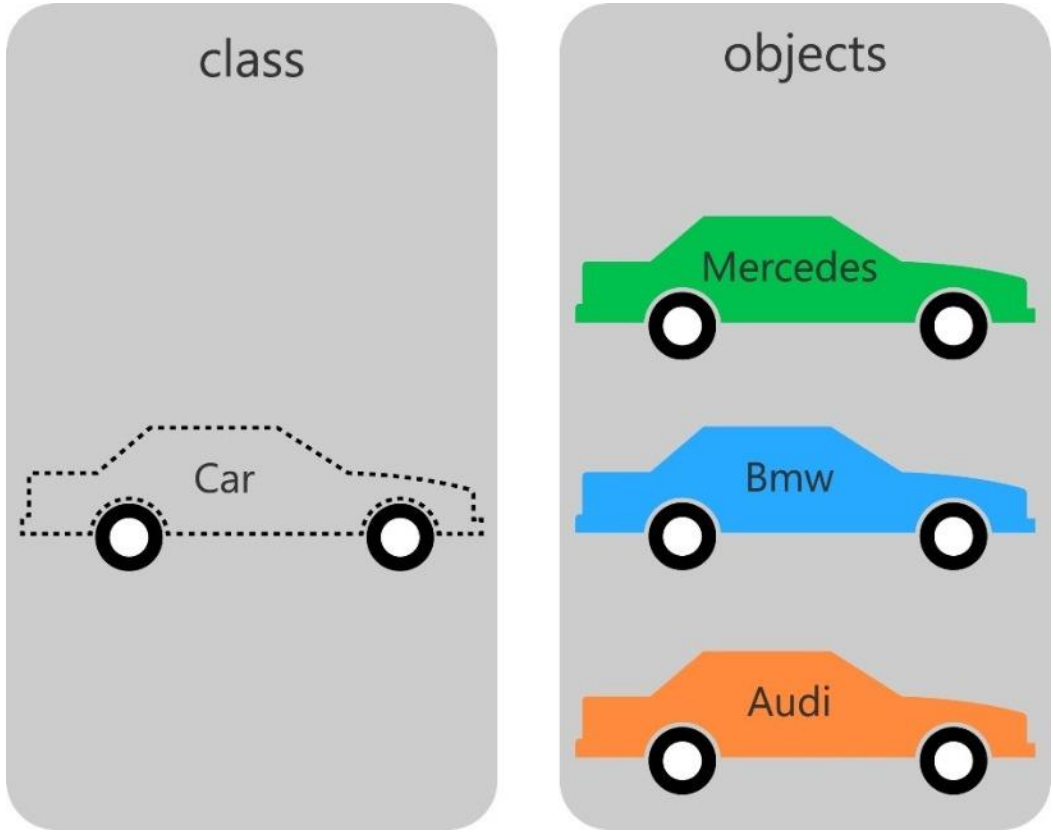
object_01.js

```
var obj = {
    a: "hello world",
    b: 18,
    c: true
};
obj.a; // "hello world"
obj.b; // 18
obj.c; // true
obj["a"]; // "hello world"
obj["b"]; // 18
obj["c"]; // true
```

Object, Classes, Constructor, Modules

- Trong **ES6**, JavaScript đã giới thiệu **từ khóa class** để làm mẫu cho các đối tượng trong JavaScript.

- Khi dùng **class**, chúng ta thường **thêm 1 phương thức** có tên **constructor()** trong class để làm **phương thức khởi tạo cho lớp đối tượng**.

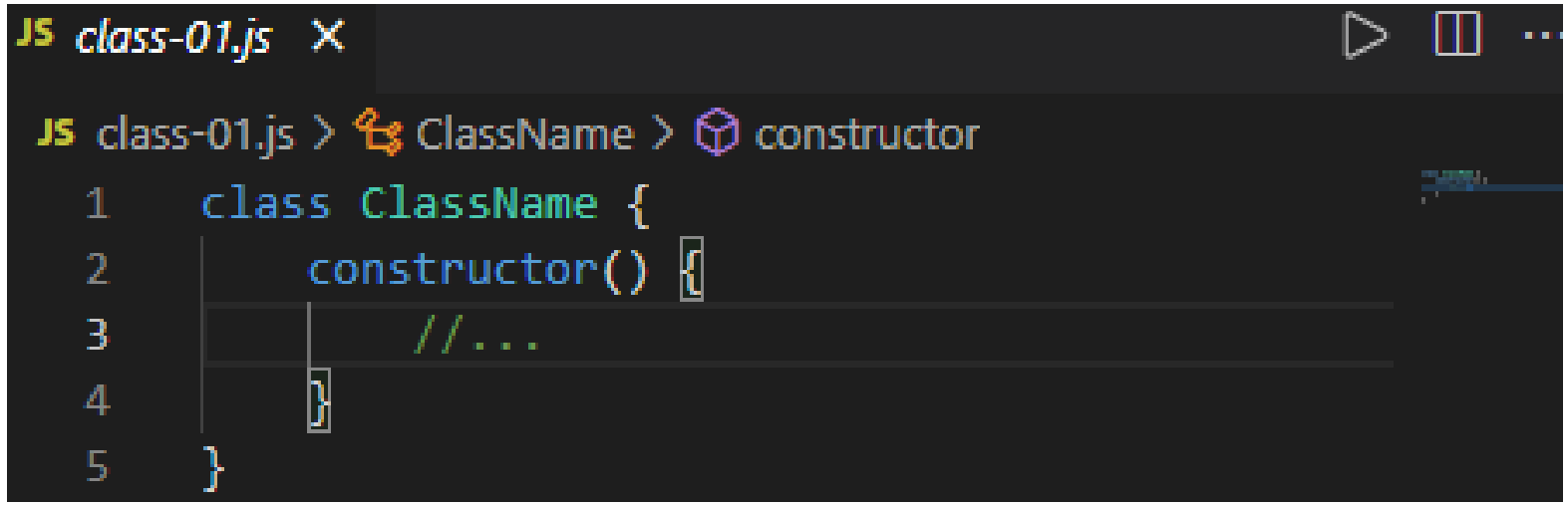


Object, Classes, Constructor, Modules

- Một **class** trong JavaScript **không phải là một đối tượng**.
- Cú pháp:

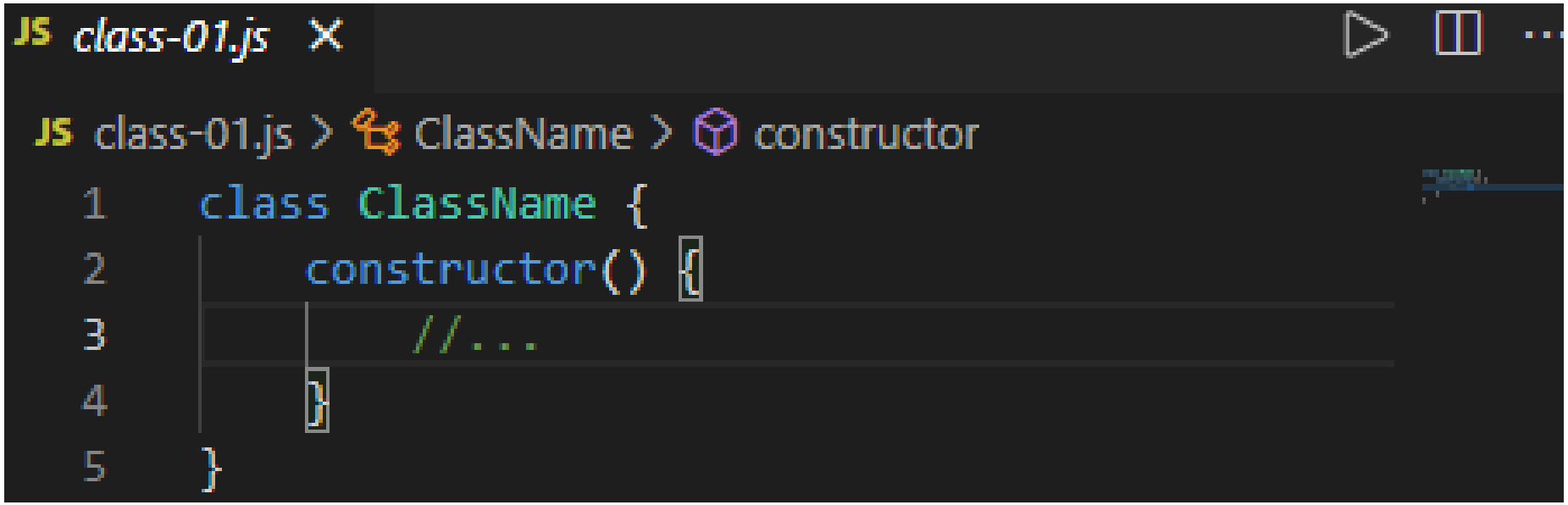
```
class ClassName {
    constructor() { ... }
}
```

- Tên **class** thường **viết hoa**, **không viết** theo kiểu **camel case**.



Object, Classes, Constructor, Modules

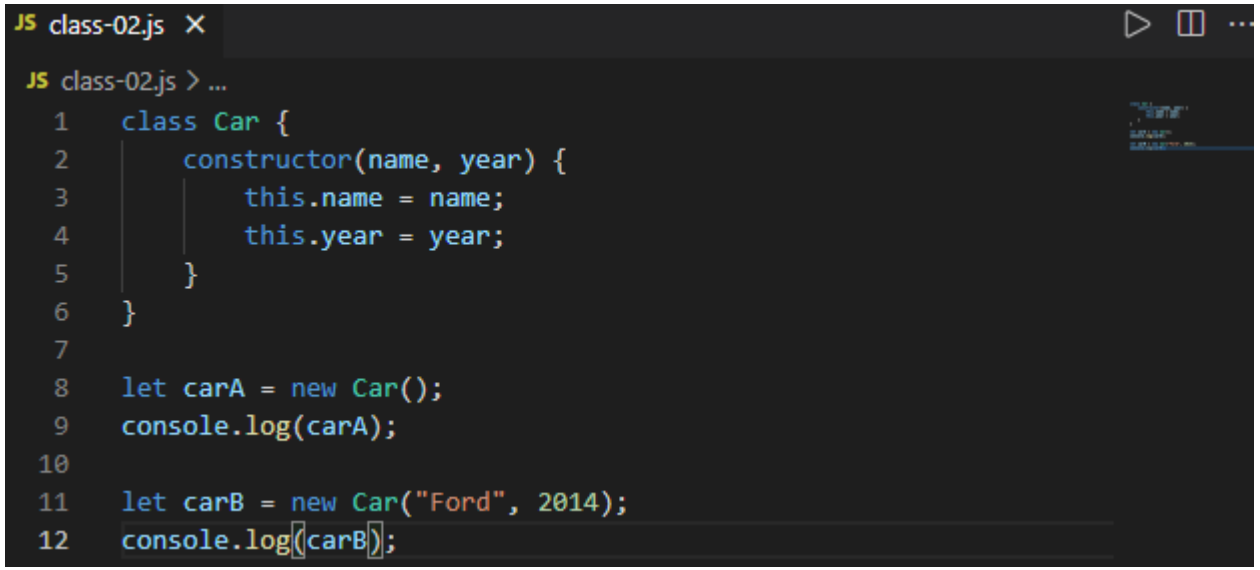
- Một class thường **có thể không có** hoặc **sẽ có 1 hoặc nhiều**:
  - Thuộc tính (properties)
  - Phương thức (method)
- **Lưu ý**: nếu dùng class để **khởi tạo đối tượng mà không xác định một phương thức khởi tạo**, **JavaScript** sẽ thêm **một phương thức khởi tạo rỗng**.



Object, Classes, Constructor, Modules

- Phương thức khởi tạo là **một phương thức đặc biệt**:
  - Được dùng với **từ khóa constructor.**
  - Được thực thi tự động khi một đối tượng được tạo.
  - Được dùng để khởi tạo các thuộc tính của đối tượng.
- Ví dụ:

```
class Car {
    constructor(name, year) {
        this.name = name;
        this.year = year;
    }
}
```



Object, Classes, Constructor, Modules

- **Các phương thức** của **class** được tạo với cú pháp **giống như các phương thức** của **đối tượng**.
- Chúng ta **có thể thêm vào class một** hoặc **nhiều phương thức** khác nhau.
- Ví dụ:

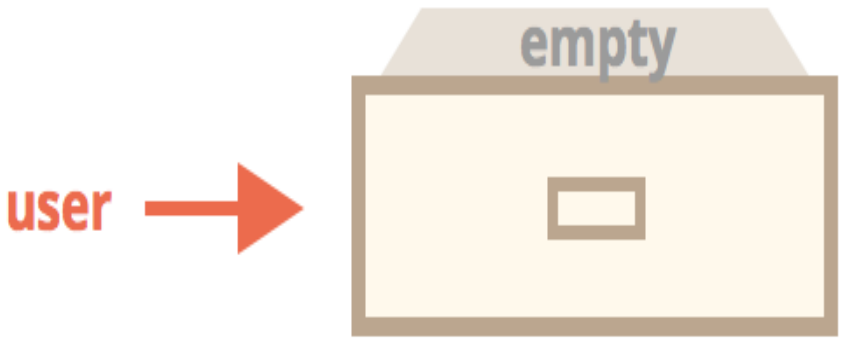
```
class ClassName {
    constructor() { ... }
    methodA() { ... }
    methodB() { ... }
    methodC() { ... }
}
```

```
class-03.js
class-03.js > ...
1   class Car {
2       constructor(name, year) {
3           this.name = name;
4           this.year = year;
5       }
6       age() {
7           let date = new Date();
8           return date.getFullYear() - this.year;
9       }
10  }
11
12  let myCar = new Car("Ford", 2014);
13  console.log("Xe của tôi đã " + myCar.age() + " tuổi.");
```

Object, Classes, Constructor, Modules

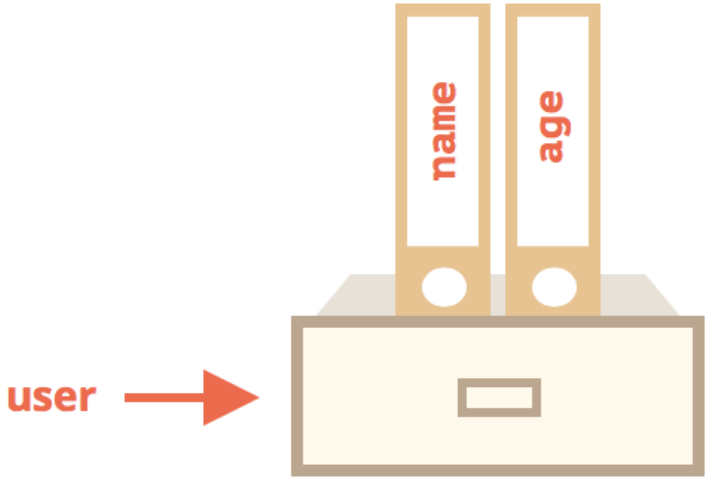
❑ **Tạo 1 object rỗng:**

```
var emptyObject = {
    // nothing here
    // ...
};
```



❑ **Tạo 1 object user có 2 thuộc tính:**

```
let user = {
    name: "Teo",
    age: 18
};
```



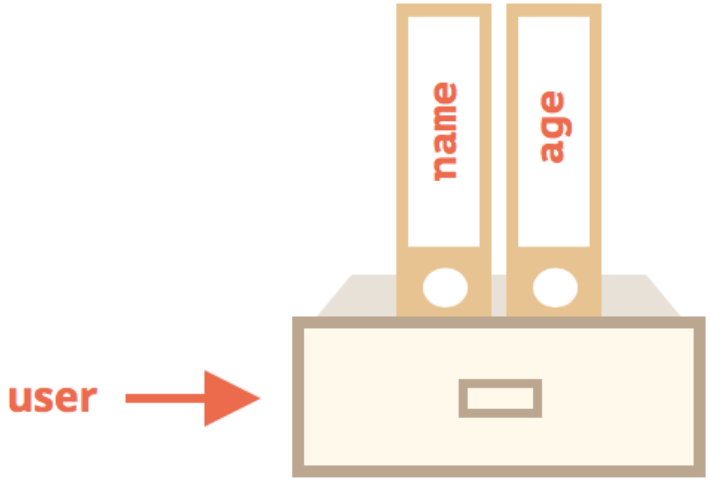
Object, Classes, Constructor, Modules

- **Tạo 1 object user có 2 thuộc tính và 1 biểu thức hàm:**

```
var user = {
    name: "Teo",
    age: 18,
    sayHello: function () {
        console.log("Hello World!");
    }
};
user.sayHello(); // Hello World!
```

properties

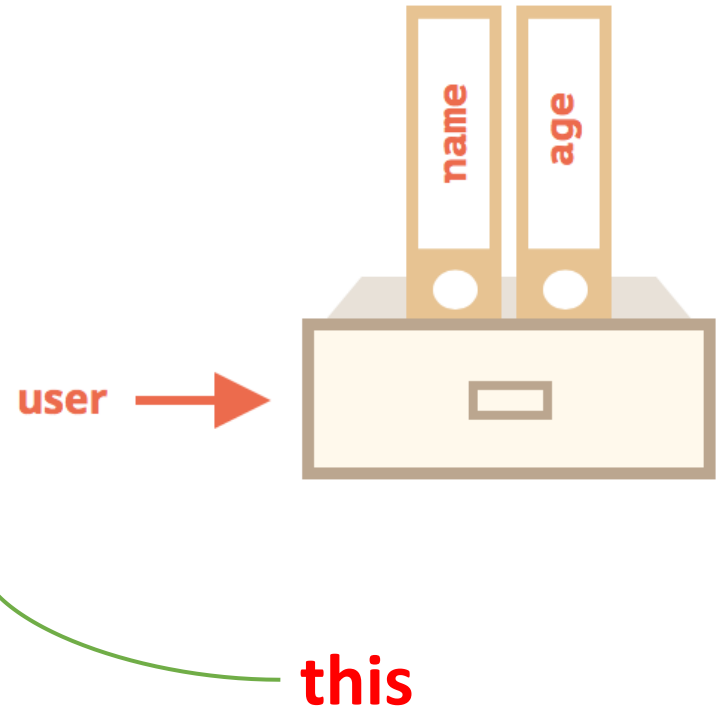
method (function expression)



Object, Classes, Constructor, Modules

- Khi **1 method của object muốn truy xuất các properties của chính object đó**, ta nên **sử dụng từ khóa this**:

```
var user = {
    name: "Teo",
    age: 18,
    sayName: function () {
        console.log(this.name);
    }
};
user.sayName(); // Teo
```



Object, Classes, Constructor, Modules

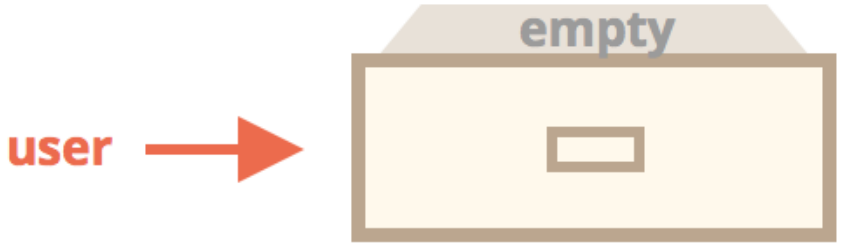
❑ Tạo 1 object với từ khóa new:

- ➢ Để tạo 1 object, ta còn có cách sử dụng từ khóa new như sau:

```
var <object-name> = new Object();
```

- ➢ Tạo 1 object rỗng:

```
var emptyObject = new Object();
```



JS new_01.js

JS new_01.js > ...

```
1   var emptyObject = new Object();
2   console.log(emptyObject); // {}
```

Object, Classes, Constructor, Modules

- Quan sát **giá trị** của **obj** một cách **trực quan** như sau:

obj

a: **"hello world"**	**b**: **18**	**c**: **true**

- **Thuộc tính** có thể **được truy cập bằng 2 cách**:
  - Dấu **chấm** (**.**) -> Ví dụ: **obj.a**
  - Dấu **ngoặc** (**[]**) -> Ví dụ: **obj["a"]**
- Dấu **chấm** **ngắn hơn** và **dễ đọc hơn** nên **thường được ưa thích hơn**.
- Dấu **ngoặc** **hữu dụng** khi bạn có **một tên thuộc tính có ký tự đặc biệt** trong đó (Ví dụ: obj["hello world**!**"])

Object, Classes, Constructor, Modules

- Quan sát **giá trị** của **obj** một cách **trực quan** như sau:

obj

a: “hello world”	b: 18	c: true

- Dấu **ngoặc hữu dụng** khi bạn có **một tên thuộc tính có ký tự đặc biệt** trong đó (Ví dụ: **obj["hello world!"]**)
- Dấu **ngoặc yêu cầu** phải có **một biến** hoặc **một string nguyên bản**.
- Một **string nguyên bản** thường **được bao bởi**:
  - Dấu **nháy kép** “..”
  - Dấu **nháy đơn** ‘..’

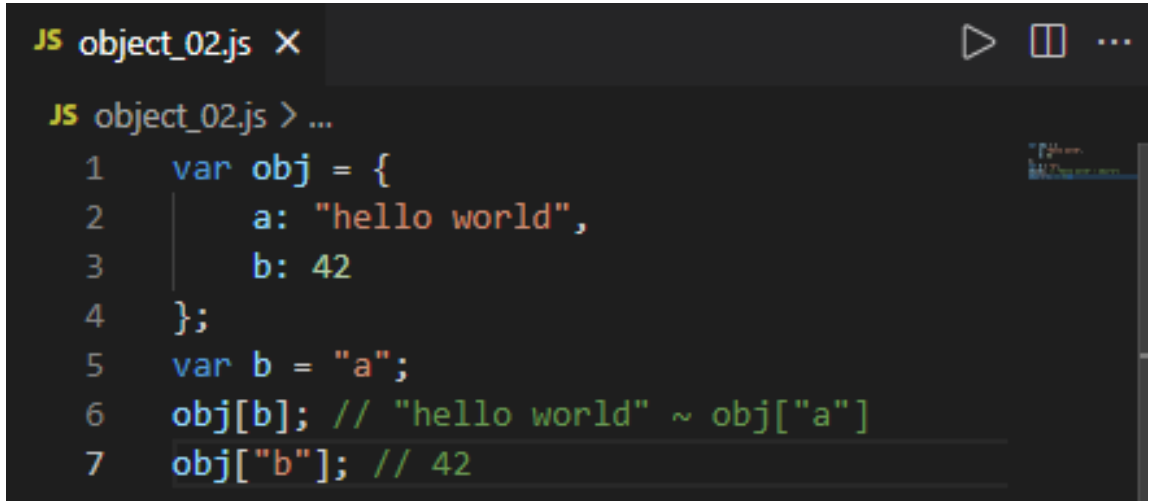
Object, Classes, Constructor, Modules

- Quan sát **giá trị** của **obj** một cách **trực quan** như sau:

obj

a: “hello world”	**b**: 18	**c**: true

- Dấu **ngoặc hữu dụng** khi bạn **muốn tiếp cận** một **thuộc tính/chìa khóa (key) nhưng tên được lưu trữ ở biến khác**. Ví dụ:



Object, Classes, Constructor, Modules

❑ **Thêm thuộc tính vào object**:

➢ Để **thêm mới thuộc tính** vào **object**, ta **dùng tên object** và **chấm (.) tên thuộc tính mới** muốn thêm vào **object**.

➢ **Ví dụ**:

object_06.js

```
var user = {
    name: "Teo",
    age: 18
};
user.isAdmin = true;
console.log(user); //{ name: 'Teo', age: 18, isAdmin: true }
```

Object, Classes, Constructor, Modules

❑ **Xóa thuộc tính của object**:

➢ Để **xóa thuộc tính** của **object**, ta **dùng từ khóa delete, khoảng trắng và chấm (.) tên thuộc tính của object** muốn xóa.

➢ **Ví dụ**:

object_07.js

```
var user = {
    name: "Teo",
    age: 18
};
user.isAdmin = true;
console.log(user); //{ name: 'Teo', age: 18, isAdmin: true }

delete user.age;
console.log(user); //{ name: 'Teo', isAdmin: true }
```

Object, Classes, Constructor, Modules

❑ **Lặp thuộc tính của object**:

➢ Để **lặp qua từng thuộc tính (key)** của một **object**, ta **dùng cấu trúc** sau: **key in object**

➢ **Ví dụ**:

object_08.js

```
var user = {
    name: "Ty",
    age: 20
};
user.isAdmin = true;

for (let key in user) {
    console.log(key);
} // name age isAdmin
```

Object, Classes, Constructor, Modules

- **keys, values, entries của object**:
  - **Một số hỗ trợ** của **đối tượng** (object) **phổ biến** như sau:
    - **Object.keys(obj)**: trả về **mảng các keys**
    - **Object.values(obj)**: trả về **mảng các values**
    - **Object.entries(obj)**: trả về **mảng các cặp [key, value]**

object_09.js

```
var user = {
    name: "Teo",
    age: 20
};

console.log(Object.keys(user)); // ['name', 'age']
console.log(Object.values(user)); // ['Teo', 20]
console.log(Object.entries(user)); // [['name', 'Teo'], ['age', 20]]
```

Object, Classes, Constructor, Modules

❑ Modules trong ES6:

➢ **Export/Import** giá trị: trước đây khi code JavaScript, tất cả các file mà có khởi tạo biến, hàm,…thì các file load sau nó đều có thể sử dụng được.

➢ Cú pháp:

- **export data**;
- **import name from path**;
- Trong đó:
  - **data:** những gì muốn xuất ra cho các module khác có thể sử dụng.
  - **name:** là biến muốn gán để nhận dữ liệu trả về từ module đó.
  - **path:** đường dẫn chứa module bạn cần import.

Object, Classes, Constructor, Modules

❑ Modules trong ES6:

Card.js

src > components > common > Card.js > ...

```
1  import React from "react";
2
3  function Card() {
4    return (
5      <div className="col-12 col-lg-6 col-xl">
6        <div className="card">
7          <div className="card-body">
8            <div className="row align-items-center">
9              <div className="col">
10               <h6 className="text-uppercase text-muted mb-2">Value</h6>
11               <span className="h2 mb-0">$24,500</span>
12             </div>
13             <div className="col-auto">
14               <span className="h2 fe fe-dollar-sign text-muted mb-0">Xem</span>
15             </div>
16           </div>
17         </div>
18       </div>
19     </div>
20   );
21 }
22
23 export default Card;
24
```

Admin.js

src > components > main > Admin.js > ...

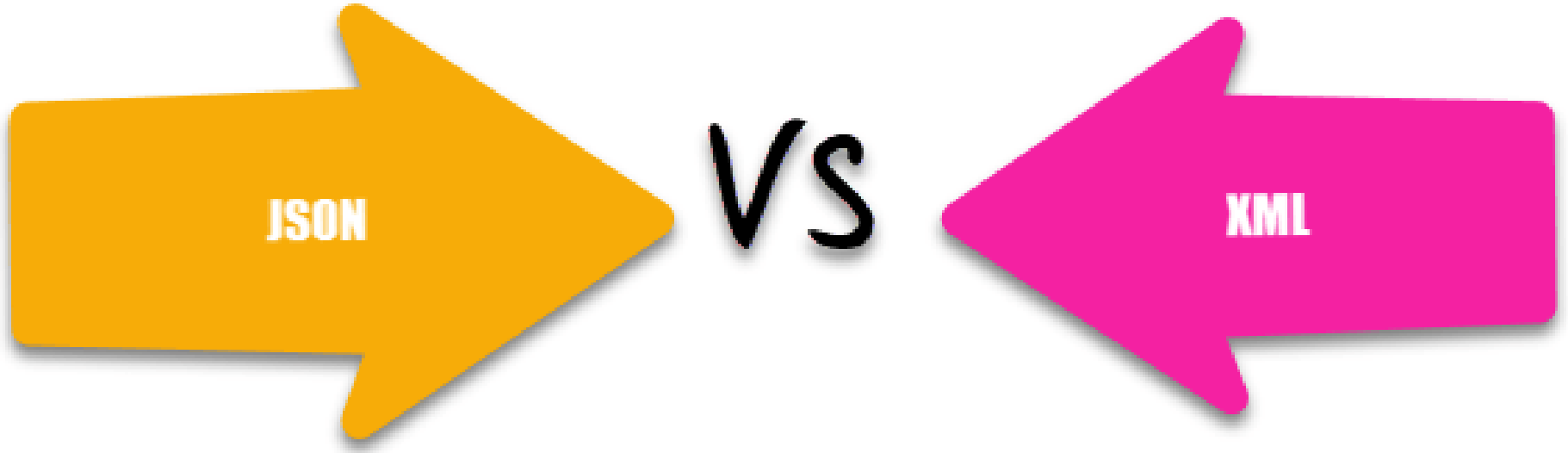
```
1  import React from "react";
2  import Card from "../common/Card";
3  import Navbar from "../common/Navbar";
4  import Table from "../common/Table";
5
6  function Admin() {
7    return (
8      <div className="main-content">
9        {" "}
10       <Navbar />
11       <div className="container-fluid">
12         <div className="row justify-content-center">
13           <div className="col-12">
14             <div className="row">
15               <Card />
16               <Card />
17               <Card />
18             </div>
19
20             <div className="card">
21               <div className="card-header">
22                 <h4 className="card-header-title">Tài khoản</h4>
23                 <div>
24                   <button className="btn btn-success">Tạo tài khoản</button>
25                 </div>
26               </div>
27               <Table />
28             </div>
29           </div>
30         </div>
31       </div>
32     </div>
33   );
34 }
35
36 export default Admin;
```

Kiểu dữ liệu JSON, các phương thức của JSON

❑ **Định nghĩa**:

➢ **JSON** (JavaScript Object Notation) là **định dạng giao đổi dữ liệu** (phổ biến) **cực kỳ nhẹ**

➢**JSON** được dùng **để giao đổi dữ liệu giữa máy chủ và máy khách**.

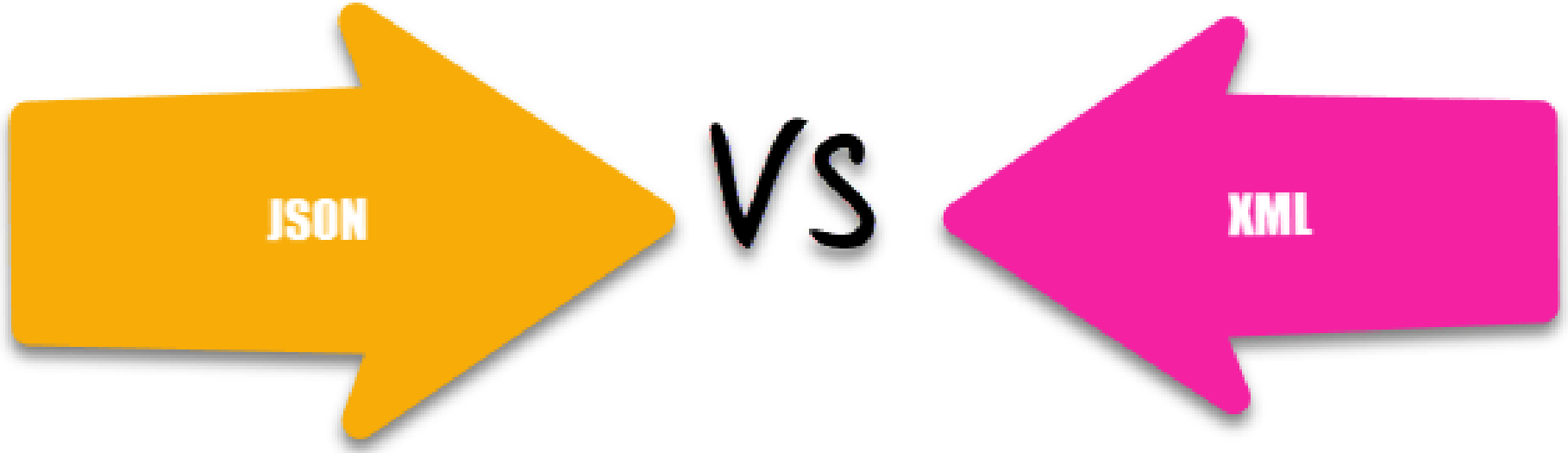


Kiểu dữ liệu JSON, các phương thức của JSON

❑ **Định nghĩa**:

➢ **Giống** như **XML**, **JSON** là **một dạng định dạng dựa trên văn bản** dễ viết và dễ hiểu cho cả người và máy tính.

➢ Nhưng **không giống XML**, cấu trúc **JSON** **chiếm ít băng thông hơn các phiên bản XML**.



Kiểu dữ liệu JSON, các phương thức của JSON

- **JSON** thường được dùng trong các ngôn ngữ lập trình hiện đại (trong đó có JavaScript).
  - **JavaScript** cung cấp **phương thức JSON.stringify()** để **chuyển đổi giá trị JavaScript thành một chuỗi JSON** như sau:

json_01.js

```
// Đối tượng trong JS
var obj = {
    "name": "Teo",
    "age": 18
};

// Chuyển đối tượng JS thành chuỗi JSON
var json = JSON.stringify(obj);
console.log(json); // {"name":"Teo","age":18}
console.log(typeof json); // string
```

Kiểu dữ liệu JSON, các phương thức của JSON

- **JSON** thường được dùng trong các ngôn ngữ lập trình hiện đại (trong đó có JavaScript).
  - **JavaScript** cung cấp **phương thức JSON.parse ()** để **chuyển đổi chuỗi JSON thành object trong JavaScript** như sau:

json_02.js

```
// Lưu chuỗi JSON như vào biến JS
var json = '{"name": "Teo","age": 18}';
console.log(json); // {"name":"Teo","age":18}
console.log(typeof json); // string

// Chuyển đổi chuỗi JSON thành đối tượng trong JS
var obj = JSON.parse(json);
console.log(obj); // {"name":"Teo","age":18}
console.log(typeof obj); // object

// Truy cập phần tử riêng lẻ như đối tượng trong JS
console.log(obj.name); // Teo
console.log(obj.age); // 18
```

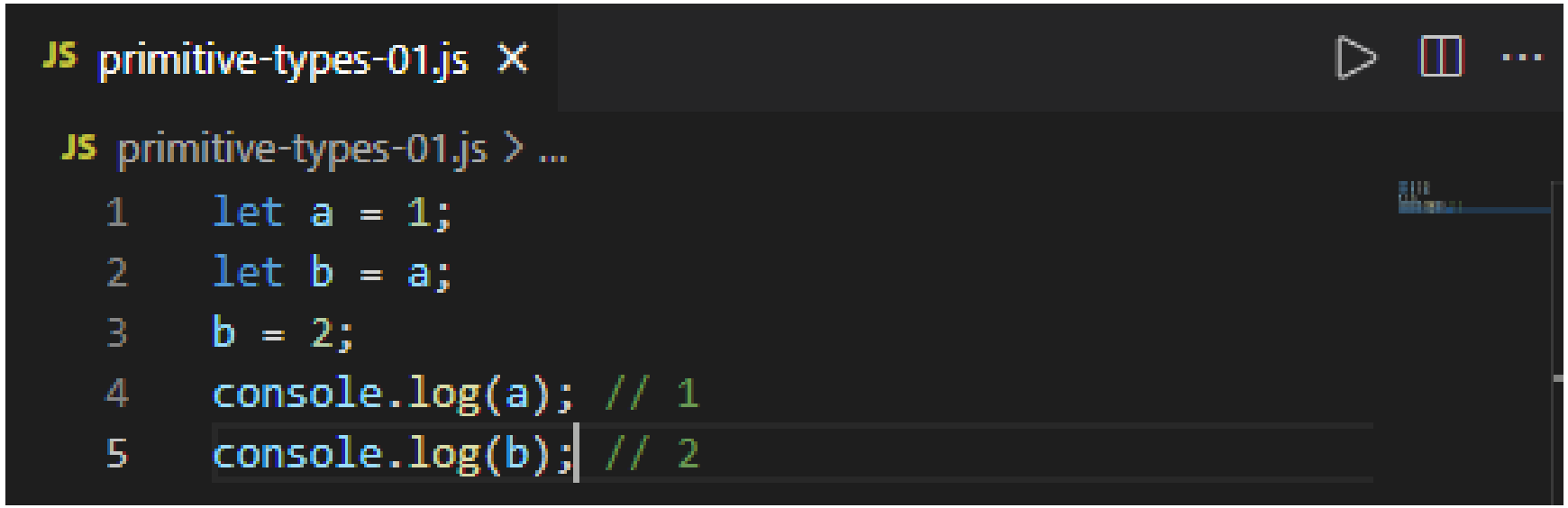
Tham trị và tham chiếu trong JavaScript

❑ **Tham trị với primitive data type**:

➢ Có **6 kiểu dữ liệu nguyên thủy** (primitive data type): **undefined**, **boolean**, **number**, **string**, **bigint**, **symbol**.

➢ Khi **ta gán giá trị biến này cho biến khác** thì **giá trị của 2 biến này độc lập** và **không liên quan với nhau** nữa.

➢ **Ví dụ**:



Tham trị và tham chiếu trong JavaScript

- **Tham trị với primitive data type**:
  - **Ví dụ**:

primitive-types-01.js

```
1   let a = 1;
2   let b = a;
3   b = 2;
4   console.log(a); // 1
5   console.log(b); // 2
```

  - Dù gán **b = a**, nhưng khi **b** thay đổi thì **a** vẫn không thay đổi.
  - Khi **giá trị thuộc kiểu dữ liệu nguyên thủy**, **biến sẽ chứa giá trị của biến đó**.

Tham trị và tham chiếu trong JavaScript

❑ **Tham chiếu với Object**:

➢ Trong **JavaScript**: **object, array, function** đều **được coi là object**.

➢ Khi **gán 1 biến thuộc kiểu object** thì **biến đó** chỉ **lưu địa chỉ của giá trị đó trên vùng nhớ, không lưu giá trị được gán**.

➢ **Ví dụ**:

reference_types.js

```
let cars1 = ['BMW', 'Mercedes'];
let cars2 = cars1;
cars2 = ['Toyota', 'Hyundai'];
console.log(cars1); // ['BMW', 'Mercedes']
console.log(cars2); // ['Toyota', 'Hyundai']
```

Tổng kết:

- Kiểu dữ liệu của biến, Symbol, typeof
- Object, Classes, Constructor, Modules
- Kiểu dữ liệu JSON, các phương thức của JSON
- Tham trị, tham chiếu trong JavaScript