



VIETNAM
AUSTRALIA
Vocational College

Tổng quan về tư duy lập trình, kỹ thuật lập trình

Mentor: Nguyễn Bá Minh Đạo



Nội dung:

1. Tổng quan về tư duy, kỹ thuật lập trình
2. Tổng quan về ngôn ngữ lập trình JavaScript
3. Môi trường, công cụ lập trình JavaScript
4. Mã lệnh, câu lệnh, thực thi chương trình
5. Tư duy lập trình với lưu đồ thuật toán



Tổng quan về tư duy, kỹ thuật lập trình

❑ Xây dựng **nền tảng, tư duy** về **lập trình**:

- Không quan trọng ngôn ngữ lập trình (rất nhiều, thay đổi liên tục,...)
- Quan trọng về cách phân tích, giải quyết vấn đề.
- Ban đầu nên tập trung học thật vững 1 ngôn ngữ. (rèn tư duy, phân tích)





Tổng quan về tư duy, kỹ thuật lập trình

❑ Rèn luyện các **kỹ thuật, thuật toán** trong **lập trình**:

- Phân tích vấn đề, vẽ lưu đồ thuật toán, lập trình, sửa lỗi,...
- Học các kiến thức lập trình cơ bản bằng JavaScript.
- Rèn luyện các kỹ thuật, thuật toán thường dùng trong lập trình.





Tổng quan về tư duy, kỹ thuật lập trình

❑ **JavaScript** là **ngôn ngữ lập trình vừa**:

➤ Vừa biên dịch, vừa thông dịch

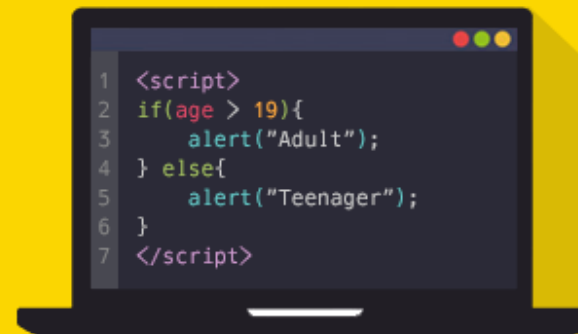
❑ **JavaScript** được **dùng rộng rãi cho các trang web**:

➤ Phía **người dùng** (Front-end): HTML, CSS, JavaScript,...

➤ Phía **máy chủ** (Back-end): JavaScript, NodeJS,...



JavaScript





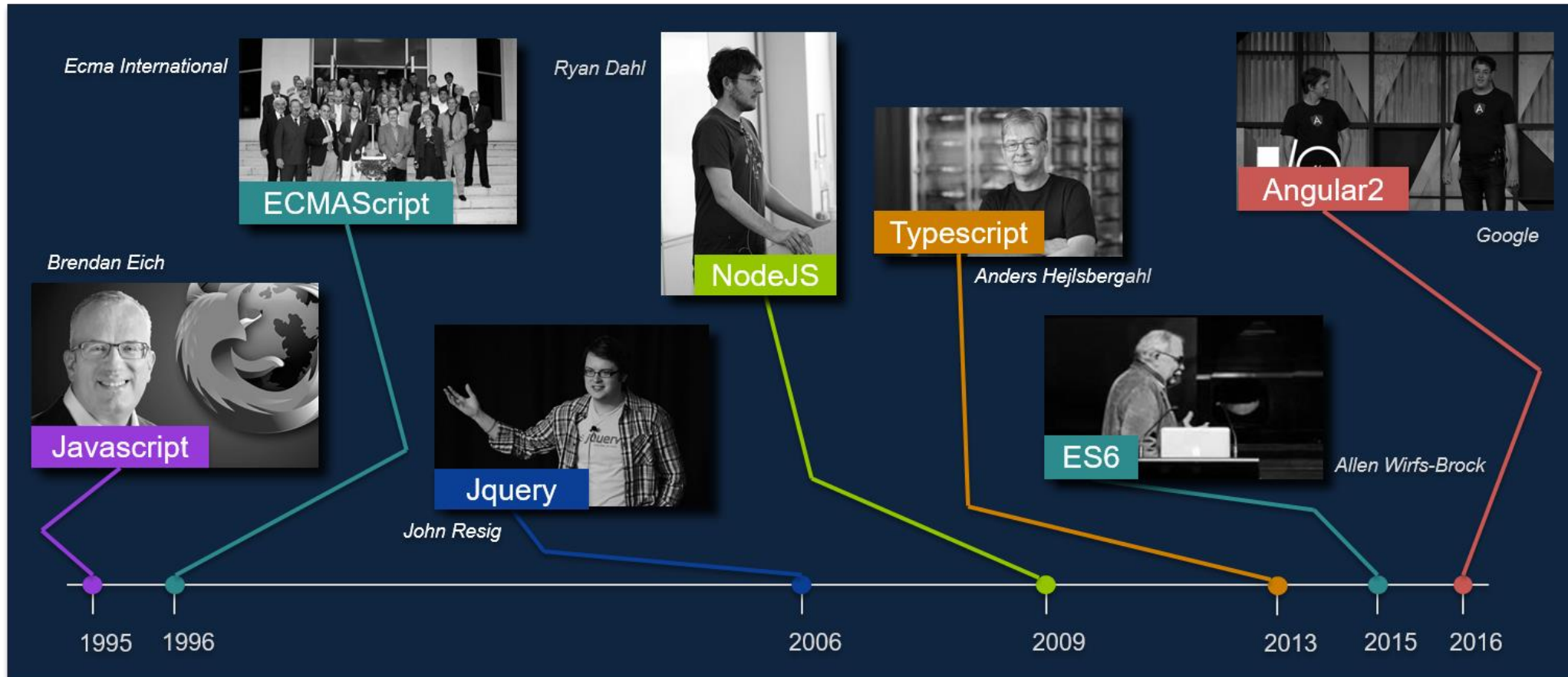
Tổng quan về tư duy, kỹ thuật lập trình

- ❑ **JavaScript có thể dùng chung** với các **language/library/framework** khác:
 - **Front end languages**: HTML, CSS, JavaScript
 - **Front end libraries/frameworks**: Angular.js, React.js, jQuery,...
 - **Back end languages**: JavaScript, Java, PHP, Python, C#,....
 - **Back end frameworks**: Express, Spring, Laravel, Django, .NET,...





Tổng quan về ngôn ngữ lập trình JavaScript



Lịch sử JavaScript






Môi trường, công cụ lập trình JavaScript

- ❑ Tập tin **JavaScript** có **phần mở rộng là .js**.
- ❑ Để lập trình **JavaScript**, ta có thể **3 cách thông dụng** sau:
 - **Cách 1 - Cửa sổ lệnh: Command line (Window) / Terminal (Mac)**
 - Để xài cửa sổ lệnh, ta cần cài **môi trường thực thi mã JavaScript** là **Node.js** qua link: <https://nodejs.org/en/download/>

Downloads

Latest LTS Version: **14.17.0** (includes npm 6.14.13)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS Recommended For Most Users	Current Latest Features	
 Windows Installer node-v14.17.0-x64.msi	 macOS Installer node-v14.17.0.pkg	 Source Code node-v14.17.0.tar.gz



Môi trường, công cụ lập trình JavaScript

- ❑ Tập tin **JavaScript** có **phần mở rộng là .js**.
- ❑ Để lập trình **JavaScript**, ta có thể **3 cách thông dụng** sau:
 - Cài xong, mở cửa sổ lệnh **check phiên bản của node.js** qua lệnh sau: **node --version**

```
Command Prompt
Microsoft Windows [Version 10.0.19042.1052]
(c) Microsoft Corporation. All rights reserved.

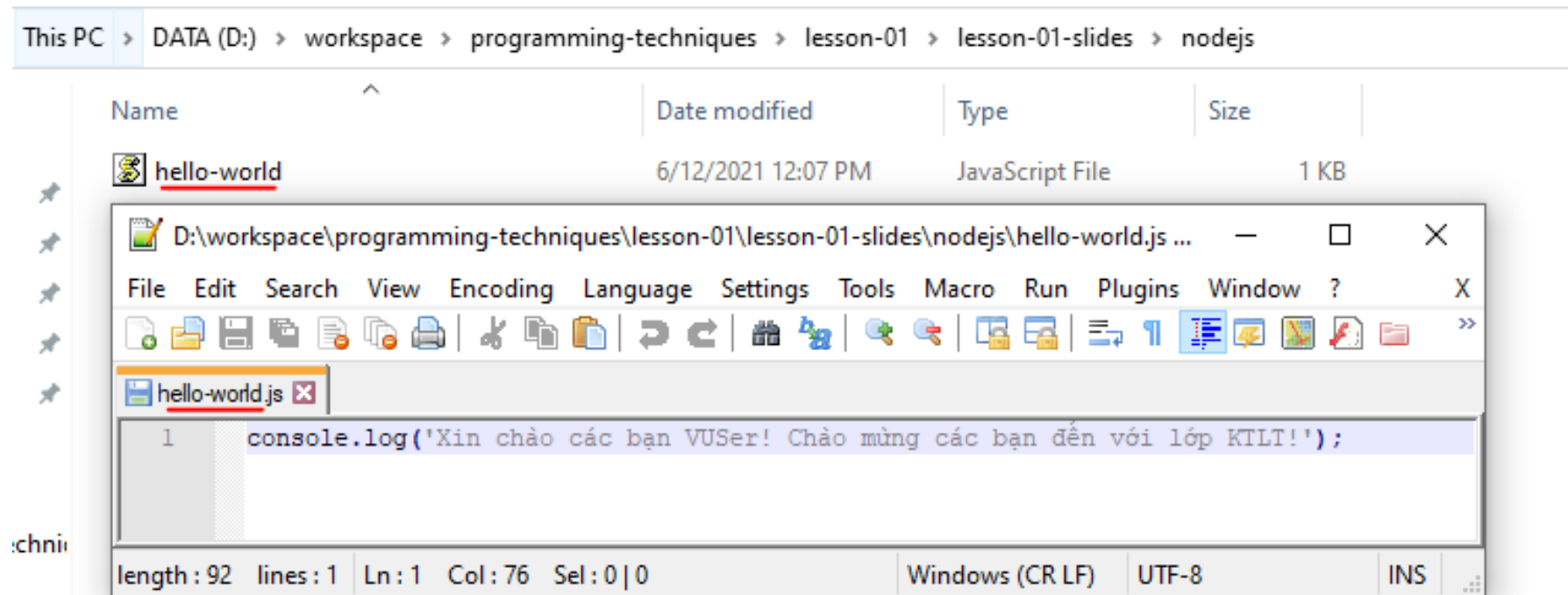
C:\Users\ASUS>node --version
v14.17.0

C:\Users\ASUS>
```



Môi trường, công cụ lập trình JavaScript

- ❑ Tập tin **JavaScript** có **phần mở rộng là .js**.
- ❑ Để lập trình **JavaScript**, ta có thể **3 cách thông dụng** sau:
 - Vào folder chứa bất kỳ, tạo file **hello-world.js** và **gõ lệnh in chữ ra màn hình** như ví dụ sau:





Môi trường, công cụ lập trình JavaScript

- ❑ Tập tin **JavaScript** có **phần mở rộng là .js**.
- ❑ Để lập trình **JavaScript**, ta có thể **3 cách thông dụng** sau:
 - Môi trường thực thi mã JavaScript – **Node.js** đã cài lúc này sẽ **giúp thực thi câu lệnh trong file hello-world.js** và **xuất kết quả** như sau:

```
C:\Windows\System32\cmd.exe

D:\workspace\programming-techniques\lesson-01\lesson-01-slides\
nodejs>node hello-world.js
Xin chào các bạn VUser! Chào mừng các bạn đến với lớp KTLT!

D:\workspace\programming-techniques\lesson-01\lesson-01-slides\
nodejs>
```



Môi trường, công cụ lập trình JavaScript

- ❑ Tập tin **JavaScript** có **phần mở rộng là .js**.
- ❑ Để lập trình **JavaScript**, ta có thể **3 cách thông dụng** sau:
 - **Cách 2 - Sử dụng Developer Console:**
 - Developer Console: **Cửa sổ lệnh của trình duyệt** (Browser) cũng là một môi trường có thể thực thi các mã lệnh JavaScript.
 - Một số **trình duyệt phổ biến** ta thường thấy: Chrome, FireFox, Microsoft Edge, Safari,...

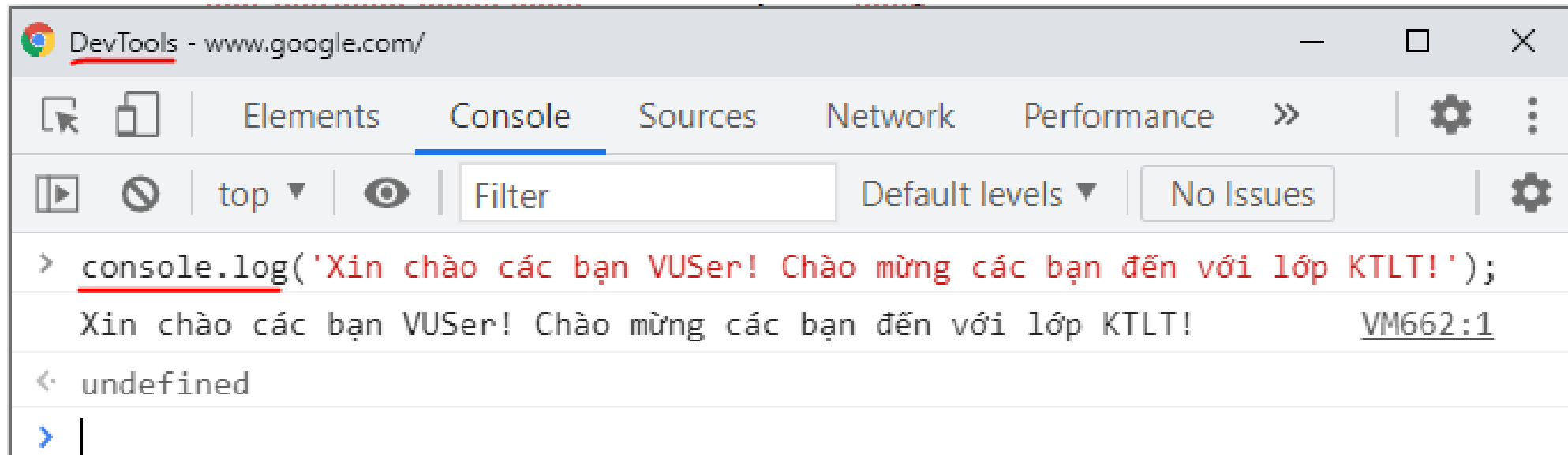


- Nhớ bật tính năng **Developer Tools** trong **Settings** lên (nếu chưa bật).



Môi trường, công cụ lập trình JavaScript

- ❑ Tập tin **JavaScript** có **phần mở rộng là .js**.
- ❑ Để lập trình **JavaScript**, ta có thể **3 cách thông dụng** sau:
 - Chuột phải vào trình duyệt bất kỳ -> Chọn **Inspect** (kiểm tra) hoặc nhấn **tổ hợp phím Ctrl + Shift + I** để mở **DevTools** (Developer Tools) -> chọn tab **Console** để có thể viết mã JavaScript ở đây.





Môi trường, công cụ lập trình JavaScript

- ❑ Tập tin **JavaScript** có **phần mở rộng là .js**.
- ❑ Để lập trình **JavaScript**, ta có thể **3 cách thông dụng** sau:
 - Để **gõ nhiều dòng trên console cùng lúc**, ta sử dụng **<shift> + <enter>** để chuyển sang dòng mới. Khi nhấn **<enter>**, console **sẽ chạy tất cả những gì bạn vừa viết**.

The screenshot shows the Chrome DevTools Console interface. The 'Console' tab is selected. The input area contains the following code:

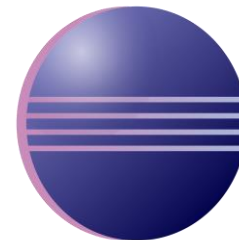
```
> a = 18;  
  b = a * 2;  
  console.log(b);
```

The output area shows the result of the last line of code: 36, with the source location VM289:3 on the right. Below the output, the prompt '< undefined' is visible, and the input area is ready for the next command with a blue prompt character '>' and a cursor.



Môi trường, công cụ lập trình JavaScript

- ❑ Tập tin **JavaScript** có **phần mở rộng là .js**.
- ❑ Để lập trình **JavaScript**, ta có thể **3 cách thông dụng** sau:
 - **Cách 3 - Sử dụng IDE:**
 - IDE (**Integrated Development Environment**): môi trường tích hợp dùng để viết code **phát triển** ứng dụng.
 - IDE **tích hợp** các **tool hỗ trợ** việc **lập trình**: trình biên dịch (Compiler), trình thông dịch (Interpreter), kiểm tra lỗi (Debugger), định dạng hoặc nổi bật code, tổ chức thư mục code, tìm kiếm code,...





Môi trường, công cụ lập trình JavaScript

❑ Sử dụng IDE **Visual Studio Code**:

➤ Vào folder chứa bất kỳ, tạo file **hello-world.js**, dẫn đường dẫn tới tool **VSCo** và gõ lệnh in chữ ra màn hình như ví dụ sau:

The screenshot shows the Visual Studio Code interface. The editor window displays a file named `hello-world.js` with the following code:

```
1 console.log('Xin chào các bạn VUSer! Chào mừng các bạn đến với lớp KTLT!');
```

The bottom panel shows the **TERMINAL** tab. The terminal output is as follows:

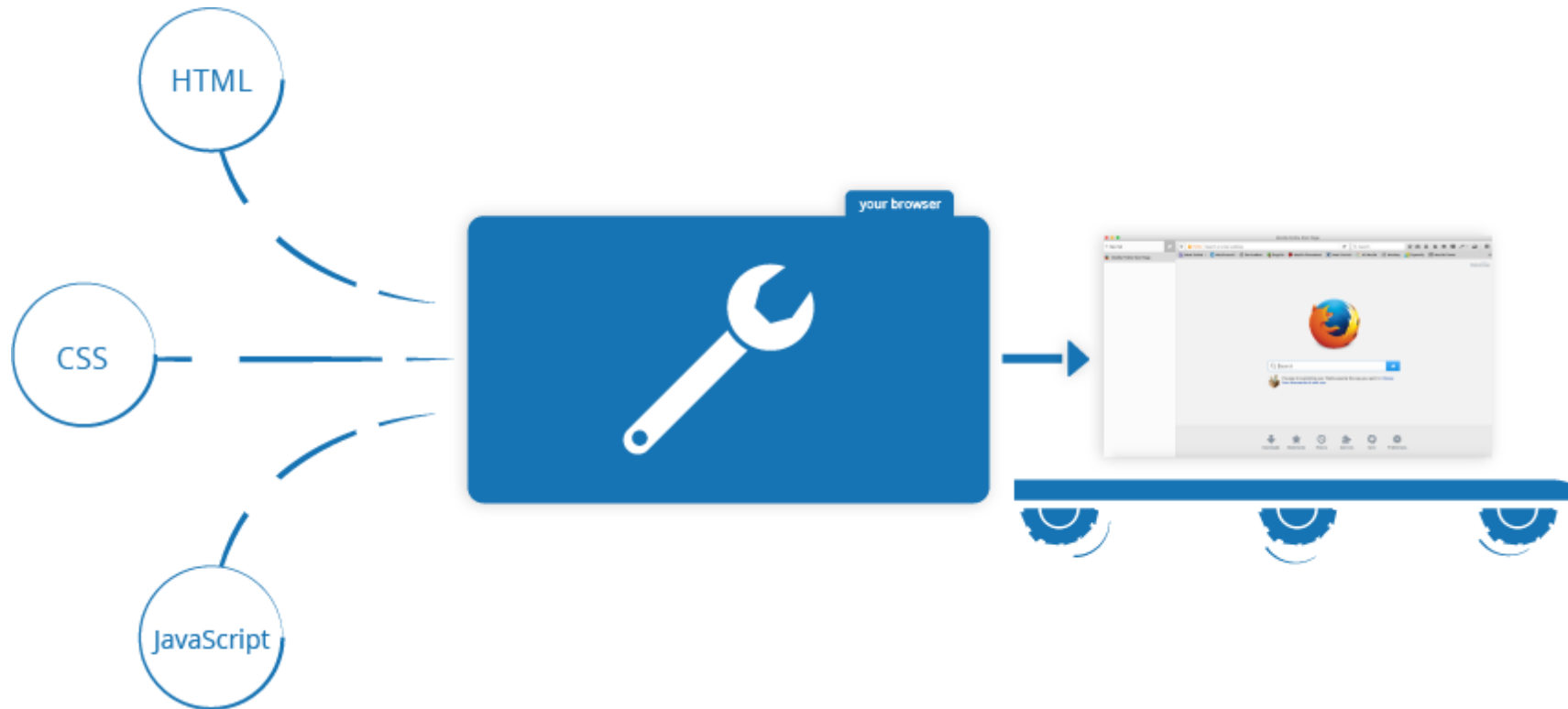
```
Microsoft Windows [Version 10.0.19042.1052]  
(c) Microsoft Corporation. All rights reserved.  
  
D:\workspace\programming-techniques\lesson-01\lesson-01-slides\vscode>node hello-world.js  
Xin chào các bạn VUSer! Chào mừng các bạn đến với lớp KTLT!  
  
D:\workspace\programming-techniques\lesson-01\lesson-01-slides\vscode>
```

The status bar at the bottom indicates the current line and column: **Ln 1, Col 76**. Other status information includes **Spaces: 4**, **UTF-8**, **CRLF**, and **JavaScript**.



Môi trường, công cụ lập trình JavaScript

- ❑ Tập tin **JavaScript** có **phần mở rộng là .js**.
- ❑ Khi dùng VSCode để code **JavaScript**, ta có thể **2 cách thông dụng** sau:
VSCode (HTML + JS + DevTools), **VSCode (JS + Terminal + Node.js/npm)**.





Mã lệnh, câu lệnh, thực thi chương trình

❑ Mã lệnh (**Code**) là gì?

➤ Một chương trình, thường được gọi là **source code/code**, là **một tập hợp các hướng dẫn** để **yêu cầu máy tính cần xử lý một nhiệm vụ nào đó**.

➤ Thông thường, **code được lưu dưới dạng file văn bản**, với js bạn có thể gõ code trực tiếp lên console của trình duyệt.

```
> console.log('Xin chào các bạn VUSer! Chào mừng các bạn đến với lớp KTLT!');  
Xin chào các bạn VUSer! Chào mừng các bạn đến với lớp KTLT! VM662:1  
< undefined  
> |
```



Mã lệnh, câu lệnh, thực thi chương trình

❑ Các **câu lệnh (Statement)** là gì?

➤ Trong ngôn ngữ máy tính, **một nhóm từ/số/cách thức thực thi một nhiệm vụ cụ thể** nào đó được **gọi là câu lệnh**.

➤ Ví dụ: **a = b * 2;**

- Ký tự **a** và **b** là các biến (**variable**)
- Số **2** chỉ là một giá trị -> giá trị ký tự (**literal value**)
- Dấu **=** và ***** là toán tử (**operators**) -> thực thi các hành động với giá trị và biến như sự phân công và phép toán nhân.
- Hầu hết các câu lệnh JS kết thúc bằng **dấu chấm phẩy (;)** ở cuối câu.



Mã lệnh, câu lệnh, thực thi chương trình

❑ Các **câu lệnh (Statement)** là gì?

➤ Ví dụ: **`a = b * 2;`**

- Lệnh **`a = b * 2;`** báo cho máy tính giá trị được lưu trữ trong biến **b**, nhân giá trị đó với **2**, sau đó lưu kết quả vào 1 biến khác gọi là **a**.

➤ **Lập trình** tương tự như **bộ sưu tập của nhiều câu lệnh cùng nhau mô tả tất cả các bước để thực thi mục đích lập trình.**

```
JS statement.js X
JS statement.js
1  a = b * 2;
```



Mã lệnh, câu lệnh, thực thi chương trình

□ Biểu thức (**Expressions**) là gì?

➤ **Các câu lệnh được tạo thành từ** một hay nhiều **biểu thức**. Một biểu thức là **bất kỳ tham chiếu trên một biến hoặc giá trị, hoặc tập hợp các giá trị và các biến kết hợp thành toán tử**.

➤ Ví dụ: $a = b * 2$;

- Lệnh này có **4 biểu thức bên trong** nó:

- **2** là **giá trị biểu thức trực kiện** (không chứa trong biến nào cả).

- **b** là **giá trị biểu thức** (nghĩa là sẽ lấy giá trị hiện tại của biến b).

- $b * 2$ là **biểu thức toán học** (nghĩa là sẽ làm phép nhân).

- $a = b * 2$ là **biểu thức gán** (nghĩa là sẽ gán kết quả của biểu thức $b * 2$ cho biến a).



Mã lệnh, câu lệnh, thực thi chương trình

❑ Biểu thức (**Expressions**) là gì?

➤ Ngoài ra, một **biểu thức** **chung** **đứng** **một** **mình** còn được gọi là **lệnh biểu thức** (expression statement). Ví dụ: **b * 2;**

➤ **Lệnh biểu thức** **thường** **không** **hữu** **dụng**, bởi nó chẳng có tác dụng đối với chương trình đang chạy (Lấy giá trị b nhân với 2, xong không làm gì cả).

➤ Một loại câu lệnh biểu thức nữa là **biểu thức lệnh gọi** (call expression), khi **toàn** **câu** **lệnh** là **một** **hàm** **tự** **gọi** **biểu** **thức**. Ví dụ: **alert(a);**

```
JS expression.js X
JS expression.js
1 // Nhiều biểu thức (expressions)
2 a = b * 2;
3 // Lệnh biểu thức (Expression Statement)
4 b * 2;
5 // Biểu thức lệnh gọi (Call Expression)
6 alert(a);
```



Mã lệnh, câu lệnh, thực thi chương trình

❑ Thực thi chương trình là gì?

➤ Làm cách nào mà tập hợp các câu lệnh lập trình có thể yêu cầu máy tính phải làm gì? **Chương trình cần được thực thi**, hay còn được biết đến với khái niệm đó là **chạy chương trình**.



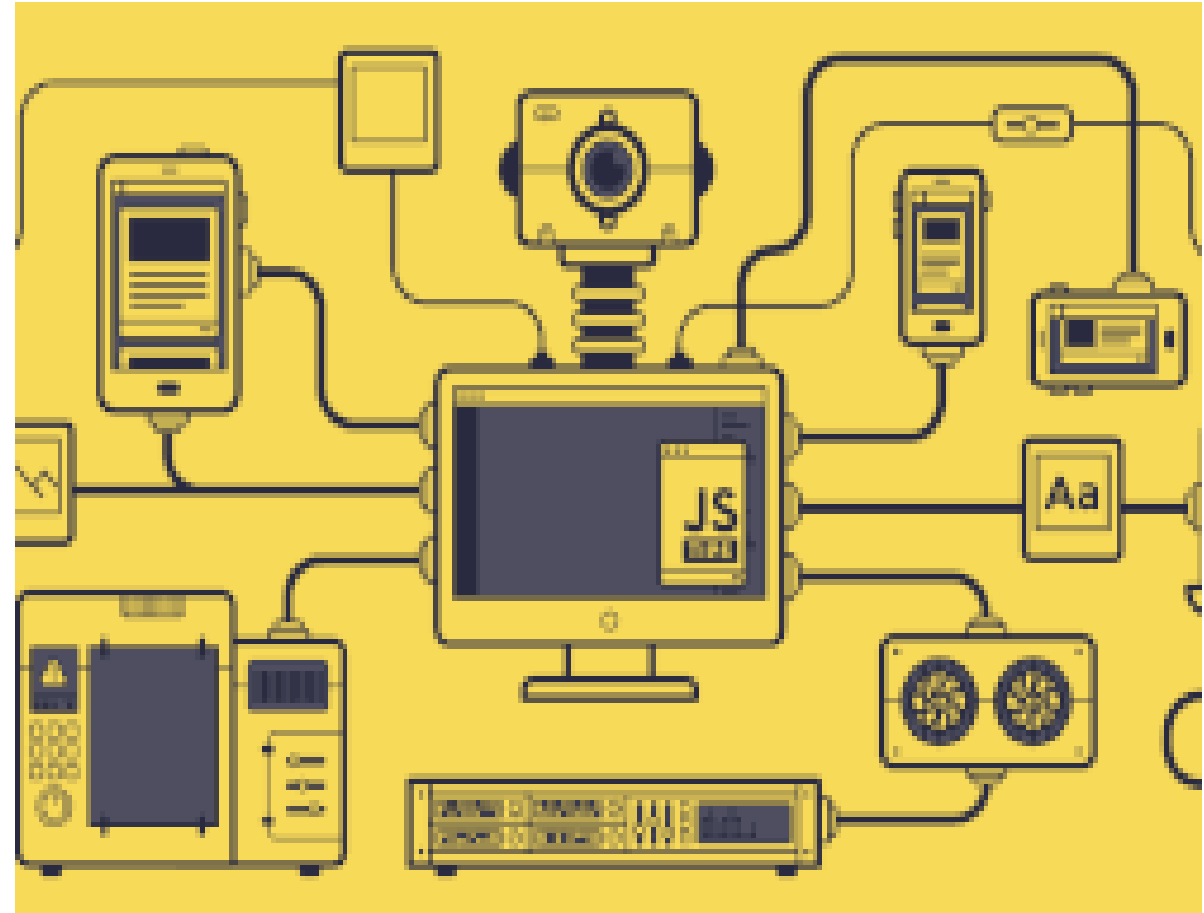


Mã lệnh, câu lệnh, thực thi chương trình

❑ **Thực thi chương trình là gì?**

➤ Các lệnh giống như $a = b * 2$; rất **đễ cho lập trình viên đọc** và **viết**, nhưng nó không hoàn toàn ở dạng để máy tính hiểu trực tiếp.

➤ Một **trình tiện ích** đặc biệt trong máy tính (vừa là **thông dịch - interpreter** vừa là **biên dịch - compiler**) được sử dụng để dịch code bạn viết thành các lệnh mà máy tính có thể hiểu được.





Mã lệnh, câu lệnh, thực thi chương trình

❑ Thực thi chương trình là gì?

- Đối với một số máy tính, mỗi khi chương trình chạy thì bản dịch của **các câu lệnh** thường **được hoàn thành từ trên xuống dưới, từng dòng một**, nó thường được gọi là **thông dịch mã**.
- Một số ngôn ngữ khác, **bản dịch được hoàn thiện trước**, gọi là biên dịch mã, rồi sau đó chương trình mới chạy những gì đã được biên dịch xong, được gọi là **biên dịch mã**.





Mã lệnh, câu lệnh, thực thi chương trình

❑ **Thực thi chương trình là gì?**

- **JavaScript** thường **được khẳng định là ngôn ngữ thông dịch**, bởi vì mã nguồn JavaScript được xử lý mỗi lần chạy.
- Điều trên không hoàn toàn chính xác, thực ra là **JavaScript engine biên dịch chương trình** và **sau đó chạy ngay mã đã được biên dịch**.
- **Kết luận:** JavaScript vừa là ngôn ngữ thông dịch và cũng là ngôn ngữ biên dịch.

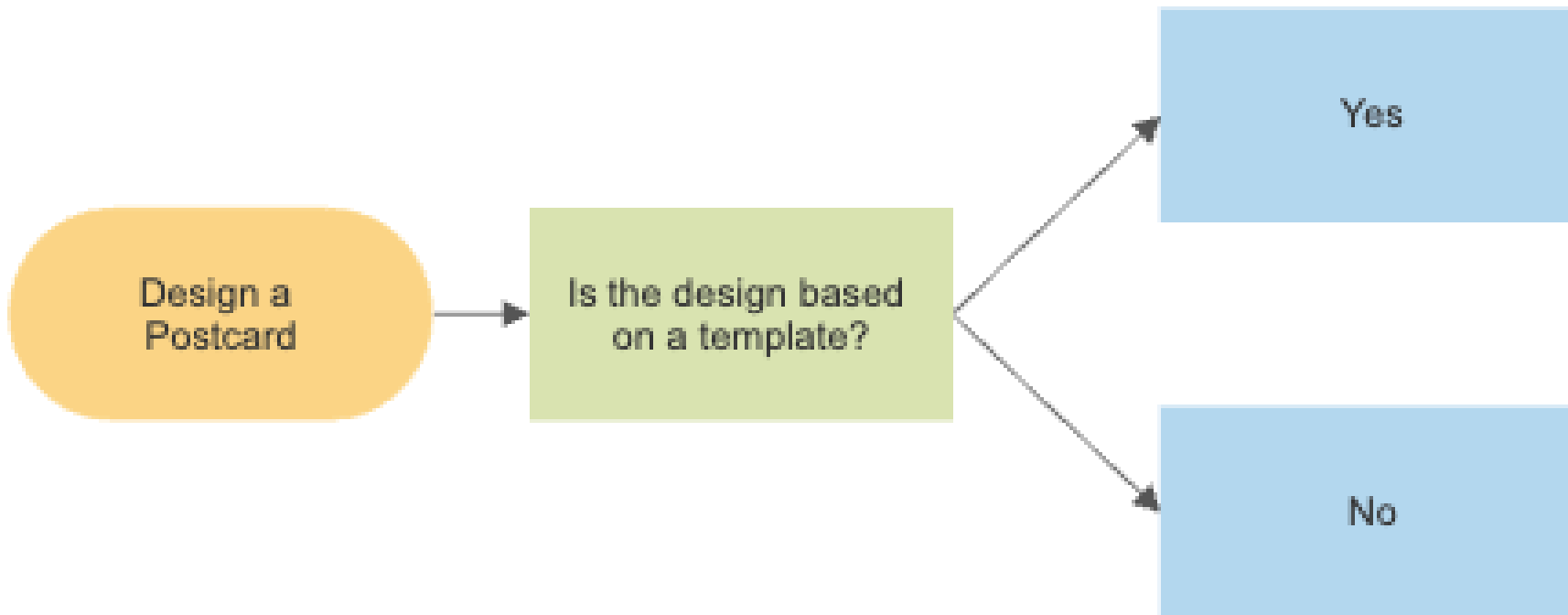




Tư duy lập trình với lưu đồ thuật toán

□ Định nghĩa:

➤ **Lưu đồ thuật toán** (flowchart) là **công cụ dùng để biểu diễn thuật toán, mô tả dữ liệu nhập** (input), **dữ liệu xuất** (output) và **luồng xử lý thuật toán** thông qua các ký hiệu hình học.

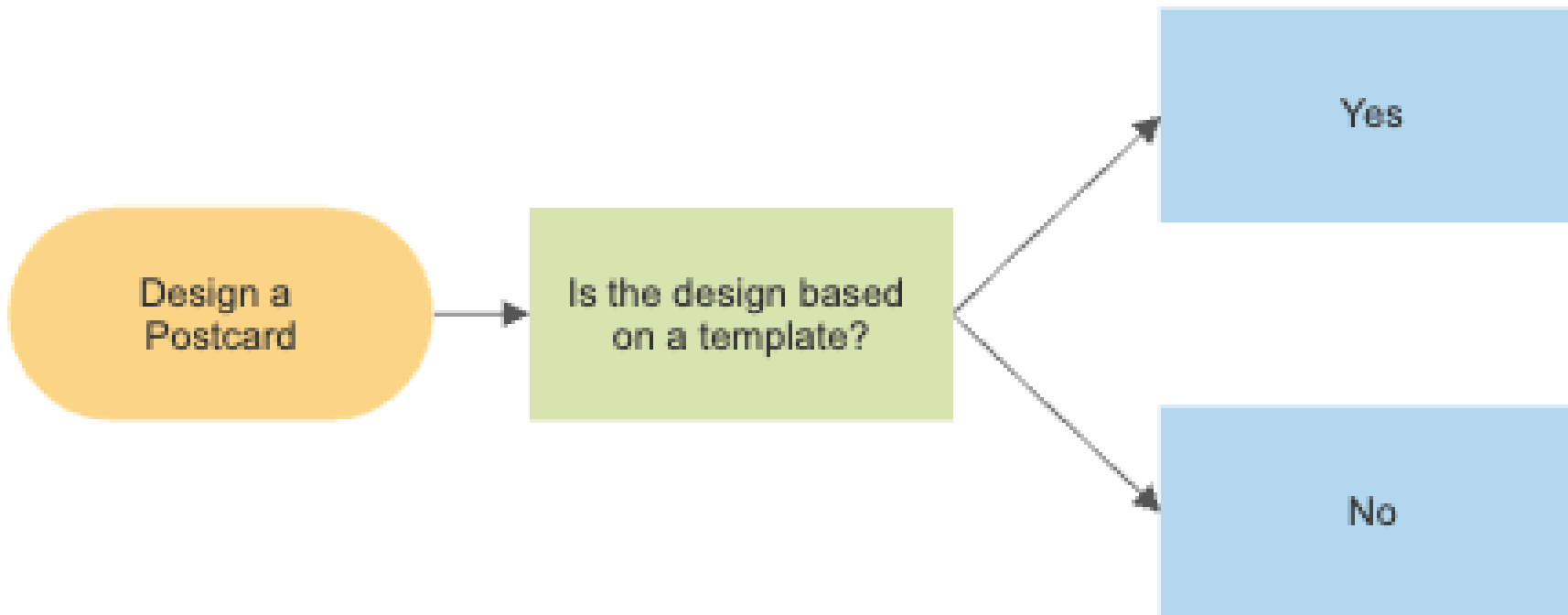




Tư duy lập trình với lưu đồ thuật toán

□ Công dụng:

➤ **Giúp kiểm tra tính khả thi** của **việc lập trình**, **đưa ra** những **giải thuật** để **viết chương trình** một cách **nhANH chóng**, **hiệu quả** qua các bước trong lưu đồ được thực hiện theo một trình tự đơn giản.





Tư duy lập trình với lưu đồ thuật toán

□ Các ký hiệu:

➤ Để vẽ lưu đồ thuật toán, bạn cần nhớ và **tuân thủ các ký hiệu sau**:

STT	Ký hiệu	Ý nghĩa
1		Bắt đầu / Kết thúc chương trình
2		Điều kiện rẽ nhánh (lựa chọn)
3		Luồng xử lý
4		Nhập



Tư duy lập trình với lưu đồ thuật toán

□ Các ký hiệu:

➤ Để vẽ lưu đồ thuật toán, bạn cần nhớ và **tuân thủ các ký hiệu sau**:

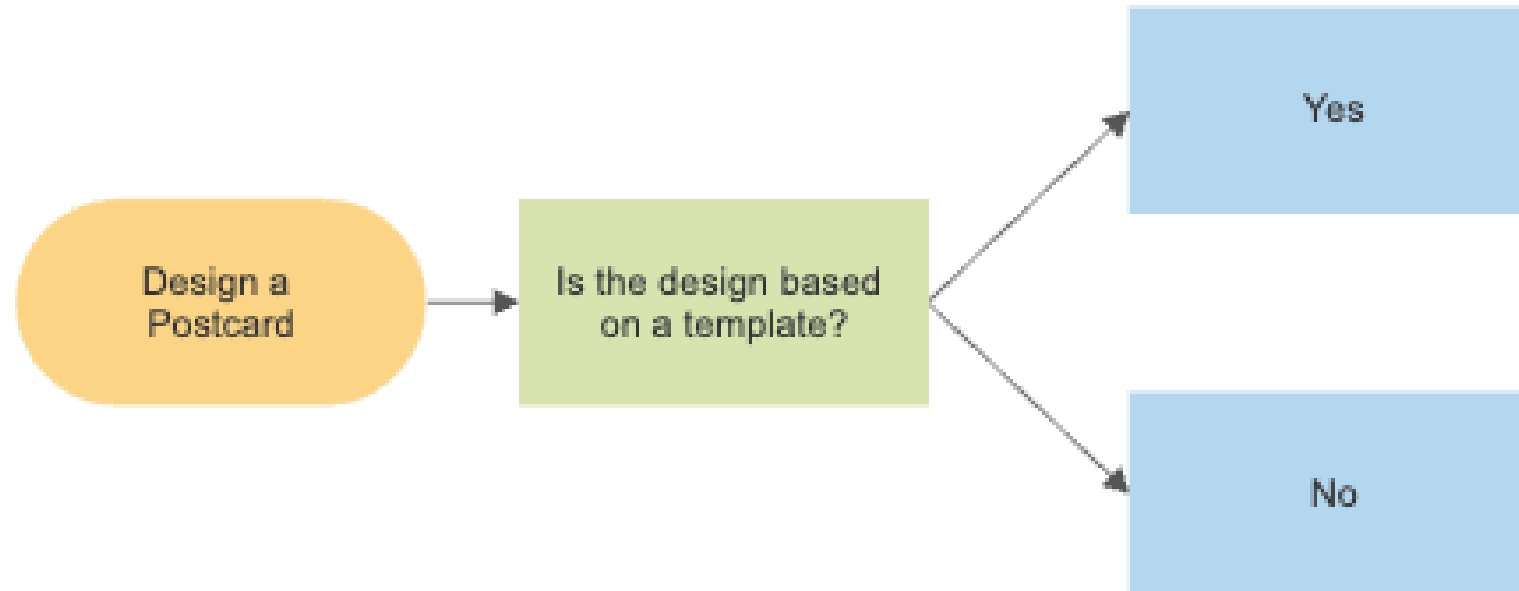
5		Xuất
6		Xử lý / tính toán / gán
7		Trả về giá trị (return)
8		Điểm nối liên kết



Tư duy lập trình với lưu đồ thuật toán

□ Trình tự:

- Lưu đồ thuật toán được duyệt theo trình tự:
 - Duyệt từ **trên xuống dưới**.
 - Duyệt từ **trái sang phải**.

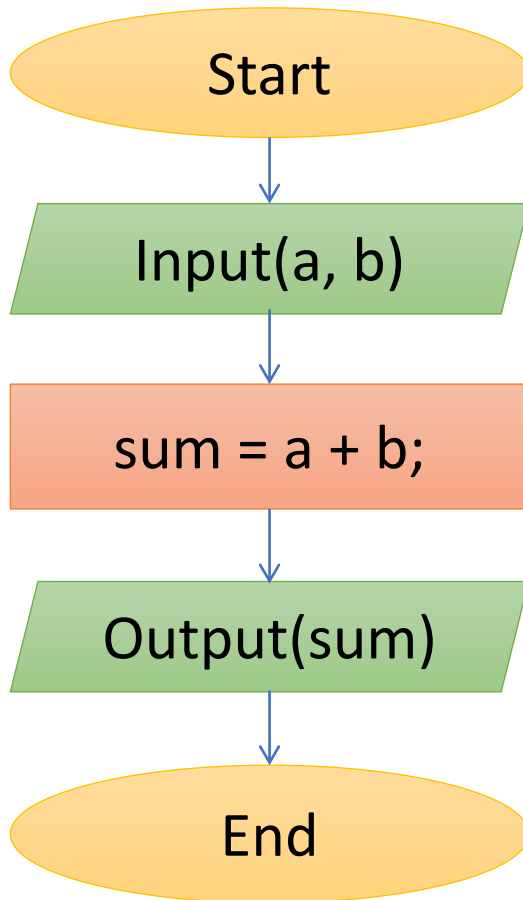




Tư duy lập trình với lưu đồ thuật toán

❑ Ví dụ:

- Vẽ lưu đồ thuật toán nhập 2 số a và b. Tính tổng 2 số và xuất ra kết quả.



```
DevTools - about:blank
```

Elements Console Sources Network

top Filter Default

```
> var a = prompt("Mời bạn nhập số hạng a: ");  
var b = prompt("Mời bạn nhập số hạng b: ");  
var sum = Number(a) + Number(b);  
console.log("Tổng của 2 số a và b là: " + sum);
```

Tổng của 2 số a và b là: 11



Tổng kết:

- ☐ Tổng quan về tư duy, kỹ thuật lập trình
- ☐ Tổng quan về ngôn ngữ lập trình JavaScript
- ☐ Môi trường, công cụ lập trình JavaScript
- ☐ Chú thích, các cú pháp căn bản
- ☐ Tư duy lập trình với lưu đồ thuật toán

Let's
Recap

