

Biến, hằng, toán tử, biểu thức, các lệnh cơ bản

Mentor: Nguyễn Bá Minh Đạo



Nội dung:

1. Chú thích, các cú pháp căn bản
2. Biến, mục tiêu biến, hằng trong ES6
3. Giá trị biến, các kiểu giá trị nguyên thủy
4. Chuyển đổi giữa các kiểu dữ liệu
5. Biểu thức, toán tử, toán tử điều kiện
6. Khối lệnh, cách lệnh JavaScript căn bản



Chú thích, các cú pháp căn bản

❑ Chú thích (Comment) trong JavaScript:

- Sử dụng để **giải thích code JavaScript** và **giúp cho code dễ đọc hơn**.
- Sử dụng để **ngăn chặn** việc **thực thi code JavaScript**.

```
JS learn-comment.js X
JS learn-comment.js
1  //console.log('Xin chào các bạn VUSer!');
2
3  /*console.log('Chào mừng các bạn đến với lớp KTLT!');*/
4
5  console.log('2 dòng trên bị chú thích lại rồi, không chạy đâu!');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL cmd + - ^ X

D:\workspace\programming-techniques\lesson-01\lesson-01-slides\vscode>node learn-comment.js
2 dòng trên bị chú thích lại rồi, không chạy đâu!

D:\workspace\programming-techniques\lesson-01\lesson-01-slides\vscode>

Ln 5, Col 66 Spaces: 4 UTF-8 CRLF JavaScript



Chú thích, các cú pháp căn bản

❑ Trong **JavaScript**, chúng ta có 2 loại giá trị thường gặp:

➤ **Giá trị cố định (Fixed value)**: Là **loại giá trị không thay đổi** trong suốt quá trình hoạt động của chương trình (còn được gọi là **Literal value**).

- Có 2 loại phổ biến:

- **Number** (Kiểu số): **1.75** hoặc **175** hoặc ...

- **String** (Kiểu chuỗi): **"VUSer"** hoặc **"FastTracker"** hoặc...

```
JS literal-value.js
1 1.75
2 175
3 "VUSer"
4 "FastTracker"
```



Chú thích, các cú pháp căn bản

❑ Trong **JavaScript**, chúng ta có 2 loại giá trị thường gặp:

➤ **Giá trị biến đổi (Variable value)**: Là **loại giá trị có thể thay đổi** trong suốt quá trình hoạt động của chương trình.

- Kiểu biến thường bắt đầu bằng giá trị khởi tạo **let/var/const**,...

- Ví dụ: **let name** = "Dao Nguyen";

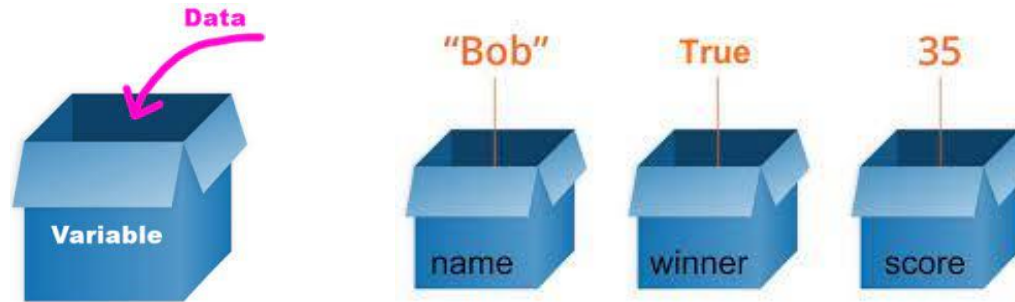
hoặc: **var height** = 1.75;

```
File Edit Selection View Go Run Terminal Help variable-value.js - vscode - Visual Studio ...  
EXPLORER  
VSCODE  
JS hello-world.js  
JS learn-comment.js  
JS literal-value.js  
JS variable-value.js  
variable-value.js X  
JS variable-value.js > ...  
1 let name = "Dao Nguyen";  
2 var height = 1.75;  
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL  
cmd + ^ X  
D:\workspace\programming-techniques\lesson-01\lesson-01-slides\vscode>
```



Biến, mục tiêu biến, hằng trong ES6

- ❑ **Biến** là một biểu tượng (giống như là một thùng chứa) có tên dùng để lưu trữ một giá trị dữ liệu.



- ❑ Có 2 kiểu giá trị phổ biến trong các ngôn ngữ lập trình:
 - **Static typing** (kiểu tĩnh): Tách biệt về kiểu dữ liệu của biến.
 - **Dynamic typing** (kiểu động): Linh động về kiểu dữ liệu của biến.
- ❑ **Biến** trong JavaScript là kiểu **Dynamic typing**.
 - **Khai báo biến** sử dụng từ khóa **var** (không có thông tin kiểu cụ thể khi khai báo biến)



Biến, mục tiêu biến, hằng trong ES6

❑ Ví dụ cho thấy **biến** trong **JavaScript** là **kiểu Dynamic typing**

➤ Quan sát sẽ thấy kiểu dữ liệu của biến **amount** bị tự động thay đổi kiểu nhiều lần trong quá trình hoạt động của chương trình.

JS variable.js X

JS variable.js > ...

```
1  var amount = 99.99;
2  amount = amount * 2;
3  console.log(amount); // 199.98
4
5  // chuyển `amount` sang một string,
6  // và thêm "$" ở đầu
7  amount = "$" + String(amount);
8  console.log(amount); // "$199.98";
```



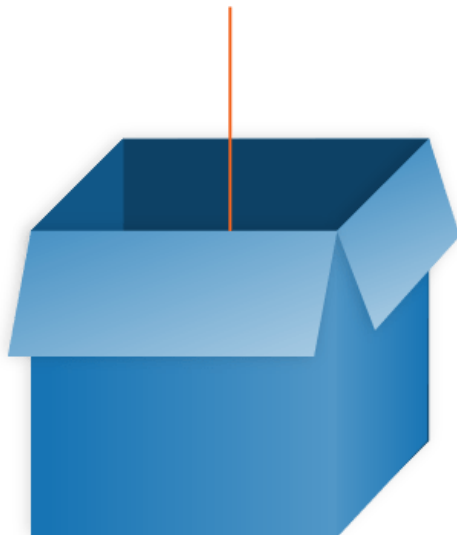
Biến, mục tiêu biến, hằng trong ES6

❑ Mục đích của biến:

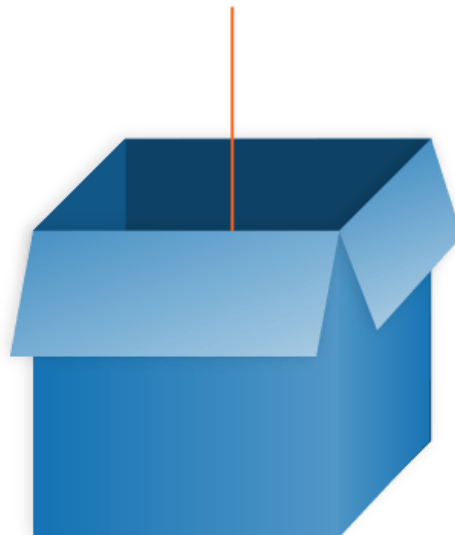
➤ Quản lý **state** của chương trình. **State** là **theo dõi các thay đổi của giá trị khi chương trình hoạt động**.

➤ Ví dụ: **var box = "Bob"; -> box = true; -> box = 35;**

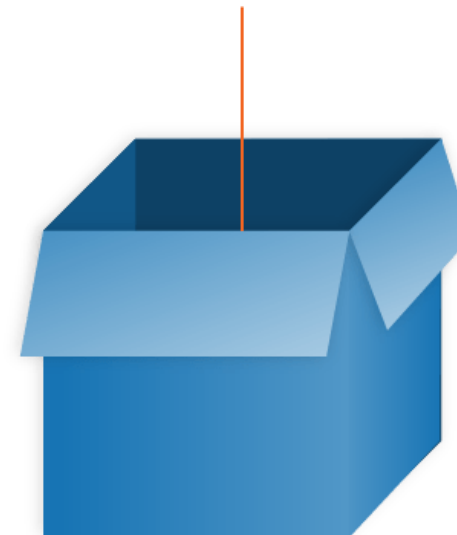
"Bob"



true



35



```
> var box = "Bob";  
< undefined  
  
> box  
< "Bob"  
  
> box = true;  
< true  
  
> box  
< true  
  
> box = 35;  
< 35  
  
> box  
< 35
```




Biến, mục tiêu biến, hằng trong ES6

- ❑ **Hằng** - biến **hằng** (constants), là cách **khai báo** một **biến** với một **giá trị xác định** và **giá trị này không thay đổi trong suốt chương trình**.
- ❑ Hằng **thường khai báo ở đầu chương trình**, tiện để **thay đổi khi cần**.
- ❑ Các **biến hằng** thường **được viết hoa**, có **gạch dưới _** giữa các liên từ.

```
JS constant.js X
JS constant.js > ...
1  var TAX_RATE = 0.08; // 8% sales tax
2  var amount = 99.99;
3  amount = amount * 2;
4  amount = amount + (amount * TAX_RATE);
5  console.log(amount); // 215.9784
6  console.log(amount.toFixed(2)); // "215.98"
```

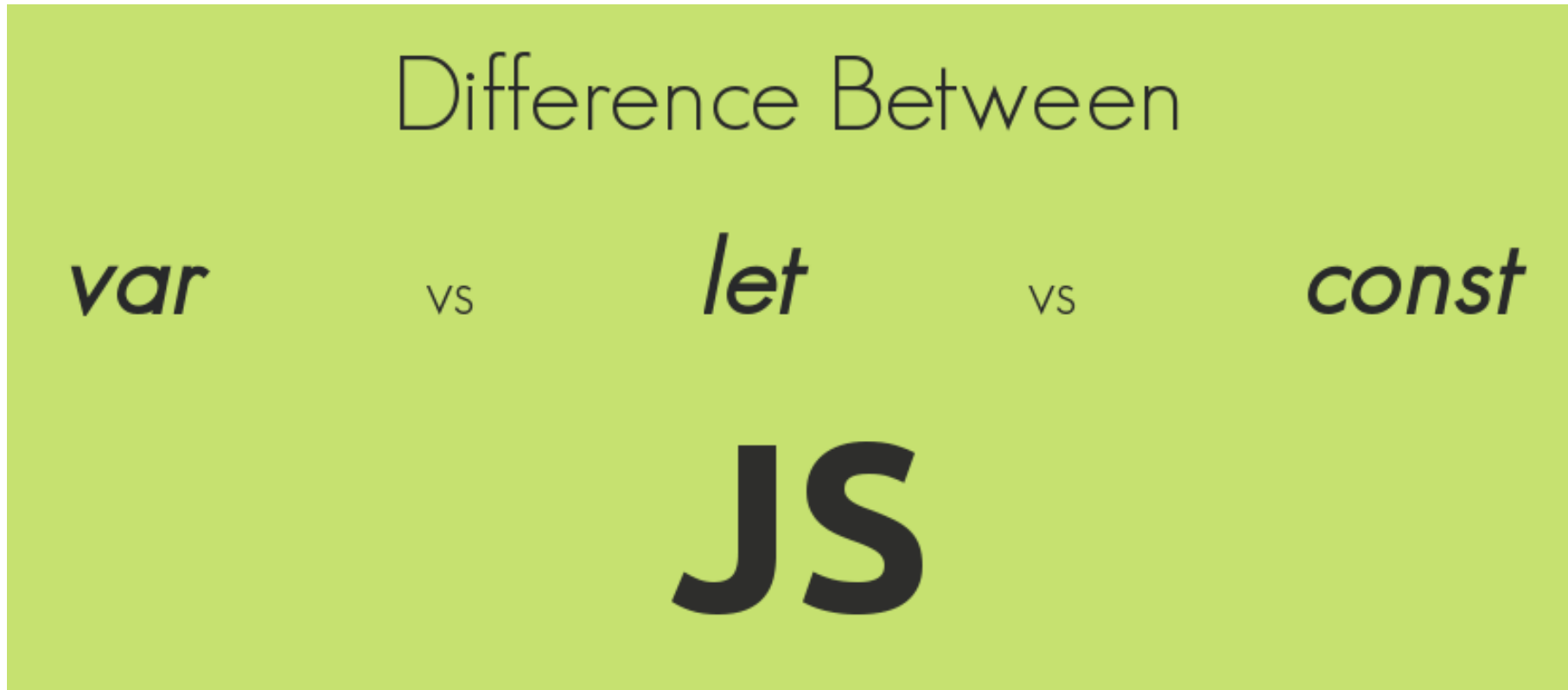
The screenshot shows a VS Code terminal window with a dark theme. The title bar at the top says 'JS constant.js X'. The terminal content shows a series of JavaScript code lines. Line 1 declares a constant 'TAX_RATE' with a value of 0.08 and a comment '// 8% sales tax'. Line 2 declares a variable 'amount' with a value of 99.99. Line 3 doubles the amount. Line 4 adds 8% tax to the amount. Line 5 logs the amount, showing '215.9784'. Line 6 logs the amount formatted to two decimal places, showing '"215.98"'. The bottom of the terminal shows tabs for 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL', with 'TERMINAL' being the active tab. There are also icons for 'cmd', a plus sign, a checkmark, and a close button.



Biến, mục tiêu biến, hằng trong ES6

❑ Hằng trong ES6:

➤ Phiên bản mới thông dụng nhất của JS hiện nay là **ES6** đã có cách khai báo **constants**, bằng cách sử dụng **const** thay cho **var**.





Biến, mục tiêu biến, hằng trong ES6

❑ Hằng trong ES6:

- **Const** giống như **var** với giá trị không đổi.
- **Const** khác **var** ở chỗ sẽ ngăn ngừa sự thay đổi giá trị vô tình xảy ra ở đâu đó sau giá trị khởi tạo.
- Nếu khi thay đổi **TAX_RATE** sau lần khai báo đầu tiên **sẽ bị lỗi**.

```
JS constant_es6.js X
JS constant_es6.js > ...
1  const TAX_RATE = 0.08; // 8% sales tax
2  var amount = 99.99;
3  amount = amount * 2;
4  TAX_RATE = 0.09;
5  amount = amount + (amount * TAX_RATE);
6  console.log(amount); // 215.9784
7  console.log(amount.toFixed(2)); // "215.98"

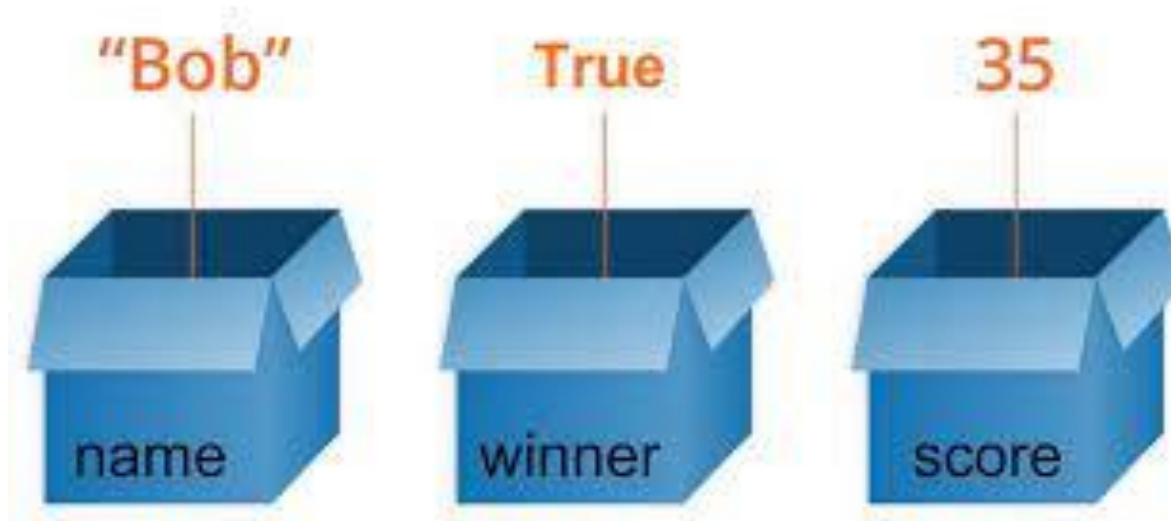
PROBLEMS  OUTPUT  TERMINAL  ...
D:\workspace\programming-techniques\lesson-02\lesson-02-slides\
ode constant_es6.js
D:\workspace\programming-techniques\lesson-02\lesson-02-slides\
constant_es6.js:4
TAX_RATE = 0.09;
    ^

TypeError: Assignment to constant variable.
    at Object.<anonymous> (D:\workspace\programming-techniques\
lesson-02\lesson-02-slides\constant_es6.js:4:10)
```



Giá trị biến, các kiểu giá trị nguyên thủy

- ❑ Nếu bạn hỏi nhân viên tại một cửa hàng điện thoại **giá của một cái điện thoại gì đó**, họ sẽ trả lời **“chín chín, chín chín”** (Ví dụ: \$99.99) -> tức họ đã cho bạn một hình dung **giá trị tiền bạn cần phải trả**.
- ❑ Sau đó, bạn lại hỏi **điện thoại có đồ sạc hay không**, câu trả lời có thể là **“có”** hoặc **“không”** -> **giá trị đồ vật mà bạn có thể có hoặc không** có khi mua chiếc điện thoại đó.

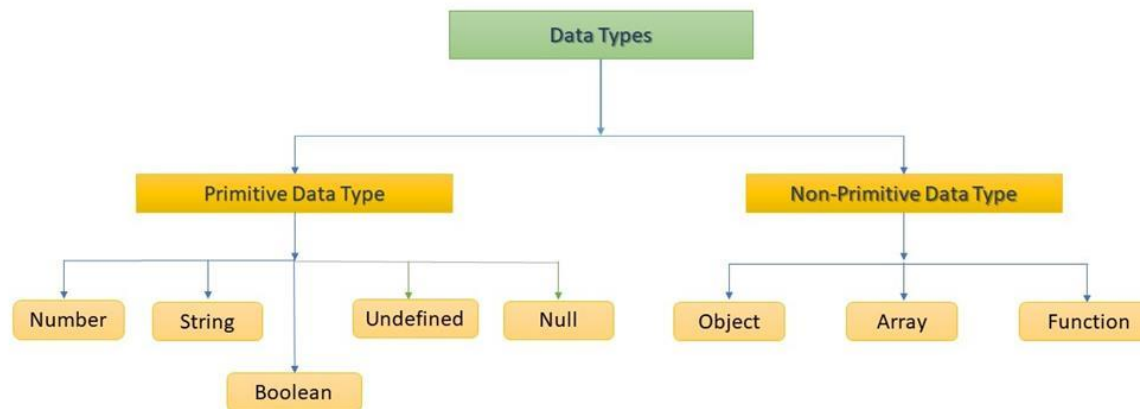




Giá trị biến, các kiểu giá trị nguyên thủy

❑ Các **kiểu giá trị nguyên thủy** (primitive types):

- Khi bạn muốn làm toán, bạn muốn **number** (kiểu số).
- Khi bạn muốn in giá trị trên màn hình, bạn cần **string** (kiểu chuỗi).
- Khi bạn muốn tạo một quyết định trên chương trình, bạn cần **boolean** (**true** hoặc **false**).
- Ngoài, **string/number/boolean**, còn có **arrays, objects, functions,...**






Chuyển đổi giữa các kiểu dữ liệu

❑ Chuyển đổi giữa các kiểu dữ liệu (conversion):

➤ Khi một **number** muốn hiển thị trên browser, bạn cần **chuyển đổi giá trị** thành **string**. Trong JavaScript gọi việc chuyển đổi này là “ép kiểu”.

▪ **Ví dụ:** Chuyển đổi ký tự số trên **form của trang bán hàng** từ **string** sang **number** để tính toán.

Shopping Cart (1 item in your cart)

	Name	Color	Quantity	Price	Total
	Samsung C7 Pro	<input checked="" type="radio"/> Silver	1	₹22000	₹22000

Order Total: ₹22000

[Continue Shopping](#) [Add to Cart](#)

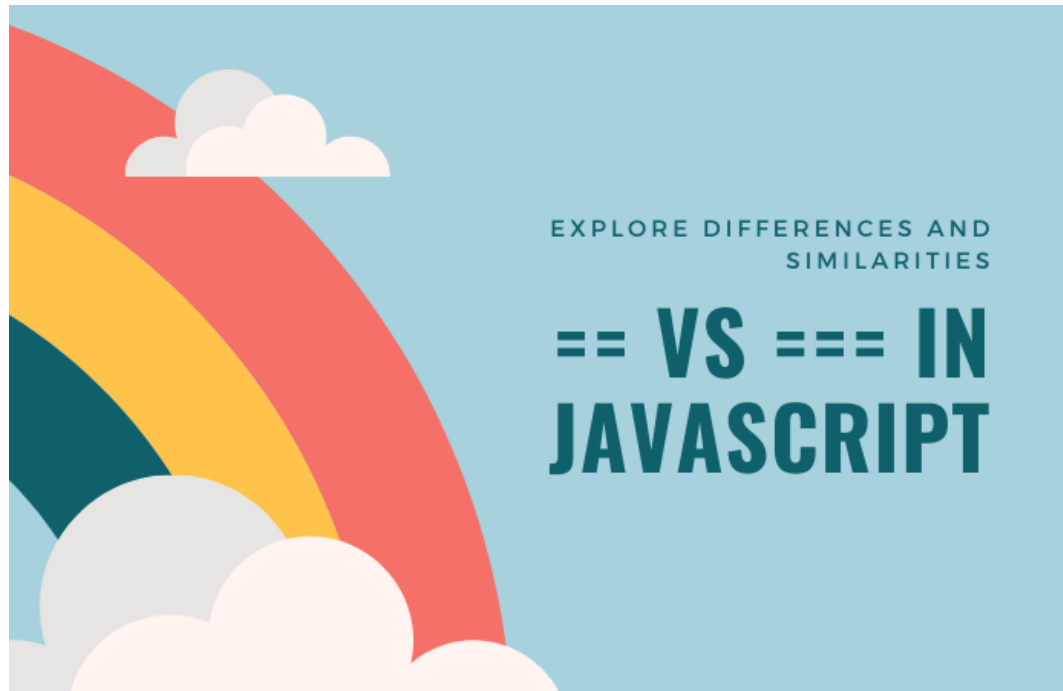


Chuyển đổi giữa các kiểu dữ liệu

❑ Chuyển đổi giữa các kiểu dữ liệu (conversion):

➤ JavaScript **cung cấp** một số **hàm có sẵn để hỗ trợ ép kiểu**:

- **Number(...)**: function hỗ trợ ép bất kỳ kiểu khác sang **number**.
- Dấu bằng tương đối **==**: Ép kiểu ngầm định khi so sánh chuỗi số với số.





Chuyển đổi giữa các kiểu dữ liệu

❑ Chuyển đổi giữa các kiểu dữ liệu (conversion):

➤ JavaScript **cung cấp** một số **hàm có sẵn để hỗ trợ ép kiểu**:

- Ví dụ: **"99.99" == 99.99**

- ♦ JavaScript sẽ ép kiểu ngầm định vế trái từ chuỗi số thành số

"99.99" -> 99.99

- ♦ Sau đó, mới so sánh với vế phải (99.99)

99.99 == 99.99 -> true

- **Lưu ý**: ép kiểu ngầm định rất dễ gây ra bug nếu không cẩn thận!



Biểu thức, toán tử, toán tử ba ngôi

❑ Biểu thức (expressions):

- **Biểu thức trong JavaScript** là **tập hợp các toán hạng** (các chữ số) và **toán tử** (các phép toán). Ví dụ: $5 + 4$ là một biểu thức, trong đó:
 - Chữ số **5, 4** là: toán hạng
 - Phép toán **+** là: toán tử
- Một số dạng biểu thức trong JavaScript:
 - **Gán:** **=** như $a = 2$
 - **Toán:** **+** (cộng), **-** (trừ), ***** (nhân), **/** (chia)
 - **Tổ hợp gán:** **+=**, **-=**, ***=**, **/=** là **tổ hợp kết hợp** giữa **các biểu thức toán** với **các giá trị gán**, ví dụ: $a += 2$ (tương tự: $a = a + 2$)
 - **Tăng/giảm:** **++** (tăng), **--** (giảm), ví dụ: $a++$ (tương tự: $a = a + 1$)



Biểu thức, toán tử, toán tử ba ngôi

□ Biểu thức (expressions):

➤ Một số dạng biểu thức trong JavaScript:

- **Tương đương:** `==` (bằng tương đối), `===` (bằng tuyệt đối), `!=` (khác tương đối), `!==` (khác tuyệt đối). Ví dụ: `a == b` hoặc `a === b`
- **So sánh:** `<` (nhỏ hơn), `>` (lớn hơn), `<=` (nhỏ hơn hoặc bằng tương đối), `>=` (lớn hơn hoặc bằng tương đối). Ví dụ: `a <= b`
- **Tính logic:** `&&` (và), `||` (hoặc). Ví dụ: `a || b` (chọn luôn a hoặc b), `a && b` (phải chọn cả a và b)



Biểu thức, toán tử, toán tử ba ngôi

❑ Toán tử (operators):

- **Toán tử** trong **JavaScript** có **chức năng giống toán tử trong toán học**.
- Tuy nhiên, **một vài toán tử** trong **JavaScript** có cách **viết khác so với cách viết toán tử trong toán học**.
- Danh sách **những toán tử cơ bản** trong JavaScript:

Toán tử	Tên gọi	Ví dụ	Kết quả
+	Phép cộng	10 + 4	14
-	Phép trừ	10 - 4	6
*	Phép nhân	10 * 4	40
/	Phép chia	10 / 4	2.5
%	Phép chia lấy phần số dư	10 % 4	2

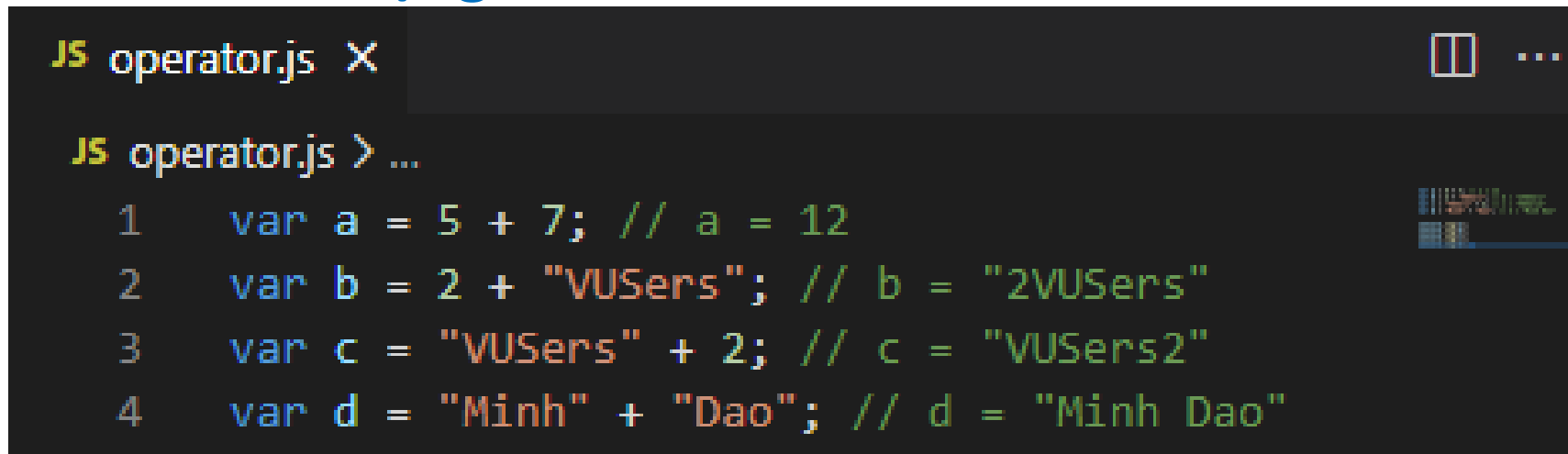


Biểu thức, toán tử, toán tử ba ngôi

❑ Toán tử (operators):

➤ **Phép cộng** trong JavaScript **tương đối khác** với phép cộng trong toán học. Trong JavaScript:

- Số có thể **cộng** số => cho **ra số**.
- Số có thể **cộng** chuỗi (hoặc chuỗi có thể **cộng** số) => cho **ra chuỗi**.
- Chuỗi có thể **cộng** chuỗi => cho **ra chuỗi**.



```
JS operator.js X
JS operator.js > ...
1  var a = 5 + 7; // a = 12
2  var b = 2 + "VUSers"; // b = "2VUSers"
3  var c = "VUSers" + 2; // c = "VUSers2"
4  var d = "Minh" + "Dao"; // d = "Minh Dao"
```



Biểu thức, toán tử, toán tử ba ngôi

❑ Toán tử (operators):

➤ Độ ưu tiên của toán tử:

Mức độ ưu tiên	Toán tử		
1	()		
2	*	/	%
3	+		-

➤ Ví dụ:

JS priority-operator.js ✕

JS priority-operator.js > ...

```
1 var result = 7 + 8 * (10 - 2) + 3 * 4;
```

```
2 console.log(result); // result = 83
```



Biểu thức, toán tử, toán tử ba ngôi

❑ Toán tử (operators):

➤ Ví dụ:

$7 + 8 * (10 - 2) + 3 * 4 \Rightarrow 7 + 8 * 8 + 3 * 4$

$7 + 8 * 8 + 3 * 4 \Rightarrow 7 + 64 + 3 * 4$

$7 + 64 + 3 * 4 \Rightarrow 7 + 64 + 12$

$7 + 64 + 12 \Rightarrow 71 + 12$

$71 + 12 \Rightarrow 83$

❑ Toán tử (operators):

➤ Khoảng trắng:

■ Trong JavaScript, dấu khoảng trắng giữa các toán hạng và toán tử **không ảnh hưởng đến kết quả phép toán**. Ví dụ: ba biểu thức dưới đây có cùng 1 kết quả.

```
JS space-operator.js X
JS space-operator.js > ...
1  var a = 7 + 8 - 3 * 4;
2  var b = 7 + 8 - 3 * 4;
3  var c = 7 + 8 - 3 * 4;
4  console.log(a); // 3
5  console.log(b); // 3
6  console.log(c); // 3
```



Biểu thức, toán tử, toán tử ba ngôi

❑ Toán tử ba ngôi (ternary operators):

➤ Cú pháp: (điều kiện) ? (giá trị nếu đúng) : (giá trị nếu sai)

➤ Trong đó:

- **điều kiện**: biểu thức điều kiện của phép toán, sẽ trả về true/false
- **giá trị nếu đúng**: trả về giá trị này nếu kết quả biểu thức là true
- **giá trị nếu sai**: trả về giá trị này nếu kết quả biểu thức là false

➤ Ví dụ: > *// Kiểm tra đậu, rớt của bài kiểm tra (>=5: đậu, <5: rớt)*

```
var avgScore = 9;
```

```
var result = (avgScore >= 5) ? "Đậu" : "Rớt";
```

```
console.log("Kết quả bài kiểm tra là: " + result);
```

```
Kết quả bài kiểm tra là: Đậu
```



Khối lệnh, các lệnh JavaScript căn bản

- ❑ Nhân viên cửa hàng điện thoại phải **đi qua tất cả các khâu để hoàn tất việc thanh toán** khi bạn mua điện thoại.
- ❑ Tương tự, trong code chúng ta thường nhóm các biểu thức với nhau, thường được gọi là block.
- ❑ Trong JavaScript, một **block** được **xác định bằng cách bao một hoặc nhiều lệnh trong một cặp dấu ngoặc nhọn {...}**
- ❑ Ví dụ:

```
JS block.js  X
JS block.js > ...
1  var amount = 99.99;
2  // a general block
3  {
4      amount = amount * 2;
5      console.log(amount); // 199.98
6  }
```




Khối lệnh, các lệnh JavaScript căn bản

- ❑ Kiểu **block** `{..}` chung này **hợp lệ**, nhưng **không thường thấy trong các chương trình JS**.
- ❑ Một **block** **thường được gắn liền với một lệnh điều khiển** (if -> sẽ học sau).
- ❑ Ví dụ:

```
JS block-if.js X
JS block-if.js > ...
1  var amount = 99.99;
2  // is amount big enough?
3  if (amount > 10) { // <-- block attached to `if`
4      amount = amount * 2;
5      console.log(amount); // 199.98
6  }
```

- ❑ Chú ý: Một **block** lệnh **không cần dấu chấm phẩy (;) để kết thúc**.



Khởi lệnh, các lệnh JavaScript căn bản

❑ Có **6 lệnh JavaScript cơ bản** cần biết:

- `console.log(..)`
- `alert(..)`
- `confirm(..)`
- `prompt(..)`
- `document.write(..)`

❑ Ngoài lệnh **`console.log(..)`**, các lệnh khác đều là hàm của đối tượng **`window`** (chỉ được khởi tạo trong **browser**), không hỗ trợ trong **Node.js**

❑ Do vậy, ta sẽ phải sử dụng **DevTools** của **browser** hoặc cài thư viện **`prompt-sync`** cho các bài toán cần **nhập xuất dữ liệu, in dữ liệu ra trình duyệt,...**



Khởi lệnh, các lệnh JavaScript căn bản

❑ Lệnh **console.log(..)**:

➤ Phương thức **console.log(..)** là phương thức **hỗ trợ viết một tin nhắn, đoạn văn bản ra console.**

➤ Cú pháp: **console.log(*message*)**

▪ *message*: Là một tin nhắn hoặc đối tượng được viết ra console.

➤ Ví dụ: in đoạn chữ “**Xin chào FastTracker**” ra màn hình console.

```
JS console-log.js X
JS console-log.js
1 console.log('Xin chào FastTracker');
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL cmd + - ^ X

```
D:\workspace\programming-techniques\lesson-02\lesson-02-slides>node
console-log.js
Xin chào FastTracker
```



Khởi lệnh, các lệnh JavaScript căn bản

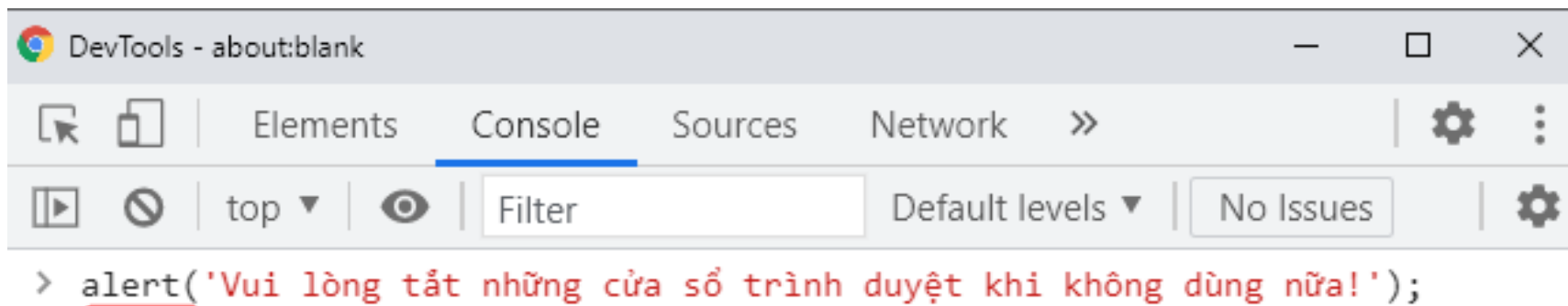
❑ Lệnh **alert(..)**:

➤ Phương thức **alert(..)** là phương thức **hỗ trợ hiển thị một hộp thoại cảnh báo với một tin nhắn và một nút OK** ra trình duyệt hiện tại.

➤ Cú pháp: **alert(*message*)**

▪ *message*: Là đoạn tin nhắn hoặc đối tượng được chuyển đổi từ một chuỗi được hiển thị trong 1 hộp thoại trên trình duyệt.

➤ Ví dụ: hiển thị thông báo với đoạn chữ “**Vui lòng tắt những cửa sổ trình duyệt khi không dùng nữa!**” ra màn hình trình duyệt.

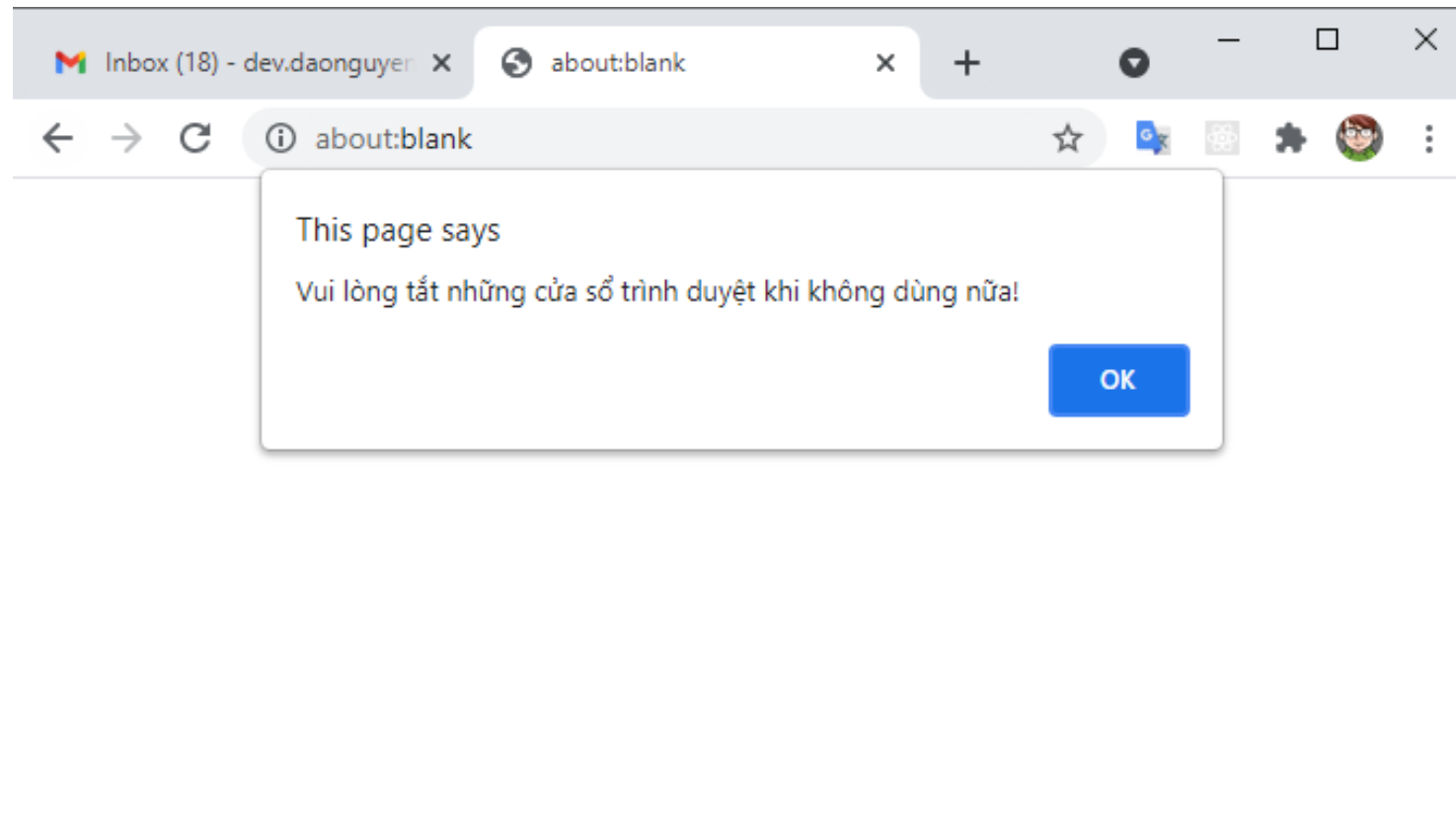




Khởi lệnh, các lệnh JavaScript căn bản

❑ Lệnh **alert(..)**:

➤ Ví dụ: hiển thị thông báo với đoạn chữ “**Vui lòng tắt những cửa sổ trình duyệt khi không dùng nữa!**” ra màn hình trình duyệt.





Khối lệnh, các lệnh JavaScript căn bản

❑ Lệnh **prompt(..)**:

- Phương thức **prompt(..)** là phương thức **hỗ trợ hiển thị một hộp thoại cho phép người dùng nhập dữ liệu vào ô nhập liệu.**
- Khi người dùng nhập liệu xong, người dùng có 2 nút “OK” và “Cancel” mà người dùng có thể nhấn:
 - Nhấn “**Cancel**”: **giá trị trả về** cho phương thức **prompt** này sẽ là **null**.
 - Nhấn “**OK**”: **giá trị trả về** cho phương thức **prompt** này là **dữ liệu người dùng đã nhập**.



Khởi lệnh, các lệnh JavaScript căn bản

❑ Lệnh **prompt(..)**:

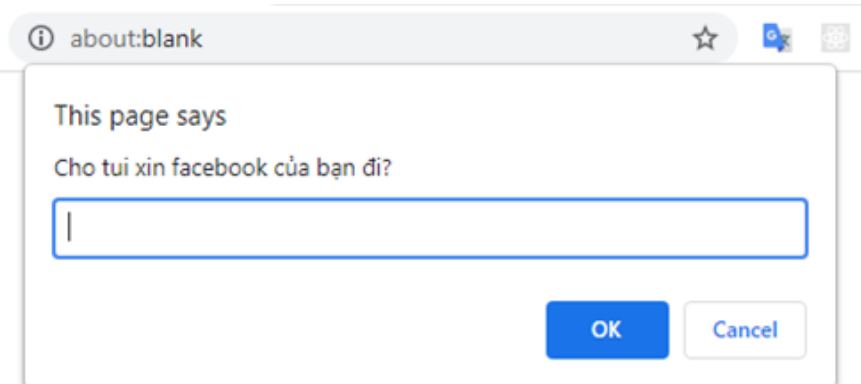
➤ Cú pháp: **prompt**(*text*, *defaultText*)

- *text*: Là **đoạn một chuỗi** được **hiển thị trong 1 hộp thoại** trên trình duyệt (**Bắt buộc có**).

- *defaultText*: Là **giá trị mặc định** cho ô nhập (**Không bắt buộc có**).

➤ Ví dụ: hiển thị câu hỏi sau “**Cho tui xin facebook của bạn đi?**” ra màn hình trình duyệt của một bạn nữ với đoạn mã sau.

```
> var question = "Cho tui xin facebook của bạn đi?";  
   var answer = prompt(question);  
  
   if (answer != null) {  
       console.log('Nick tui nè: ' + answer);  
   } else {  
       console.log('Nhìn mặt thí ghét, không cho!');  
   }  
}
```

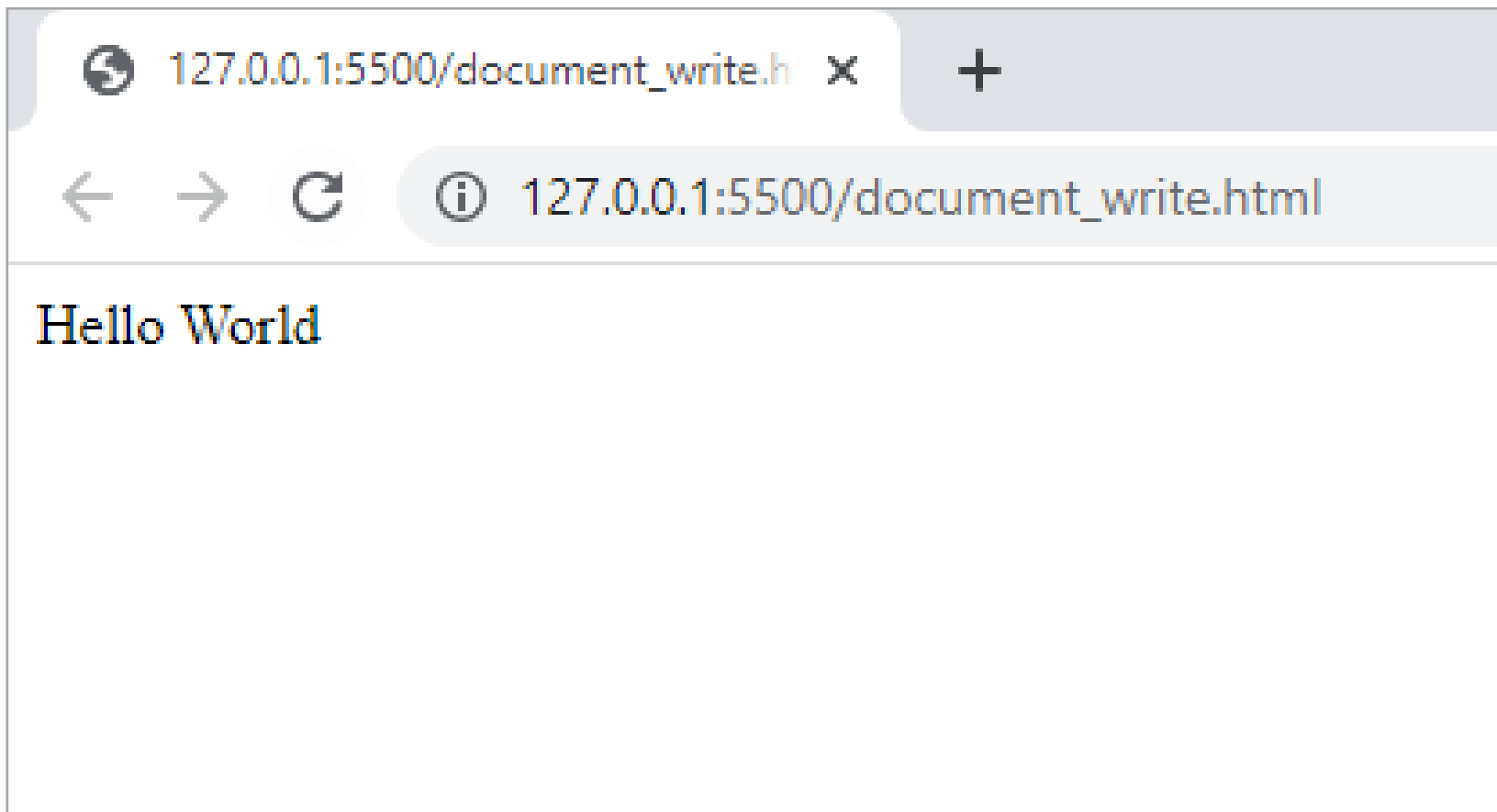




Khởi lệnh, các lệnh JavaScript căn bản

❑ Lệnh **document.write(..)**:

➤ Ví dụ: hiển thị câu chào Hello World ra màn hình trình duyệt.





Tổng kết:

- ☐ Chú thích, các cú pháp căn bản
- ☐ Biến, mục tiêu biến, hằng trong ES6
- ☐ Giá trị biến, các kiểu giá trị nguyên thủy
- ☐ Chuyển đổi giữa các kiểu dữ liệu
- ☐ Biểu thức, toán tử, toán tử điều kiện
- ☐ Khối lệnh, cách lệnh JavaScript căn bản

Let's
Recap

