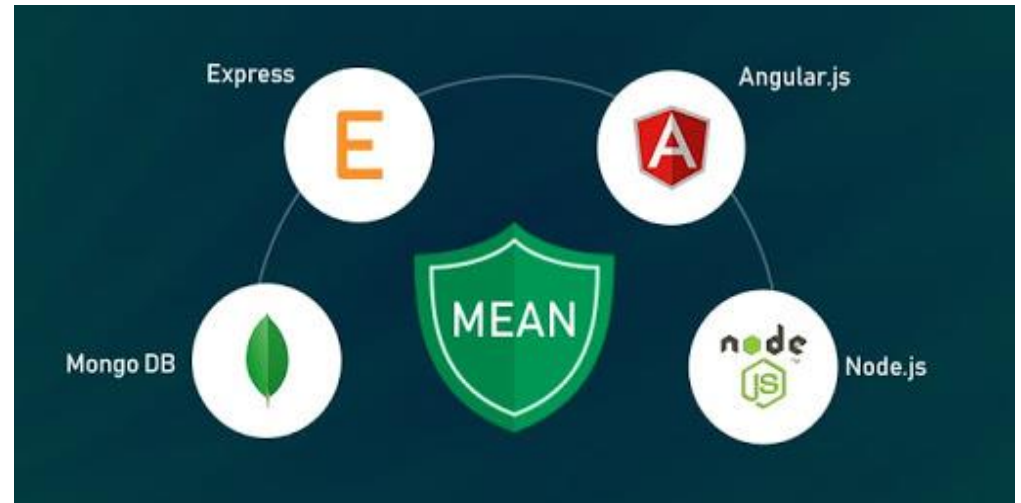


JAVASCRIPT



Giảng Viên: ThS. Giang Hào Côn

1.1/ Javascript là gì ?

- **Javascript**, nếu phân tách thuật ngữ này, chúng ta sẽ có **java** + **script**. Bản thân từ script mang ý nghĩa ám chỉ "kịch bản". Chính vì vậy Javascript là một loại ngôn ngữ kịch bản, được xử lý bởi trình duyệt web hay máy chủ (server), giúp một website hay phần mềm hoạt động theo kịch bản được soạn bởi lập trình viên, cũng như tham gia vào quá trình xử lý các tương tác giữa người dùng và website/phần mềm.
- **Javascript** (viết tắt là **JS**) là một ngôn ngữ lập trình kịch bản phía máy khách, mã lệnh được thực thi bởi trình duyệt của người dùng.

1.1/ Javascript là gì ?

- Đến nay, Javascript được chia thành 2 nhánh chính là **Client-Side Javascript** và **Server-side Javascript**. Các đoạn code Javascript thường được lưu vào tập tin có phần mở rộng là **.js** hoặc **.min.js** (Ví dụ: **my-scripts.js**).


Ví dụ 01: Chúng ta đang viết một trang web về cách nấu một món ăn, và muốn đưa vào một kịch bản chương trình rằng, nếu người xem trượt đến cuối nội dung của trang đó, một hộp thoại chứa Form (mẫu điền thông tin) sẽ hiện ra dưới dạng Popup để người dùng điền vào và gửi về cho chúng ta. Chính vì thế, chúng ta cần đến Javascript để biên soạn kịch bản dưới dạng ý tưởng thành một kịch bản mà phần mềm máy tính có thể hiểu và xử lý.

1.1/ Javascript là gì ?

- Ví dụ 02: Kiểm tra dữ liệu nhập vào Form.

CẬP NHẬT THÔNG TIN CÁ NHÂN

Họ tên:

Ngày sinh: 

Giới tính: ☒ Nam ☐ Nữ

THÔNG TIN CÁ NHÂN

Họ tên:

Ngày sinh:

Giới tính:

1.1/ Javascript là gì ?

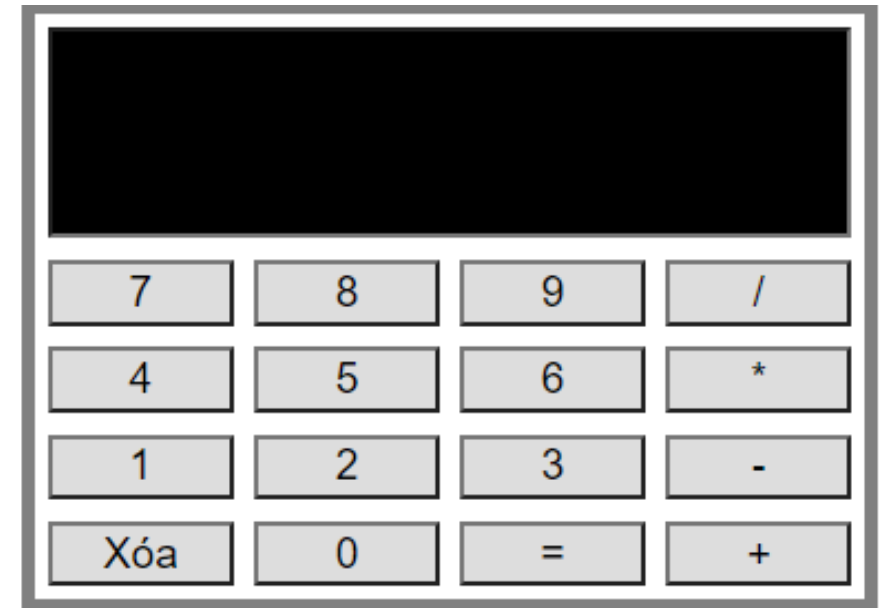
- Ví dụ 03: Xây dựng đồng hồ thời gian thực.

15:18:39

- Ví dụ 04: Hiện/ẩn nội dung trên trang web.



- Ví dụ 05: Máy tính đơn giản.



1.1.1/ Client-side Javascript là gì ?

- **Client-side Javascript** là các đoạn mã Javascript chạy trực tiếp trên website hay phần mềm trên thiết bị của người dùng. Trong lập trình website, các đoạn mã Client-side Javascript sẽ được đọc và xử lý bằng trình duyệt web (giống như HTML & CSS). Do **Client-side Javascript không thể hoạt động độc lập**, mà cần phải thông qua một trang HTML, do đó, có thể xem Client-side Javascript là một tính năng bổ sung cho HTML.
- So với **Server-side Javascript** thì **Client-side Javascript** được sử dụng phổ biến hơn nhiều, và trong nhiều trường hợp, ta có thể ngầm hiểu rằng Javascript = Client-side Javascript. Và tất nhiên, trong bài học Javascript căn bản, ta chỉ tập trung tìm hiểu về **Client-side Javascript**.

1.1.1/ Client-side Javascript là gì ?

Những ứng dụng phổ biến của Client-side Javascript

- Là cơ sở để người lập trình có thể xây dựng nên những website có khả năng tương tác với người dùng
- Giúp người lập trình tạo ra những hiệu ứng đẹp mắt cho website
- Cung cấp thêm giải pháp giúp website có khả năng responsive
- Giúp website có khả năng gửi và nhận dữ liệu mà không cần load lại trang
- Giúp nhà phát triển có thể kiểm soát hoạt động của người dùng trên website.

1.1.2/ Server-side Javascript là gì ?

Mãi đến 27/05/2009 khi NodeJS xuất hiện, khái niệm Server-side Javascript mới được ra đời. Server-side Javascript là các đoạn mã Javascript được xử lý bằng các máy chủ (thay vì xử lý trên phần mềm máy tính của người dùng), và được sử dụng để xây dựng các hệ thống back-end, tương tác với các cơ sở dữ liệu cho các website, phần mềm. Cũng sau thời điểm NodeJS được giới thiệu đến cộng đồng lập trình, cùng với mongoDB, ngày càng có nhiều dự án sử dụng Javascript để xây dựng các hệ thống back-end, thay cho PHP, ASP.net....

1.2./ Kịch bản Javascript là gì ?

Kịch bản phim	Kịch bản Javascript
Câu chuyện của chúng ta có 2 nhân vật: Nam và Kim . (Diễn biến 1)	<code>var nam, kim;</code> <code>// Statement 1</code>
Nhà Nam rất nghèo nên tài khoản ngân hàng của Nam hiện giờ chỉ có 1 triệu . (Diễn biến 2)	<code>nam = 1000000;</code> <code>// Statement 2</code>
Nhà Kim rất giàu, bố làm chủ tịch nên tài khoản ngân hàng của Kim hiện giờ có đến 100 triệu . (Diễn biến 3)	<code>kim = 100000000;</code> <code>// Statement 3</code>
Nam đã yêu thầm Kim từ lâu nên ngỏ lời cầu hôn, Kim trả lời rằng nếu tài khoản của Nam có nhiều tiền hơn tài khoản của Kim thì Kim sẽ đồng ý. (Diễn biến 4)	<code>if(nam > kim){</code> <code>alert ('Kim đồng ý lời cầu hôn của Nam');</code> <code>}else{</code> <code>alert('Kim từ chối lời cầu hôn của Nam');</code> <code>}</code> <code>// Statement 4</code>

1.2./ Kịch bản Javascript là gì ?

Như vậy, **statement** là thành tố cơ bản cấu tạo nên 1 kịch bản Javascript, cũng giống như **HTML Element** là thành tố cơ bản cấu tạo nên 1 văn bản HTML, hay **CSS Rule** là thành tố cơ bản cấu tạo nên 1 đoạn mã CSS. Một kịch bản Javascript có thể có 1 hoặc nhiều statement. Các statement có thể được viết trên những hàng riêng biệt hoặc trên cùng một hàng.

Ví dụ: 3 statement được viết trên cùng một hàng:

```
var nam, kim; nam = 1000000; kim = 1000000000;
```

1.2./ Kịch bản Javascript là gì ?

Cấu trúc của 1 Javascript statement

Không giống như HTML hay CSS, các **thành phần cấu tạo nên một statement trong Javascript không bị ràng buộc về chủng loại, số lượng và thứ tự**. Điều đó có nghĩa rằng sẽ có những statement rất dài với vô số các thành phần và những statement rất ngắn với chỉ 1 vài thành phần, tùy thuộc vào mục đích của người viết ra nó. **Điểm chung duy nhất** của các statement trong một kịch bản Javascript là **kết thúc bằng dấu ; (chấm phẩy)**.

1.2./ Kịch bản Javascript là gì ?

Cấu trúc của 1 Javascript statement

Ví dụ 1 statement ngắn:

```
var a1;
```

Ví dụ 1 statement dài:

```
var b2 = Math.round(20/3) + Math.abs(-50);
```

Trong một số trường hợp, một statement có thể chứa nhiều "statement con" bên trong:

```
if(a < 5){  
    result = true;  
}else{  
    result = false;  
}
```

Các thành phần thường xuất hiện trong 1 statement là: **Biến (Variable)**, **Hằng (Constant)**, **Dữ liệu (Data, Operator & Function)**.

1.3./ Trình tự xử lý của 1 kịch bản Javascript

- Một kịch bản Javascript sẽ được xử lý theo thứ tự từ trên xuống dưới, từ trái qua phải. Điều đó có nghĩa là các statement viết ở trên cùng sẽ được ưu tiên xử lý trước. Nếu một hàng có 2 hoặc nhiều statement thì các statement bên trái sẽ được xử lý trước.
- Nếu trường hợp 1 statement bị lỗi trong quá trình xử lý (lỗi code Javascript), quá trình xử lý sẽ bị dừng lại và các statement phía sau/dưới sẽ không được xử lý.

1.4./ Comment trong Javascript

Giống như HTML & CSS, cú pháp của Javascript cũng cho phép chúng ta chèn vào các đoạn comment (ghi chú) trong quá trình viết code. Các đoạn comment này sẽ không tham gia vào quá trình xử lý (chạy code).

Cách để viết comment trong Javascript: **// Nội dung comment**

```
console.log('Hello world'); // Comment 1  
alert('Hi everyone!');  
// Comment 2
```

Lưu ý: Comment trong Javascript không có ký tự đánh dấu kết thúc. Nếu bạn Enter xuống dòng code tiếp theo nghĩa là đã chấm dứt dòng comment trước đó.

1.5./ Phương thức hoạt động Javascript

- Đối với **Client-side Javascript**, các đoạn mã Javascript không thể hoạt động độc lập mà cần thông qua một trang HTML. Nghĩa là nếu chúng ta có một tập tin chỉ chứa code Javascript, chúng ta không thể khiến nó hoạt động bằng cách mở trực tiếp tập tin này, mà thay vào đó, **chúng ta cần chèn nó vào một trang HTML**. Thông qua HTML, **các đoạn mã Javascript sẽ được xử lý bằng trình duyệt web**.
- Có 3 phương pháp để có thể chèn Javascript vào một trang HTML: **External Javascript**, **Internal Javascript** và **Inline Javascript**.

1.5.1./ External Javascript

- **External Javascript** là phương pháp mà trong đó, các đoạn code Javascript sẽ được lưu vào một tập tin có phần mở rộng là **.js** và kết nối với trang HTML thông qua phần tử thẻ **<script></script>**.

my-scripts.js	home.html
<pre>document.addEventListener("DOMContentLoaded", function(event) { alert('External Javascript đã hoạt động'); });</pre>	<pre><html lang="vi"> <head> </head> <body> <script type="text/javascript" src="my-scripts.js"></script> </body> </html></pre>

1.5.1./ External Javascript

- **External Javascript** là phương pháp mà trong đó, các đoạn code Javascript sẽ được lưu vào một tập tin có phần mở rộng là **.js** và kết nối với trang HTML thông qua phần tử thẻ **<script></script>**.

Nội dung lưu ý	Demo
Bạn có thể chèn nhiều file External Javascript vào chung 1 trang HTML	<pre>... <body> <script type="text/javascript" src="my-scripts.js"></script> </body> ...</pre>
Bạn có thể chèn External Javascript ở bất kỳ vị trí nào trong trang HTML mà bạn muốn, kể cả trong phần tử thẻ <head></head>	<pre>... <head> <script type="text/javascript" src="my-scripts.js"></script> </head> <body> ... </body> ...</pre>

1.5.1./ External Javascript

mã lệnh trong tập tin *myscript.js*

```
document.getElementById("demo1").innerHTML = "Tài liệu học CSS";  
document.getElementById("demo2").innerHTML = "Tài liệu học MySQL";  
document.getElementById("demo3").innerHTML = "Tài liệu học PHP";
```

Ta có thể nhúng tập tin JavaScript ở bất kỳ nơi nào trong trang web. Tuy nhiên nên đặt chúng ở vị trí cuối cùng trong phần tử `<body>` giống ví dụ. Điều đó giúp cải thiện tốc độ tải giao diện của trang web và tránh một số trường hợp xảy ra lỗi thực thi ngoài mong đợi.

```
<!DOCTYPE html>  
<html>  
<head>  
  <title>Lập Trình Web</title>  
</head>  
<body>  
  <h2>Tài liệu học HTML</h2>  
  <h2 id="demo1"></h2>  
  <h2>Tài liệu học JavaScript</h2>  
  <h2 id="demo2"></h2>  
  <h2 id="demo3"></h2>  
  <script src="../file/myscript.js"></script>  
</body>  
</html>
```

1.5.1./ External Javascript

■ Ưu điểm và nhược điểm External Javascript.

Ưu điểm	Nhược điểm
<p>Bạn có thể viết một kịch bản Javascript dài và phức tạp.</p> <p>Bạn có thể viết một kịch bản Javascript với số lượng lớn dòng code trên tập tin .js</p> <p>Việc tách biệt nơi lưu trữ code Javascript và HTML giúp code của bạn trở nên gọn gàng hơn.</p> <p>Tận dụng cơ chế caching của trình duyệt để giúp trang HTML có tốc độ load và render nhanh hơn.</p> <p>Code Javascript có thể xuất ra dạng nén (.min.js) để có dung lượng nhỏ nhất có thể, từ đó cải thiện tốc độ load trang ở hầu hết các website hiện nay.</p>	<p>Tồn nhiều thao tác & bước thực hiện nhất</p> <p>Nếu xảy ra lỗi code Javascript, bạn tốn thời gian để xác định dòng code lỗi đó đang nằm trên tập tin nào.</p>

1.5.2./ Internal Javascript

Internal Javascript là phương pháp mà trong đó, các đoạn code Javascript được đặt trong phần tử thẻ **<script></script>** của một trang HTML.

```
<!DOCTYPE html>
<html>
<head>
  <title>Lập Trình Web</title>
</head>
<body>
  <h2>Tài liệu học HTML</h2>
  <h2 id="demo1"></h2>
  <h2>Tài liệu học JavaScript</h2>
  <h2 id="demo2"></h2>
  <h2 id="demo3"></h2>
  <script>
    document.getElementById("demo1").innerHTML = "Tài liệu học CSS";
    document.getElementById("demo2").innerHTML = "Tài liệu học MySQL";
    document.getElementById("demo3").innerHTML = "Tài liệu học PHP";
  </script>
</body>
</html>
```

Các đoạn **<script>** có thể được đặt ở bất kỳ nơi nào trong trang web. Tuy nhiên nên đặt chúng ở vị trí cuối cùng trong phần tử **<body>** giống ví dụ bên. Điều đó giúp cải thiện tốc độ tải giao diện của trang web và tránh một số trường hợp xảy ra lỗi thực thi ngoài mong đợi.

1.5.2./ Internal Javascript

```
<!DOCTYPE html>
<html>
<head>
  <title>Lập Trình Web</title>
</head>
<body>
  <h2>Tài liệu học HTML</h2>
  <h2 id="demo1"></h2>
  <h2>Tài liệu học JavaScript</h2>
  <h2 id="demo2"></h2>
  <h2 id="demo3"></h2>
  <script>
    document.getElementById("demo1").innerHTML = "Tài liệu học CSS";
    document.getElementById("demo2").innerHTML = "Tài liệu học MySQL";
    document.getElementById("demo3").innerHTML = "Tài liệu học PHP";
  </script>
  <h2 id="demo4"></h2>
  <h2 id="demo5"></h2>
  <script>
    document.getElementById("demo4").innerHTML = "Tài liệu học jQuery";
    document.getElementById("demo5").innerHTML = "Tài liệu học Bootstrap";
  </script>
</body>
</html>
```

Trong một trang web có thể sử dụng nhiều đoạn **<script>** và trong một đoạn **<script>** thì không giới hạn số lượng câu lệnh.

1.5.2./ Internal Javascript

Ưu và nhược điểm của phương pháp Internal Javascript

Ưu điểm	Nhược điểm
Bạn có thể viết một kịch bản Javascript dài và phức tạp. Tốn ít thao tác thực hiện hơn so với External Javascript	Bạn khó có thể viết một kịch bản Javascript với số lượng lớn dòng code bởi làm thế sẽ khiến file HTML trở nên rất dài và nặng.



- Bên trong đoạn `<script>` chỉ dùng để chứa các mã lệnh JavaScript và tuyệt đối không được đặt vào đó những đoạn `<script>` khác.

1.5.3./ Inline Javascript

Inline Javascript là phương pháp mà trong đó, code Javascript sẽ được chèn trực tiếp vào phần tử HTML thông qua các thuộc tính HTML thuộc nhóm event (Ví dụ: **onload**, **onclick**, **onmouseover**, **onmouseout...**)

Source code	Diễn giải
<pre>... <body onload="alert('Body đã load xong!');"> ... </div> ...</pre>	<p>Giá trị của thuộc tính onload chính là một Javascript statement.</p> <p>Sau khi trình duyệt hoàn tất load phần tử body, một cảnh báo với nội dung Body đã load xong sẽ được hiện lên.</p>
<pre>... <button onclick="alert('Body đã click vào nút hiện cảnh báo!');"> Hiện cảnh báo </button> ...</pre>	<p>Giá trị của thuộc tính onclick chính là một Javascript statement.</p> <p>Ngay sau khi người dùng click vào phần tử <code><button></button></code>, một cảnh báo sẽ xuất hiện.</p>

1.5.3./ Inline Javascript

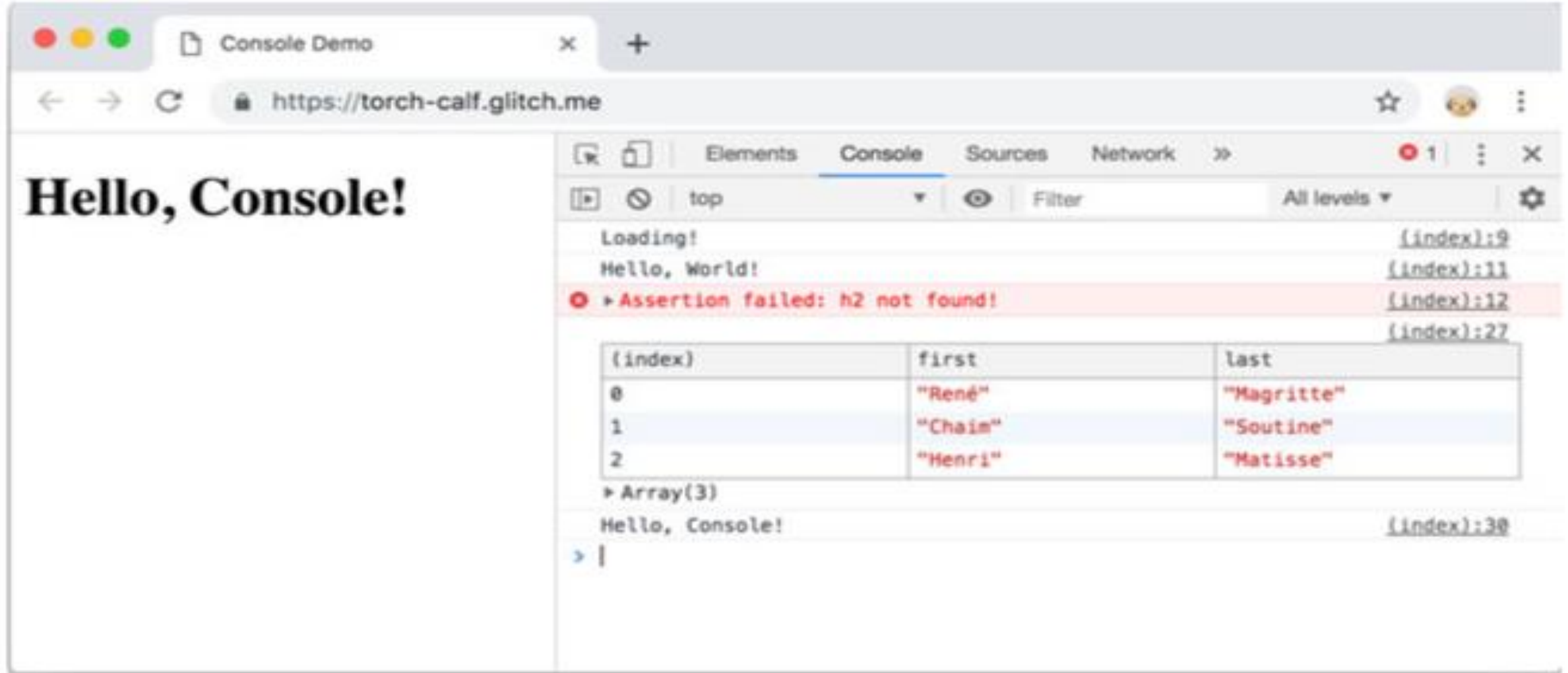
Ưu và nhược điểm của phương pháp Inline Javascript

Ưu điểm	Nhược điểm
Tốn ít thao tác thực hiện nhất	<p>Bạn có thể viết một kịch bản rất ngắn với 1 vài statement đơn giản.</p> <p>Thời điểm trình duyệt xử lý Inline Javascript phụ thuộc vào thuộc tính HTML mà bạn lựa chọn</p>

1.6./ Javascript Console

- **Javascript Console** (có thể gọi là **Browser Console**, **Web Console** hoặc **Console**) là một công cụ mặc định dành cho những nhà phát triển web được tích hợp sẵn bên trong trình duyệt web (**Chrome**, **Firefox**, **Safari**, **Opera...**). Đây là công cụ ghi lại và xuất ra màn hình các dòng **log** (**dòng thông tin**, **thông báo**) về lỗi code, cảnh báo hoặc những dòng thông tin tùy chỉnh được lập trình bằng Javascript.
- **Javascript Console** thường xuất hiện dưới dạng một tab trong giao diện Devtool (công cụ dành cho những nhà phát triển) của các trình duyệt (Elements, Console, Sources, Network...)

1.6./ Javascript Console



The screenshot shows a web browser window with the address bar displaying `https://torch-calf.glitch.me`. The page content on the left says "Hello, Console!". The right-hand console panel is open, showing a log of messages and an error.

Console Log:

- Loading! `(index):9`
- Hello, World! `(index):11`
- Assertion failed: h2 not found!** `(index):12`
- `(index):27`

(index)	first	last
0	"René"	"Magritte"
1	"Chaim"	"Soutine"
2	"Henri"	"Matisse"

Below the table, the console shows:

- `> Array(3)`
- Hello, Console! `(index):30`
- `> |`

1.6./ Javascript Console

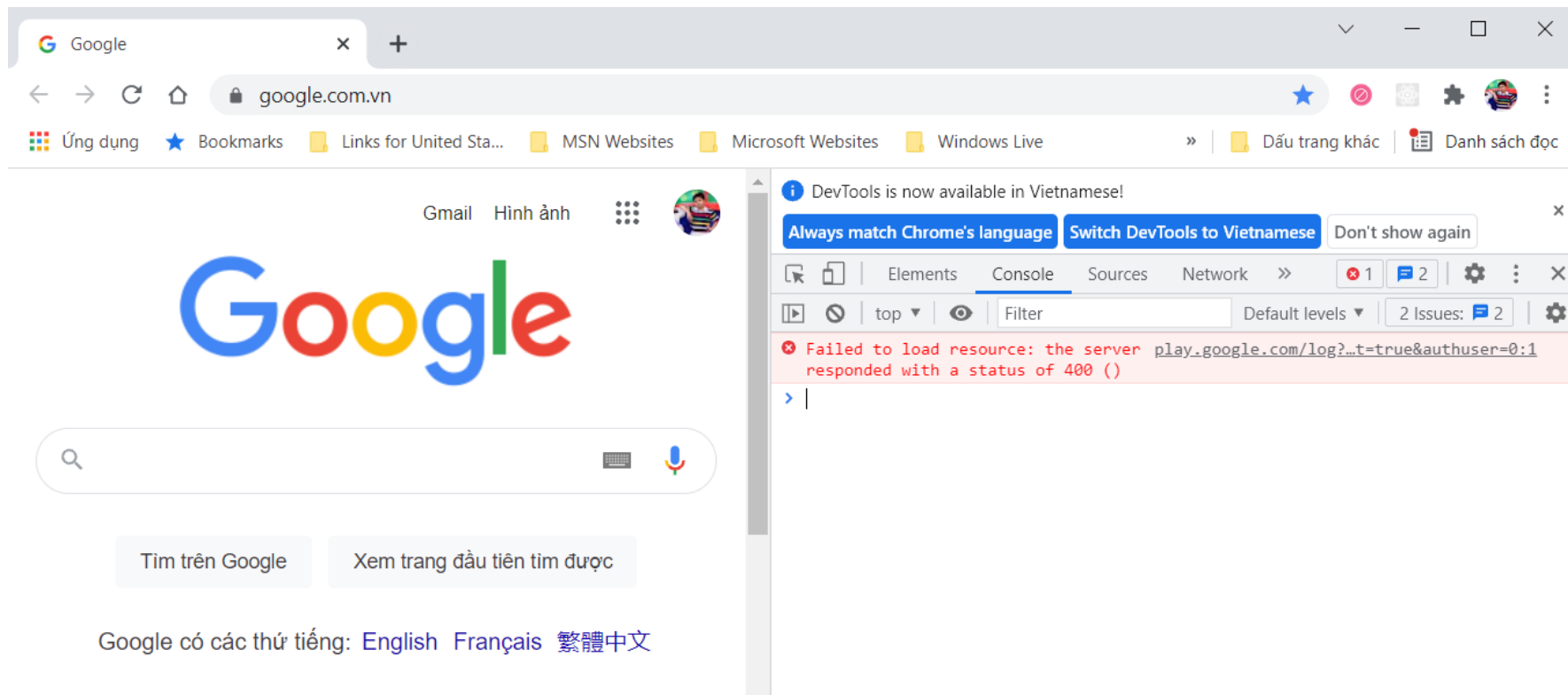
Hầu hết các trình duyệt web chạy trên các hệ điều hành dành cho máy tính (desktop) đều hỗ trợ công cụ Javascript Console. *Trong khi đó chưa có trình duyệt web nào trên mobile hỗ trợ công cụ này.*

Các trình duyệt web hỗ trợ công cụ Javascript Console	Các trình duyệt web chưa hỗ trợ công cụ Javascript Console
Chrome (<i>desktop</i>)	Chrome (<i>mobile</i>)
Safari (<i>desktop</i>)	Safari (<i>mobile</i>)
Firefox (<i>desktop</i>)	Firefox (<i>mobile</i>)
Opera (<i>desktop</i>)	Opera (<i>mobile</i>)
Edge (<i>desktop</i>)	
Internet Explorer (<i>desktop</i>)	

1.6./ Javascript Console

Cách mở giao diện Javascript Console.

Thao tác: nhấn **Ctrl + Shift + J**

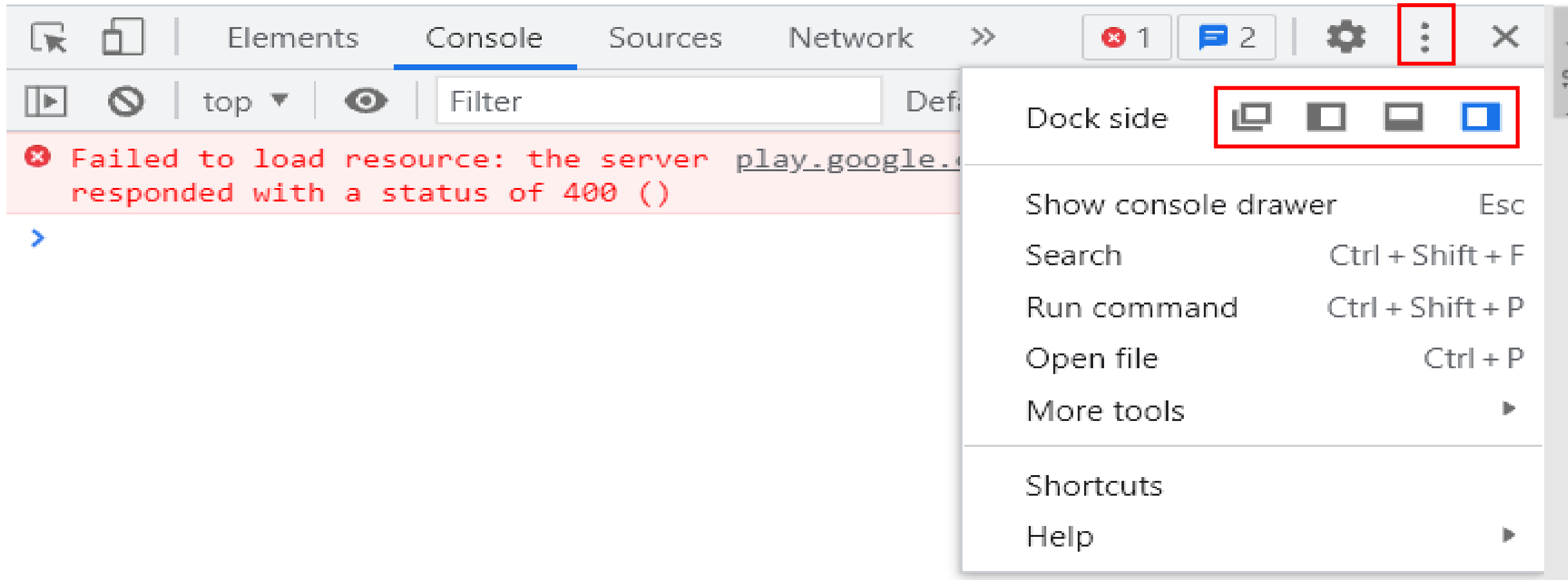


Khu vực thanh
công cụ

Khu vực
hiển thị log

1.6./ Javascript Console

Qui định vị trí hiển thị cửa sổ Console.



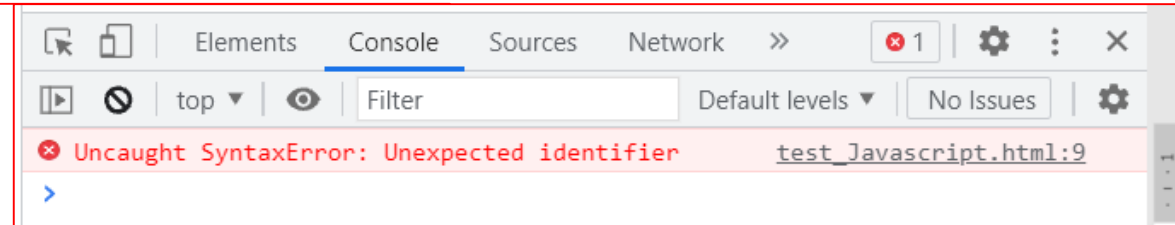
1.6./ Javascript Console

Nhận biết error log (lỗi) trên Javascript Console.

Mặc định, khi một statement trong Javascript xảy ra lỗi (**cú pháp, thiếu các thành phần bắt buộc, không tìm thấy...**), lập tức Javascript Console sẽ hiển thị dòng error log màu đỏ như hình dưới đây:

```
<!DOCTYPE html>
<html>
<body>

  <h1>Tài liệu học HTML</h1>
  <h1 id="skud">Tài liệu học JavaScript</h1>
  <h1>Tài liệu học CSS</h1>
  <script>
    document.getElementById("skud").innerHTML = "Lập Trình Web";
  </script>
</body>
</html>
```

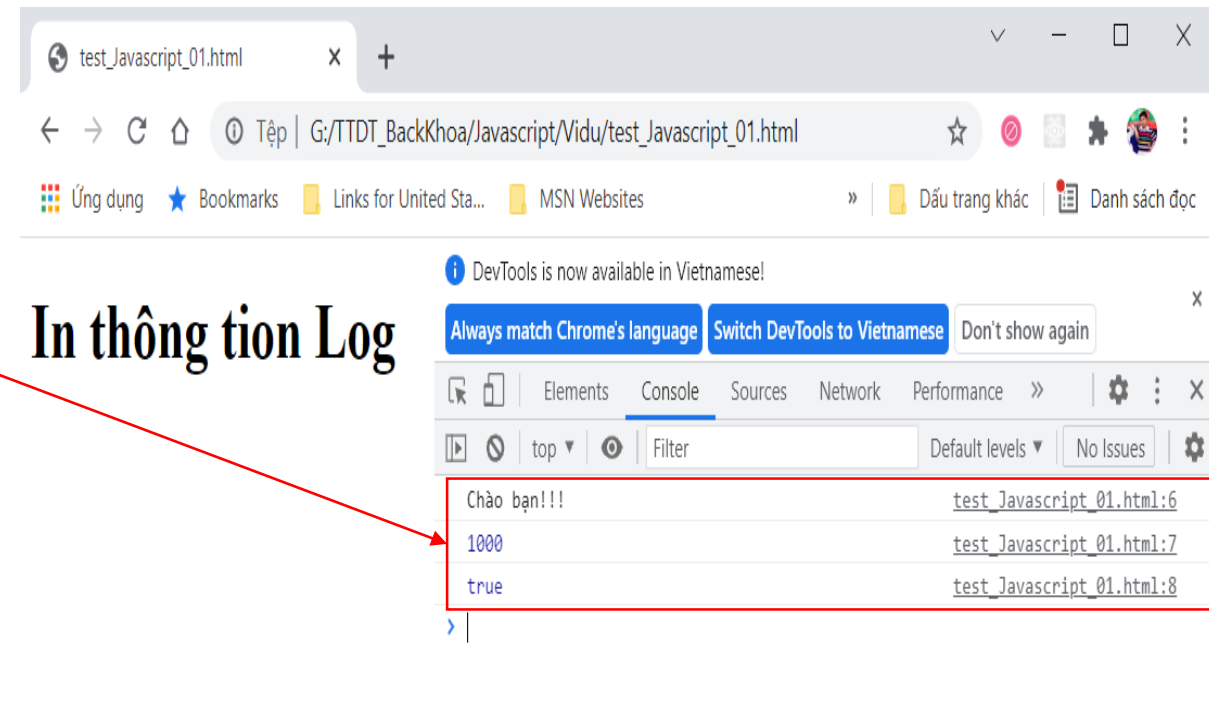


1.6./ Javascript Console

Hiển thị nội dung log lên Javascript Console.

Cú pháp: **console.log('Nội dung');**

```
<!DOCTYPE html>
<html>
<body>
  <h1>In thông tin Log</h1>
  <script>
    console.log('Chào bạn!!!');
    console.log(1000);
    console.log(true);
  </script>
</body>
</html>
```

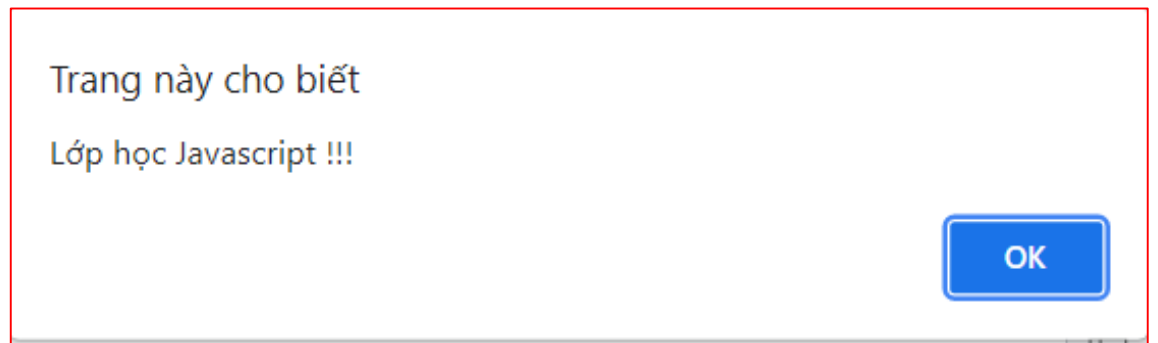


1.7./ Hiển thị dữ liệu ra màn hình trong Javascript

1. Sử dụng alert()

Cú pháp: **alert(Nội dung hiển thị);**

```
<!DOCTYPE html>
<html>
<body>
    <h1>Sử dụng alert</h1>
    <script>
        alert("Lớp học Javascript !!!");
    </script>
</body>
</html>
```

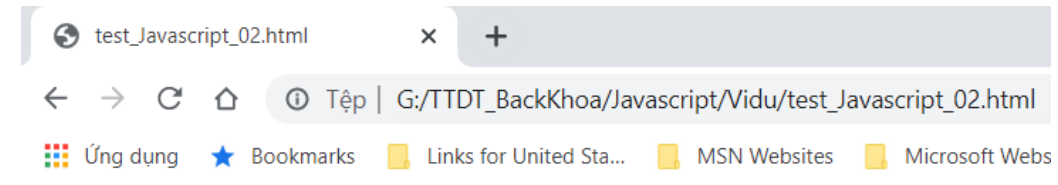


1.7./ Hiển thị dữ liệu ra màn hình trong Javascript

2. Sử dụng document.write()

Cú pháp: **document.write(Nội dung hiển thị);**

```
<!DOCTYPE html>
<html>
<body>
  <h1>Tài liệu học HTML</h1>
  <script>
    document.write("<h1>Tài liệu học CSS</h1>");
  </script>
  <h1>Tài liệu học JavaScript</h1>
  <script>
    document.write("<h1>Tài liệu học MySQL</h1>");
  </script>
  <script>
    document.write("<h1>Tài liệu học PHP</h1>");
  </script>
</body>
</html>
```



Tài liệu học HTML

Tài liệu học CSS

Tài liệu học JavaScript

Tài liệu học MySQL

Tài liệu học PHP

1.7./ Hiển thị dữ liệu ra màn hình trong Javascript

3. Sử dụng document.getElementById().innerHTML

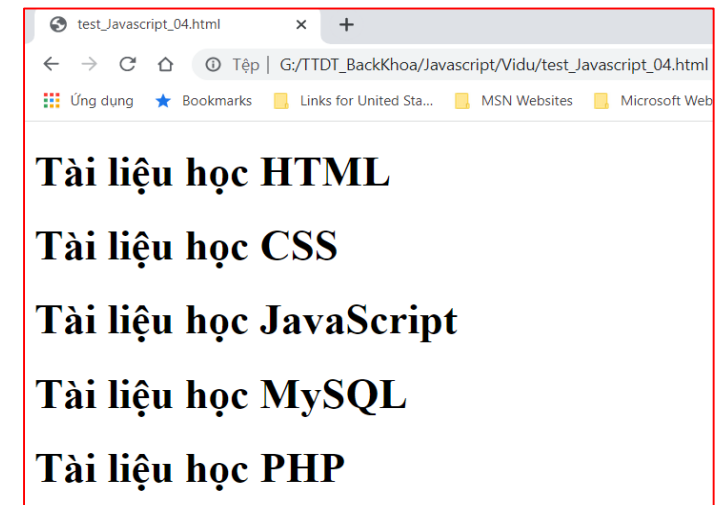
Cú pháp: **document.getElementById().innerHTML=Nội Dung;**

```
<!DOCTYPE html>
<html>
<body>

  <h1>Tài liệu học HTML</h1>
  <h1 id="skud"></h1>
  <h1>Tài liệu học JavaScript</h1>
  <h1>Tài liệu học MySQL</h1>
  <h1 id="raw"></h1>

  <script>
    document.getElementById("skud").innerHTML = "Tài liệu học CSS";
    document.getElementById("raw").innerHTML = "Tài liệu học PHP";
  </script>

</body>
</html>
```



1.7./ Hiển thị dữ liệu ra màn hình trong Javascript

4. Một vài vấn đề trong việc viết nội dung muốn hiển thị

Vấn đề 01

```
<script>
  document.getElementById("demo1").innerHTML = "Lập Trình Web"; //ĐÚNG
  document.getElementById("demo2").innerHTML = 'Lập Trình Web'; //ĐÚNG
  document.getElementById("demo3").innerHTML = "1993"; //ĐÚNG
  document.getElementById("demo4").innerHTML = '1993'; //ĐÚNG
  document.getElementById("demo5").innerHTML = 1993; //ĐÚNG
  document.getElementById("demo6").innerHTML = "Lập Trình Web"; //SAI
  document.getElementById("demo7").innerHTML = 'Lập Trình Web'; //SAI
  document.getElementById("demo8").innerHTML = Lập Trình Web; //SAI
</script>
```

1.7./ Hiển thị dữ liệu ra màn hình trong Javascript

4. Một vài vấn đề trong việc viết nội dung muốn hiển thị

Vấn đề 02

```
<script>
  document.getElementById("demo1").innerHTML = "Lập "Trình Web"; //SAI
  document.getElementById("demo2").innerHTML = 'Lập "Trình Web"; //SAI
  document.getElementById("demo3").innerHTML = "Lập "Trình Web"; //ĐÚNG
  document.getElementById("demo4").innerHTML = 'Lập "Trình Web"; //ĐÚNG
</script>
```

Vấn đề 03

```
<script>
  document.write("<h1>Tài liệu học HTML</h1>");
  document.write("<p><i><u>Tài liệu học HTML</u></i></p>");
</script>
```

1.8./ Biến (Variable) và Hằng (Constant) trong Javascript

1) Biến (Variable) là gì?

- **Biến (Variable)** là một thành phần cơ bản trong Javascript statement có khả năng chứa dữ liệu và giá trị dữ liệu của nó có thể thay đổi trong suốt một kịch bản.
- Để khai báo (tạo) một biến trong kịch bản Javascript, ta sử dụng cú pháp: **var ten_bien;** trong đó:
 - **var** là cú pháp bắt buộc để khai báo một biến
 - **ten_bien** là tên mà ta đặt cho biến

1.8./ Biến (Variable) và Hằng (Constant) trong Javascript

1) Biến (Variable) là gì?

■ Ví dụ:

```
var a; // khai báo biến a
```

```
var b; //khai báo biến b
```

```
var a1, a2, b100; //khai báo 3 biến a1, a2, b100
```

Ta có thể vừa khai báo biến, vừa gán dữ liệu dữ liệu ban đầu cho biến đó. Ví dụ:

```
var a=100; //khai báo biến a chứa dữ liệu ban đầu là 100
```

```
var hoten= “Giang Hào Côn”; //khai báo biến hoten chứa dữ  
liệu ban đầu là chuỗi giá trị Giang Hào Côn
```

1.8./ Biến (Variable) và Hằng (Constant) trong Javascript

1) Biến (Variable) là gì?

Quy tắc đặt tên biến

- Tên biến là một tập hợp gồm một hoặc nhiều ký tự.
- Tên biến có thể chứa các ký tự trong danh sách bên dưới:

Các chữ cái in hoa	A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z
Các chữ cái thường	a, b, c, d, e, f, g, h, i, j, k, l, m, n, o, p, q, r, s, t, u, v, w, x, y, z
Các chữ số	0, 1, 2, 3, 4, 5, 6, 7, 8, 9
Dấu gạch dưới	_
Dấu đô la	\$ (khuyến khích KHÔNG NÊN sử dụng dấu đô la trong việc đặt tên biến)

- Tên biến tuyệt đối không được phép chứa các ký tự đặt biệt (**Ví dụ như: @, #, !, %, ^, &,**)

1.8./ Biến (Variable) và Hằng (Constant) trong Javascript

1) Biến (Variable) là gì?

Quy tắc đặt tên biến

- Tên biến không được bắt đầu bằng một chữ số.

Một số ví dụ đặt tên biến đúng quy tắc	Một số ví dụ đặt tên biến sai quy tắc
W webcoban Webcoban webCobAn we9co3an _webc_oban	9webcoban (sai vì bắt đầu bằng chữ số) web%^coban (sai vì chứa ký tự đặc biệt)

1.8./ Biến (Variable) và Hằng (Constant) trong Javascript

1) **Biến (Variable)** là gì?

Quy tắc đặt tên biến

- Tên biến có phân biệt trường hợp chữ in hoa và chữ thường (*Ví dụ, webcoban và Webcoban là hai biến khác nhau*)
- Tên biến phải duy nhất (*không được khai báo một biến có tên trùng với tên của một biến đã được khai báo trước đó*)
- Không được đặt tên biến trùng với các từ dành riêng trong JavaScript

1.8./ Biến (Variable) và Hằng (Constant) trong Javascript

2) Hằng (Constant) là gì?

Hằng (Constant) là một **thành phần cơ bản** trong Javascript statement có khả năng chứa dữ liệu và giá trị dữ liệu của nó **không thể thay đổi** trong suốt một kịch bản.

- Để khai báo (tạo) một biến trong kịch bản Javascript, ta sử dụng cú pháp: **const ten_hang = <giá trị>;** trong đó:
 - **const** là cú pháp bắt buộc để khai báo một hằng
 - **ten_hang** là tên mà ta đặt cho hằng
 - **<giá trị>** là giá trị của hằng

1.8./ Biến (Variable) và Hằng (Constant) trong Javascript

3) Sự khác biệt giữa Biến và Hằng

Dữ liệu của biến có thể thay đổi	Dữ liệu của hằng không thể thay đổi
<p>Ví dụ:</p> <pre>var z1 = 5; z1 = 10; // Code vẫn chạy</pre>	<p>Ví dụ</p> <pre>const z1 = 5; z1 = 10; // Code lỗi</pre>
<pre>var h5 = 1; h5 = h5 + 1; // Code vẫn chạy</pre>	<pre>const h5 = 1; h5 = h5 + 1; // Code lỗi</pre>