



LẬP TRÌNH JAVASCRIPT

MỤC TIÊU

MÔN HỌC

- 1 LẬP TRÌNH VỚI NGÔN NGỮ JAVASCRIPT**
- 2 HƯỚNG ĐỐI TƯỢNG TRONG JAVASCRIPT**
- 3 MÔ HÌNH BOM VÀ DOM**
- 4 CÁC THƯ VIỆN VÀ FRAMEWORKS**



3.1

GIỚI THIỆU JAVASCRIPT

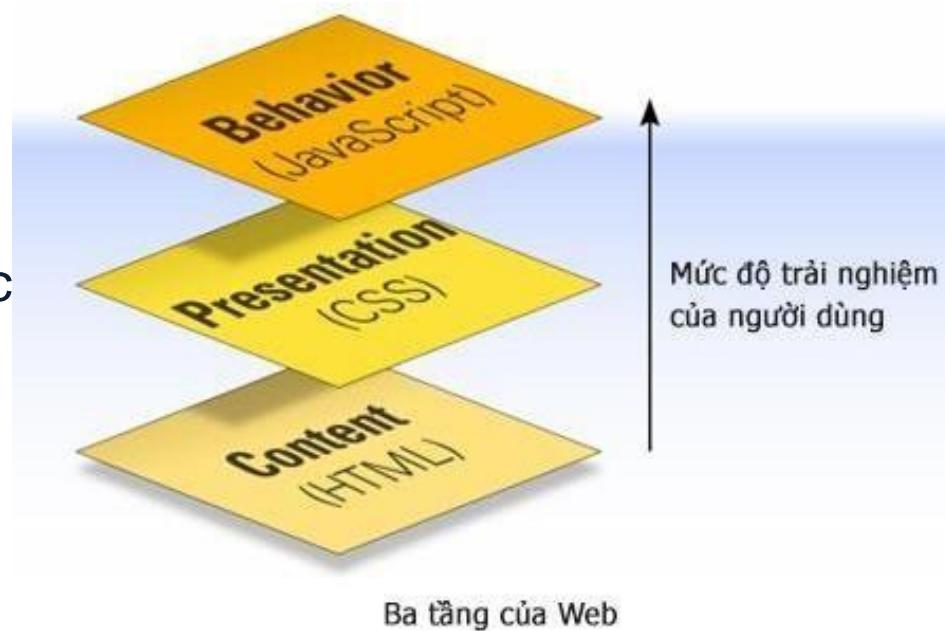
JAVASCRIPT LÀ GÌ?

❑ JAVASCRIPT

- Hành động của trang web

❑ Javascript hồi đáp lại các tương tác của người dùng

- Khi người dùng nhấn chuột
- Khi người dùng nhấn vào menu



HTML

Nội dung

CSS

Cách trình bày

SỰ PHÁT TRIỂN CỦA JAVASCRIPT

1995

Javascript ra đời với tên gọi Livescript.

1999

ES3 ra mắt

AJAX(XMLHttpRequest) trở nên phổ biến trong các ứng dụng

2009

Douglas Crockford nảy ra ý tưởng về OOP, và ES5 ra đời.

ES6/ECMAScript 2015 ra mắt

2019

ECMAScript 8

1997

2000-2005

2015

TỪ 1995

VÌ SAO NÊN HỌC JAVASCRIPT?

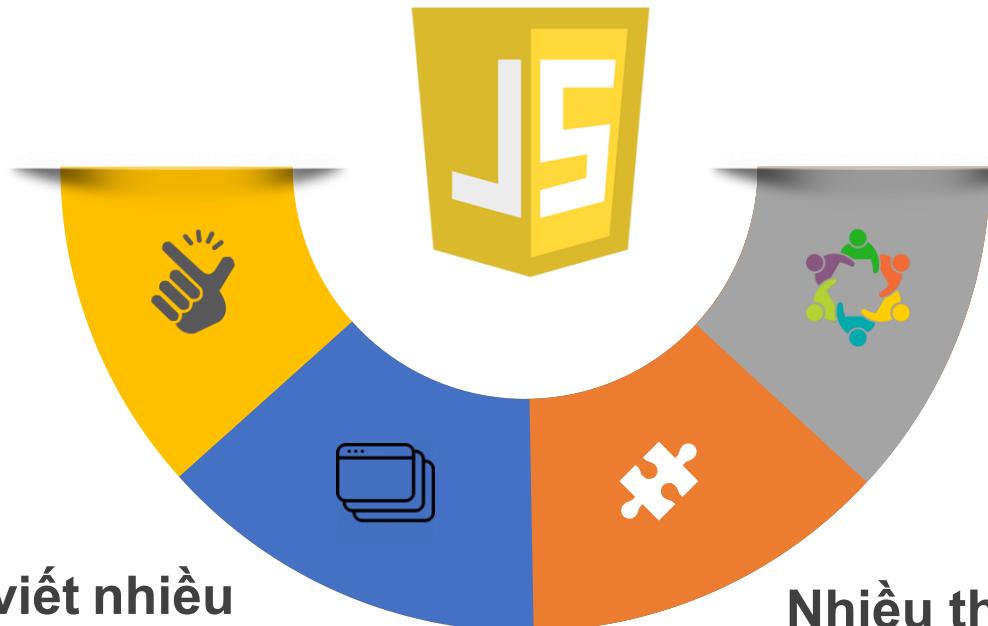
**Ngôn ngữ lập trình dành
cho người mới bắt đầu**

Phổ biến
Đễ học

**Sử dụng để viết nhiều
loại ứng dụng**

Web development, Web application (APIs),
Server Application (Node JS), Web Server
(Node JS), Games, Mobile Application,...

**Javascript là ngôn ngữ
lập trình thông dịch, nhẹ.**



Cộng đồng Javascript
Example: Stack Overflow

Nhiều thư viện và frameworks

Library: Jquery, React
Framework: Angular, APIs và
NodeJS, Vue.js

Javascript có thể làm gì?

⑩ Thực hiện các tác vụ phía Client

- Tạo menu xổ xuống
- Thay đổi nội dung trên trang web
- Thêm các thành phần động vào trang web
- Xác định tính hợp lệ của form (Validate form).
- Làm việc với ảnh để hồi đáp lại người dùng...

validate form

Dropdown menu

NO DROPODOWN DROPODOWN 1 ▾ NO DROPODOWN DROPODOWN 2 ▾

DROP ITEM 1
DROP ITEM 2
DROP ITEM 3

Sign Up
It's free and always will be.

First Name: Phạm
Last Name: Thanh
Your Email: iam_bee2000@gmail.com
Re-enter Email: iam_bee@gmail.com
New Password:
I am: Female
Birthday: Mar 17 1987
Why do I need to provide my birthday?
Sign Up

Your emails do not match. Please try again.



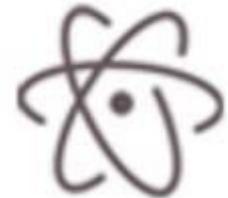
Xác định tính hợp lệ cho Form trên Client thay vì trên Server sẽ làm giảm tải cho Server và trả về kết quả lập tức cho người dùng

JAVASCRIPT CÓ THỂ LÀM GÌ?

- Một số trang web sử dụng Javascript tạo ra các hiệu ứng
 - <http://dibusoft.com/>
 - <http://www.hotel-oxford.ro/>
- Javascript có thể viết GAME
 - <http://www.themaninblue.com/experiment/BunnyHunt/>
 - <http://www.e-forum.ro/bomberman/dynagame.html#top>

CÀI ĐẶT MÔI TRƯỜNG

IDE

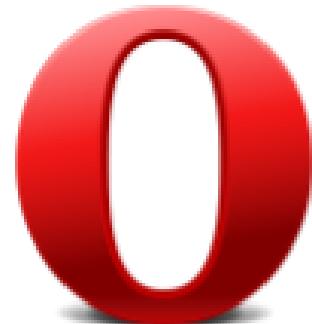


Sublime Text



WebStorm

BROWSERS



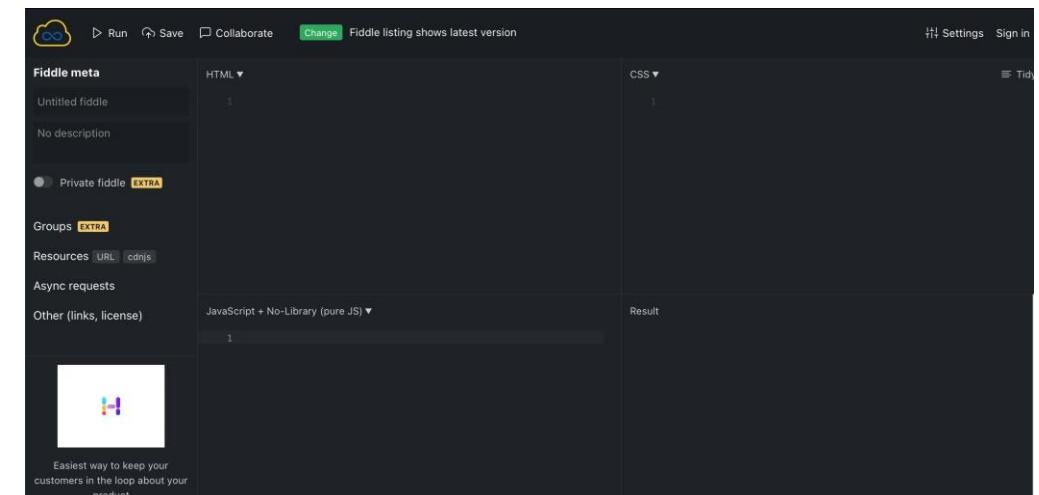
CÀI ĐẶT MÔI TRƯỜNG

Online editors

[https://playcode.io/javascript
-online](https://playcode.io/javascript-online)



<https://jsfiddle.net>



Hello World với Console trên browsers



A screenshot of a Google Chrome browser window. The main content area shows the Google homepage. A context menu is open over the search bar, with the "Inspect" option highlighted. In the bottom right corner of the browser, a floating developer tools panel is visible, titled "Console". It displays the following JavaScript code and output:

```
> console.log("Hello World");
Hello World
< undefined
>
```

The developer tools panel also includes tabs for "Elements", "Sources", "Network", and "What's New". The "What's New" tab is currently selected and shows "Highlights from the Chrome 99 update" with sections on "Throttling web socket requests" and "Recorder panel improvements".

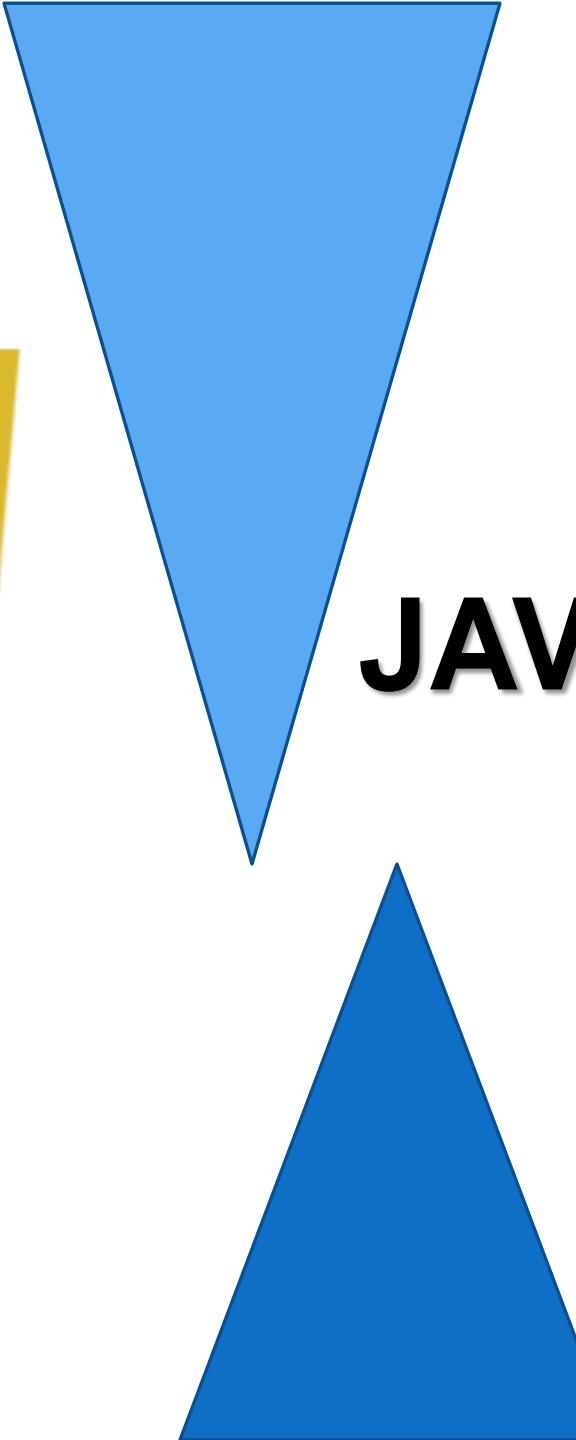
3.1.

LẬP TRÌNH

JAVASCRIPT

3.1.1 JAVASCRIPT CODE

3.1.2



3.1.1

JAVASCRIPT CODE

MỘT CHƯƠNG TRÌNH

JAVASCRIPT GỒM

Một chương trình Javascript cũng giống như chương trình viết bằng các ngôn ngữ khác

1 | Một chương trình Javascript bao gồm nhiều câu lệnh

2 | Một câu lệnh được tạo nên từ các toán tử, từ định danh, biến,....

var x = 5;

Câu lệnh này khai báo biến x có giá trị bằng 5

J A V A S C R I P T

TRÊN TRANG WEB

Có 2 cách

- 1 |** Trực tiếp trong HTML
- 2 |** Liên kết với tập tin javascript bên ngoài trang web

J A V A S C R I P T

TRÊN TRANG WEB

1 | Trực tiếp trong HTML

<> index.html ×

<> index.html > ...

```
1  <html>
2  |  <script type="text/javascript">
3  |  |  alert("Hi there!");
4  |  </script>
5  </html>
```

Popup với dòng chữ "Hi there!" xuất hiện



J A V A S C R I P T

TRÊN TRANG WEB

1 | Trực tiếp trong HTML

```
<> index.html <-->

<> index.html > ...
1  <!DOCTYPE html>
2  <html>
3    <head>
4      <title>This goes in the tab of your browser</title>
5    </head>
6    <body>
7      The content of the webpage
8      <script>
9        | console.log("Hi there!");
10       </script>
11     </body>
12   </html>
13
```

Nên đặt mã JavaScript trong thẻ `<script>` đặt ở cuối phần body để đảm bảo tất cả các thành phần đã được load

J A V A S C R I P T

TRÊN TRANG WEB

2 | Liên kết với tập tin javascript bên ngoài trang web

The screenshot shows a code editor interface with two tabs open:

- chapter1_alert.js**: Contains the following JavaScript code:

```
1 alert("Saying hi from a different file!");
```
- index.html**: Contains the following HTML code:

```
<html>
<script type="text/javascript" src="chapter1_alert.js"></script>
</html>
```

The sidebar on the left features icons for file operations like copy, paste, search, and refresh.

QUY TẮC CƠ BẢN CỦA JAVASCRIPT

- Javascript phân biệt chữ hoa thường
- Javascript bỏ qua ký tự cách
- Chú thích
 - Chú thích nhiều dòng

```
/* Đây là chú thích nhiều dòng  
Bạn có thể viết chú thích trên nhiều dòng */
```
 - Chú thích trên một dòng

```
// Đây là chú thích một dòng  
// Chú thích này chỉ chú thích được cho một dòng
```
- Ký tự **chấm phẩy** `(` để kết thúc một dòng lệnh. Ký tự này là bắt buộc.

CÂU LỆNH JAVASCRIPT

- Câu lệnh đơn

```
var x = 4;
```

- Câu lệnh kép

```
if(x==1) {  
    //Code here  
} else {  
    //Code here  
}
```

TỪ KHOÁ

- ❑ Javascript có các tập từ khoá

break	delete	if	this	while
case	do	in	throw	with
catch	else	instanceof	try	
continue	finally	new	typeof	
debugger	for	return	var	
default	function	switch	void	

BUILT IN FUNCTION

- Javascript cung cấp nhiều hàm dựng sẵn (built-in function)
 - alert ()
 - prompt ()
 - console
 - console.log ()
 - console.error ()
 - math
 - math.random ()
- Một số hàm không được hỗ trợ trên tất cả các trình duyệt
- Javascript cũng cho phép người dùng tự định nghĩa hàm.



3.3

BIẾN VÀ KIỂU DỮ LIỆU

VARIABLES (BIẾN)

- **Variable (biến)** dùng để lưu trữ dữ liệu

- Cú pháp:

```
var/let/const <variable_name>;
```

- Cách đặt tên biến

- Tên biến bao gồm chữ cái và số, nhưng không được bắt đầu bằng số
- Tên biến không bao gồm dấu cách và dấu câu, ngoại trừ dấu gạch dưới (_)

- Có thể khai báo biến trên một dòng

```
var x, y, zeta;
```

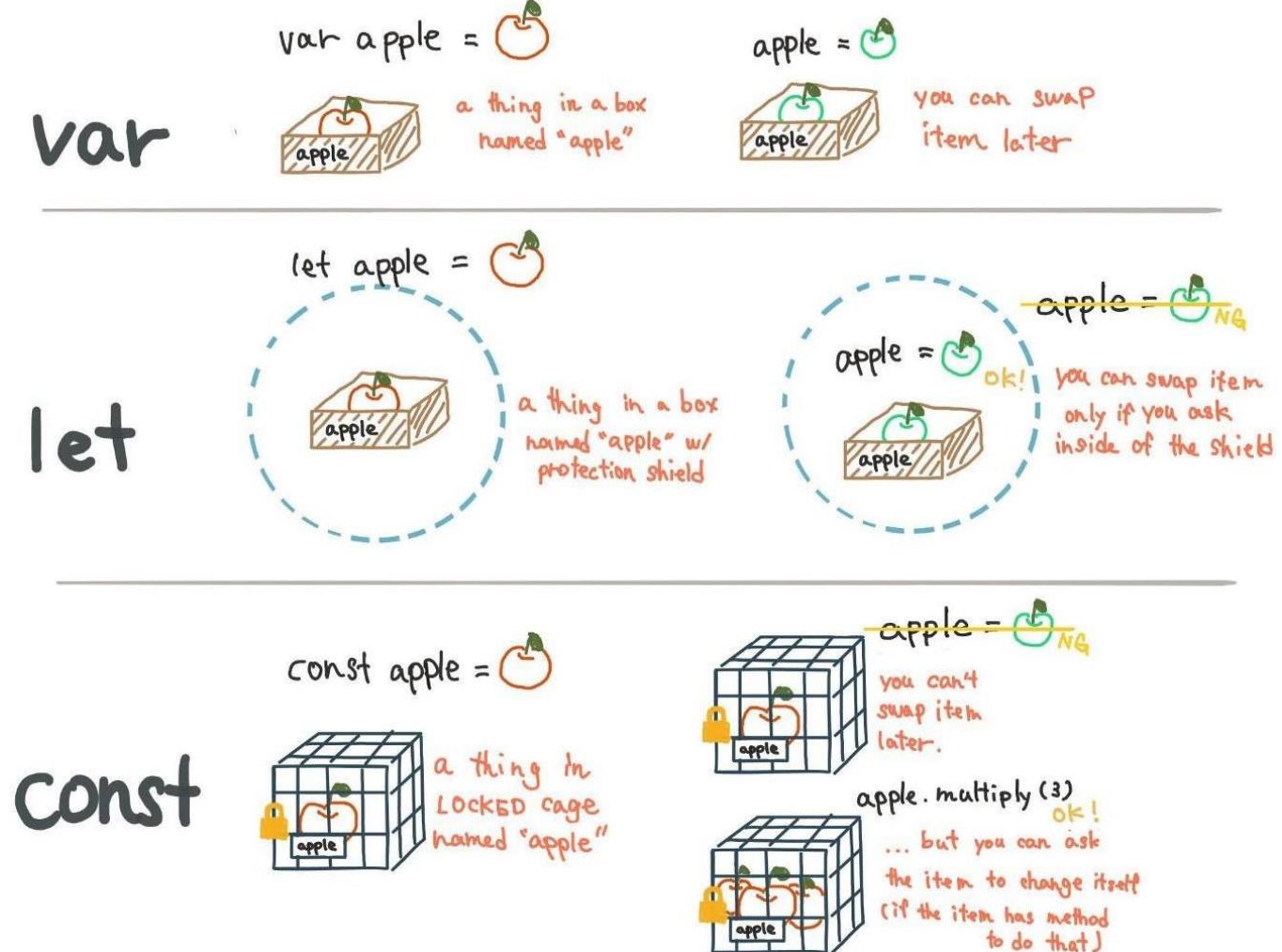
- Có thể vừa khai báo vừa khởi tạo giá trị cho biến

```
var x=1;
```

```
var x=1, y="hello";
```

VARIABLES (BIẾN)

- **var**: có phạm vi global scope
- **let**: có phạm vi block scope { }
- **const**: gán giá trị cho biến chỉ 1 lần, không được thay đổi giá trị



KIỀU DỮ LIỆU

- Javascript hỗ trợ những kiểu dữ liệu - Primitive data sau:
 - String
 - Number
 - BigInt
 - Boolean
 - Symbol
 - Undefined
 - Null

KIỀU DỮ LIỆU - STRING

- Cách để khai báo string

- Dấu nháy đôi " "
- Dấu nháy đơn ' '
- Backticks ``

- Ví dụ

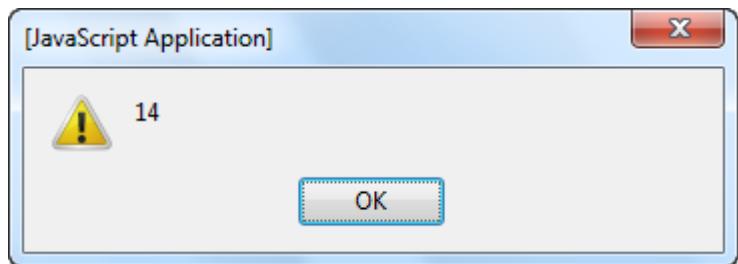
```
let funActivity = 'Let0s learn JavaScript'; //Error
```

```
let language = "JavaScript";
let message = `Let's learn ${language}`;
console.log(message);
```

KIỀU DỮ LIỆU – STRING

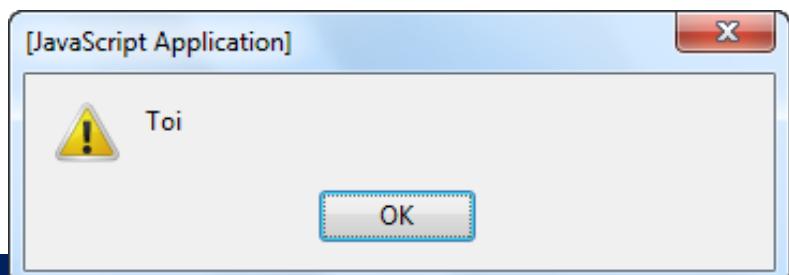
- Thuộc tính **length**

```
var x = "Toi la String."; alert(x.length);
```



- Phương thức **substring**

```
var x = "Toi la String."; alert(x.substring(0, 3));
```



KIỂU DỮ LIỆU – NUMBER

- Javascript không chia ra kiểu Double, Integer...như các ngôn ngữ khác.
- Javascript gộp lại thành một kiểu duy nhất là Number
- Giới hạn của kiểu number là $2^{53}-1$ và $-(2^{53}-1)$

```
let intNr = 1;  
let decNr = 1.5;  
let expNr = 1.4e15;  
let octNr = 0o10; //decimal version would be 8  
let hexNr = 0x3E8; //decimal version would be 1000  
let binNr = 0b101; //decimal version would be 5
```

KIỂU DỮ LIỆU – BOOLEAN

- Kiểu boolean có hai giá trị là **true** và **false**
- Các biểu thức Boolean thường được sử dụng trong các *cấu trúc điều khiển*

```
let bool1 = false;  
let bool2 = true;
```

```
if (x > 18) {  
    alert("Hi");  
}
```

KIỀU DỮ LIỆU – UNDEFINED VÀ NULL

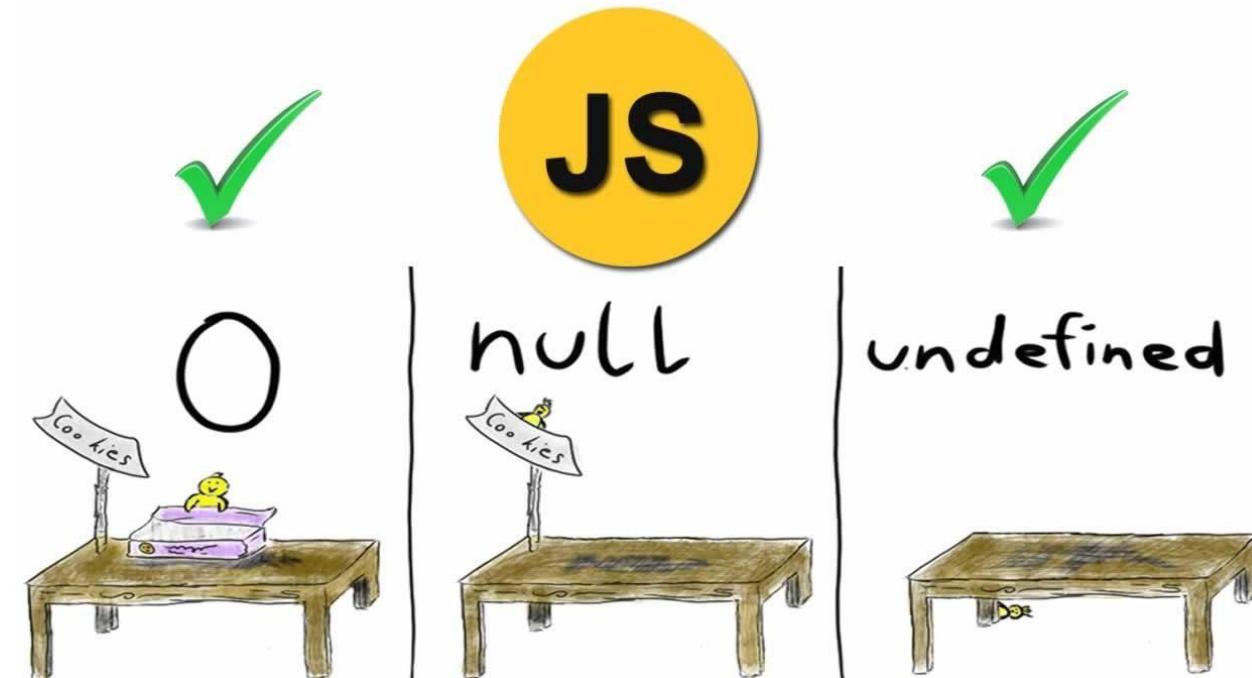
- **Null** (giá trị rỗng) & **undefined**: không thuộc bất kỳ kiểu mô tả nào
- Là giá trị đặc biệt đại diện cho nothing (không có gì), empty(rỗng) hoặc value unknown (không xác định)

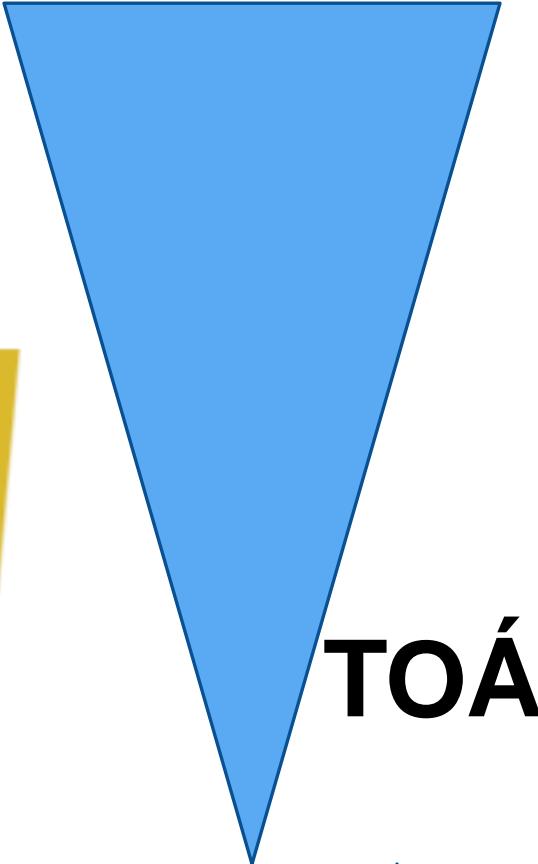
```
let terribleThingToDo = undefined;  
let lastName;  
console.log("Same undefined:", lastName ===  
terribleThingToDo);  
let betterOption = null;  
console.log("Same null:", lastName === betterOption);
```

Result:

Same undefined: true

Same null: false





3.4

TOÁN TỬ VÀ BIỂU THỨC

JAVASCRIPT MULTIPLE VALUES – ARRAY (MẢNG)

- **Array** (Mảng) : là kiểu dữ liệu dùng để lưu một tập các dữ liệu có kiểu giống nhau
- Tạo mảng

```
arr1 = new Array("hello", "world", "javascript", "array");  
arr2 = ["hello", "world", "javascript", "array"];
```

```
console.log(arr1);  
console.log(arr2);
```

- Truy xuất phần tử (element)

```
console.log(arr1[0]);  
console.log(arr1[-1]);
```

ÉP KIỂU

- Ép kiểu ngầm định
 - Trình thông dịch tự động chuyển kiểu

```
var x = 100;  
alert("Hello" + x);
```



ÉP KIỂU

- Ép kiểu tường minh
 - Ép kiểu số thành chuỗi

```
var x = String(100);  
alert(typeof(x));
```

- Ép kiểu chuỗi thành số

```
var x = "100";  
var y = Number(x);  
alert(typeof(y));
```

ÉP KIỀU

- String to Array: **split()**

```
let result = "Hello JavaScript";
let arr_result = result.split(" ");
console.log(arr_result);
```

- Array to String: **join()**

```
let letters = ["a", "b", "c"];
let x = letters.join();
console.log(x);
```

OPERATORS (TOÁN TỬ)

- Toán tử số học

Toán tử	Giải thích
+	Cộng
-	Trừ
*	Nhân
/	Chia
%	Chia lấy dư

- Toán tử logic

Toán tử	Giải thích
&	Và
	Hoặc
^	XOR
!	NOT

OPERATORS (TOÁN TỬ)

- Toán tử quan hệ và toán tử bằng

Toán tử	Giải thích
>	Lớn hơn
<	Bé hơn
>=	Lớn hơn hoặc bằng
<=	Bé hơn hoặc bằng
==	Bằng
!=	Khác

OPERATORS (TOÁN TỬ)

- Toán tử một ngôi

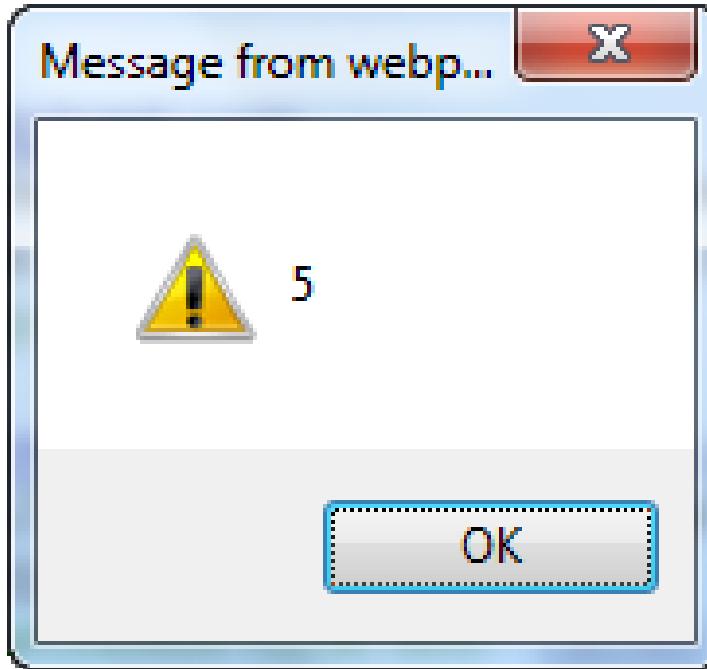
Toán tử	Giải thích
+	Chuyển toán hạng sang số dương
-	Chuyển toán hạng sang số âm
++	Tăng
--	Trừ

- Toán tử gom nhóm

Toán tử	Giải thích
(...)	$(x+y)$

OPERATORS (TOÁN TỨ)

```
var x = 4;  
x++;  
alert (x);
```



OPERATORS (TOÁN TỬ)

- Sự khác nhau giữa $++x$ và $x++$

```
var x = 4;  
var y = ++x;  
alert ("x = "+x+"y = "+y);
```



```
var x = 4;  
var y = x++;  
alert ("x = "+x+"y = "+y);
```



MATH METHODS

- Math có nhiều methods cho phép thực hiện các phép tính và phép toán trên nhiều số.
- Tìm số lớn nhất và nhỏ nhất

```
let highest = Math.max(2, 56, 12, 1, 233, 4);  
console.log(highest);
```

```
let highestOfWords = Math.max("hi", 3, "bye");  
console.log(highestOfWords);
```



3.5

CẤU TRÚC ĐIỀU KHIỂN

CÁC LỆNH LOGIC

- if và if else
- else if
- Toán tử bậc 3 có điều kiện
- switch

LOGIC STATEMENTS (LỆNH LOGIC)

- Cú pháp

```
if (expression) {  
    //code here  
}
```

- Ví dụ

```
var x = 3; var y = 4;  
if (x == y) {  
    //Thực hiện  
}
```

LOGIC STATEMENTS (LỆNH LOGIC)

- Cú pháp

```
if(expression) {  
    //code here  
} else {  
    //code here  
}
```

- Ví dụ

```
let rain = true;  
if(rain) {  
    console.log("** Taking my umbrella when I need to go outside  
    **");  
} else {  
    console.log("** I can leave my umbrella at home **");  
}
```

LOGIC STATEMENTS (LỆNH LOGIC)

- Cú pháp

```
if (expression) {  
    //code here  
} else if (expression) {  
    //code here  
} else if (expression) {  
    //code here  
} else {  
    //code here  
}
```

- Ví dụ

```
if (age < 3) {  
    console.log("Access is free  
under three.");  
} else if (age < 12) {  
    console.log("the fee is 5  
dollars");  
} else if (age < 65) {  
    console.log("A regular ticket  
costs 10 dollars.");  
} else if (age >= 65) {  
    console.log("A ticket is 7  
dollars.");  
}
```

CONDITIONAL TERNARY OPERATORS

(Toán tử bậc ba có điều kiện)

- Cú pháp

```
operand1 ? operand2 : operand3;
```

```
expression ? statement for true : statement associated  
with false;
```

- Ví dụ

```
let access = age < 18 ? "denied" : "allowed";
```

SWITCH STATEMENT (LỆNH SWITCH)

- Cú pháp

```
switch(expression) {  
    case value1:  
        // code to be executed  
        break;  
    case value2:  
        // code to be executed  
        break;  
    case value-n:  
        // code to be executed  
        break;  
}
```

- Ví dụ

```
switch(activity) {  
    case "Get up":  
        console.log("It is 6:30AM");  
        break;  
    case "Breakfast":  
        console.log("It is 7:00AM");  
        break;  
    case "Drive to work":  
        console.log("It is 8:00AM");  
        break;  
}
```

SWITCH STATEMENT (LỆNH SWITCH)

- Cú pháp

```
switch(expression) {  
    case value1:  
        // code to be executed  
        break;  
    case value2:  
        // code to be executed  
        break;  
    case value-n:  
        // code to be executed  
        break;  
    default:  
        // when no cases match  
        break;  
}
```

- Ví dụ

```
switch(activity) {  
    case "Get up":  
        console.log("It is 6:30AM");  
        break;  
    case "Breakfast":  
        console.log("It is 7:00AM");  
        break;  
    case "Drive to work":  
        console.log("It is 8:00AM");  
        break;  
    default:  
        console.log("I cannot determine the  
        current time.");  
        break;  
}
```

LOOPS (VÒNG LẶP)

- Break, continue
- Lệnh lặp không biết trước số lần lặp
 - while
 - do while
- Lệnh lặp biết trước số lần lặp
 - For
 - For of
 - For in (học ở phần object)

WHILE LOOPS

- Cú pháp

```
while (condition) {  
    // code that gets executed as  
    //long as the condition is true  
}
```

- Ví dụ

```
let i = 0;  
while (i < 10) {  
    console.log(i);  
    i++;  
}
```

DO WHILE LOOPS

- Cú pháp

```
do {  
    // code to be executed if the condition is true  
} while (condition);
```

- Ví dụ

```
do {  
    number = prompt("Please enter a number between 0 and  
100: ");  
} while (!(number >= 0 && number < 100));
```

FOR LOOPS

- Cú pháp

```
for (initialize variable; condition; statement)
{
    // code to be executed
}
```

- Ví dụ

```
for (let i = 0; i < 10; i++)
{
    console.log(i);
}
```

Sơ đồ hoạt động của vòng lặp for

1. Cài đặt biến $i=0$
2. Kiểm tra điều kiện $i < 10$
3. Nếu điều kiện là đúng, thực thi khối lệnh. Nếu điều kiện sai, vòng lặp kết thúc tại đây
4. Thực hiện câu lệnh $i++$
5. Trở lại bước 2

VÒNG LẶP VÀ MẢNG

- Cú pháp

```
let arr = [some array];
for (initialize variable; variable smaller than arr.length; statement) {
    // code to be executed
}
```

- Ví dụ

```
let names = ["Chantal", "John", "Maxime", "Bobbi", "Jair"];
for (let i = 0; i < names.length; i++) {
    console.log(names[i]);
}
```

VÒNG LẶP FOR OF

- Cú pháp

```
let arr = [some array];
for (let variableName of arr) {
    // code to be executed
    // value of variableName gets updated every iteration
    // all values of the array will be variableName once
}
```

- Ví dụ

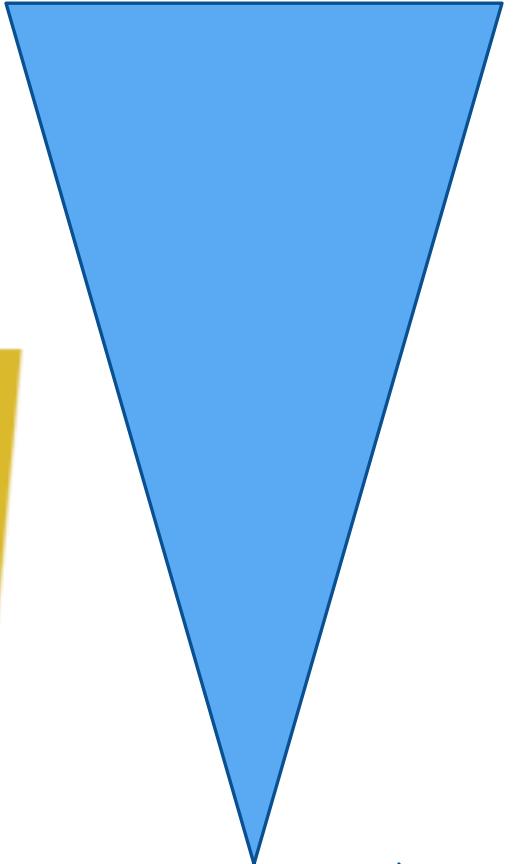
```
let names = ["Chantal", "John", "Maxime", "Bobbi", "Jair"];
for (let name of names) {
    console.log(name);
}
```

LỆNH BREAK, CONTINUE

- Dùng để kiểm soát sơ đồ thực thi vòng lặp.
- Break: dừng vòng lặp và di chuyển sang mã bên dưới vòng lặp
- Continue: dừng vòng lặp và quay lại đầu vòng lặp, kiểm tra điều kiện
(continue trong for: thực hiện câu lệnh và sau đó kiểm tra điều kiện)

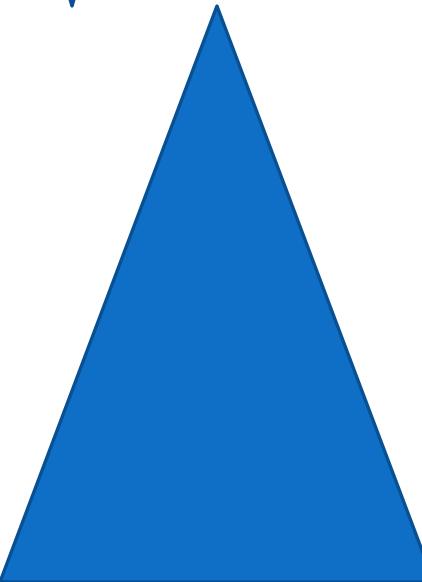
```
for (let i = 0; i < 10; i++) {  
    console.log(i);  
    if (i === 4) {  
        break;  
    }  
}
```

```
let i = 1;  
while (i < 50) {  
    if (i % 2 === 0) {  
        continue;  
    }  
    console.log(i);  
    i++;  
}
```



3.6

HÀM



HÀM (FUNCTION)

- Hàm: để thực hiện một chức năng cụ thể
- Cú pháp:
 - Hàm cơ bản
 - Hàm có tham số
 - Hàm trả về giá trị

HÀM (FUNCTION)

- **Hàm cơ bản**

- **Hàm xây dựng sẵn:** prompt(), console.log(), push(), sort(), ...
- **Hàm tự xây dựng**

```
function nameOfTheFunction() {  
    //content of the function  
}
```

- **Gọi hàm** nameOfTheFunction();

- **Ví dụ**

```
function sayHello() {  
    let you = prompt("What's your name? ");  
    console.log("Hello", you + "!");  
}  
sayHello();
```

HÀM (FUNCTION)

- Hàm có tham số
 - Cú pháp

```
function myFunc(param1, param2) {  
    // code of the function;  
}
```

- Ví dụ

```
function addTwoNumbers(x, y) {  
    console.log(x + y);  
}  
addTwoNumber(5, 5);
```

HÀM (FUNCTION)

- Hàm có tham số

```
function tester(para1, para2) {  
    console.log(para1 + " " + para2);  
}  
const arg1 = "argument 1";  
const arg2 = "argument 2";  
tester(arg1, arg2);
```

HÀM (FUNCTION)

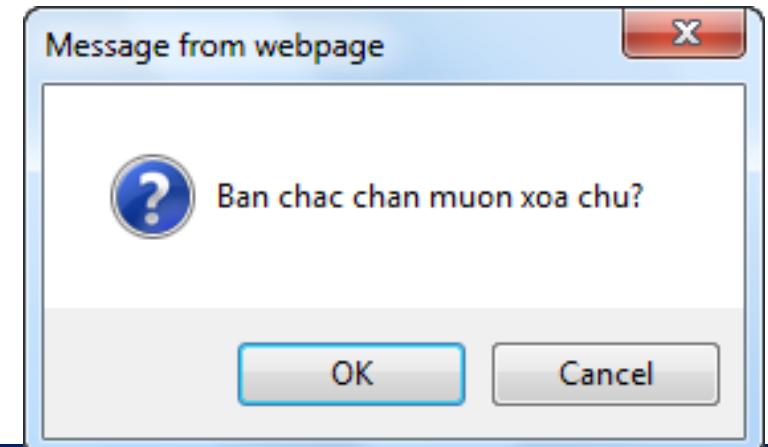
- Hàm trả về kết quả

```
function tester(para1, para2) {  
    return para1 + " " + para2;  
}  
  
const arg1 = "argument 1";  
const arg2 = "argument 2";  
let t = tester(arg1, arg2);
```

HÀM (FUNCTION)

- Confirm () : là hộp thoại nhận hồi đáp từ phía người dùng
 - Lời gọi hàm: confirm (<thông điệp>) ;
 - Hàm trả về hồi đáp của người dùng
 - Trả về True nếu người dùng nhấn vào OK
 - Trả về False nếu người dùng nhấn vào Cancel

```
var ok = confirm("Ban chac chan muon xoa chu?");  
if(ok == true){  
    doSomething();  
}  
else{  
    doAnythingElse();  
}
```



HÀM (FUNCTION)

- Ví dụ về: Confirm ()

```
function xacNhan(traloi) {  
    var ketQua = "";  
    if (traloi) {  
        ketQua = "Tuyet voi. Chuc ban chien thang!";  
    } else {  
        ketQua = "Hen gap lai ban nhe!";  
    }  
    return ketQua;  
}  
var traloi = confirm("Ban se choi game chu?");  
var thongbao = xacNhan(traloi);  
alert (thongbao);
```

PHẠM VI BIẾN

- Biến cục bộ (local variable)
 - Biến được khai báo trong hàm
 - Chỉ được tham chiếu đến trong phạm vi khai báo
- Biến toàn cục (global variable)
 - Biến được khai báo ngoài hàm
 - Có thể tham chiếu đến từ bất cứ đâu

PHẠM VI BIẾN

- Biến cục bộ (local variable)

```
function testAvailability() {  
    let y = "I'll return";  
    console.log("Available here:", y);  
    return y;  
}  
let z = testAvailability();  
console.log("Outside the function:", z);  
console.log("Not available here:", y);
```

PHẠM VI BIẾN

- Biến cục bộ (local variable) với let và var

```
function doingStuff() {  
  if (true) {  
    var x = "local";  
  }  
  console.log(x);  
}  
doingStuff();  
//result: local
```

```
function doingStuff() {  
  if (true) {  
    let x = "local";  
  }  
  console.log(x);  
}  
doingStuff();  
//result: ReferenceError: x is not  
defined
```

PHẠM VI BIẾN

- Biến toàn cục (global variable)

```
let globalVar = "Accessible everywhere!";
console.log("Outside function:", globalVar);
function creatingNewScope(x) {
    console.log("Access to global vars inside function." ,
    globalVar);
}
creatingNewScope("some parameter");
console.log("Still available:", globalVar);

//Outside function: Accessible everywhere!
//Access to global vars inside function. Accessible everywhere!
//Still available: Accessible everywhere!
```

XỬ LÝ SỰ KIỆN (EVENTS)

- Events là những thứ/những điều xảy ra trên một trang web
- Ví dụ: nhấp chuột (click) vào cái gì đó, di chuyển chuột qua (mouse over) một phần tử (element),...
- Tất cả các element trên trang web đều có một tập các sự kiện tương ứng.
- ***Một phần tử chỉ có 1 trình xử lý sự kiện (event handler) làm thuộc tính. Nếu phần tử có xử lý sự kiện onclick thì không thể có onmouseover***

XỬ LÝ SỰ KIỆN (EVENTS)

- Một số sự kiện
 - onClick: được kích hoạt khi nhấn chuột vào một element
 - onload và onUnload: được kích hoạt khi người dùng vào hoặc thoát khỏi trang web
- Click vào bất kỳ đâu trên trang web

```
<!DOCTYPE html>
<html>
<head>
  <title>Page Title</title>
</head>
<body onclick="alert('Hi Event!')"></body>
</html>
```

Sự kiện

Xử lý sự kiện



XỬ LÝ SỰ KIỆN (EVENTS)

- Có thể thêm nhiều dòng lệnh

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body onclick="alert('Hi Event!');alert('Hello
Event');"></body>
</html>
```

Sự kiện

Xử lý sự kiện

Trong trường hợp xử lý phức tạp cho sự kiện????

XỬ LÝ SỰ KIỆN (EVENTS)

- Sử dụng hàm để thực hiện các xử lý cho sự kiện

```
<!DOCTYPE html>
<html>
<head>
    <title>Page Title</title>
    <script src="chapter3.js" type="text/javascript"></script>
    <script type="text/javascript">
        function sayHi(){
            alert("Hi Event");
            alert("Hello Event");
        }
    </script>
</head>
<body>

    <button onclick="sayHi()">Add a number</button>

</body>
</html>
```

XỬ LÝ SỰ KIỆN (EVENTS)

- Sự kiện phức tạp



XỬ LÝ SỰ KIỆN (EVENTS)

```
<!DOCTYPE html>
<html>
<head>
<script type="text/javascript">
function hienThiAnh(dovat) {
    if (dovat == "game") {
        document.write("<img src = 'game.png'>");
    } else {
        document.write("<img src = 'football.jpg'>");
    }
}
</script>
</head>
<body>
    <p> Bạn thích chơi:</p>
    <input type="button" value="Game" onclick="hienThiAnh('game');"/>
    <input type="button" value="Đá banh" onclick="hienThiAnh('football');"/>
</body>
</html>
```

3.2.

HƯỚNG ĐỐI TƯỢNG TRONG JS

- 1. HƯỚNG ĐỐI TƯỢNG**
- 2. BOM**

LẬP TRÌNH HƯỚNG ĐỐI TƯỢNG

- Đối tượng (object): được sử dụng để mô tả các đối tượng ngoài đời thực (các đồ vật, sự vật)
 - Ví dụ đối tượng: quả bóng, cái bàn, ô tô, bông hoa, con người, nhà máy,...
- Mỗi đối tượng có đặc tính và hành động riêng
- Ý tưởng chủ đạo của phương thức lập trình hướng đối tượng: mô phỏng cuộc sống thực trong lập trình
 - Trong cuộc sống có những đối tượng như quả bóng, cái bàn...với các đặc tính và hành động riêng thì lập trình mô phỏng các đối tượng đó với các đặc tính và hành động như thế

ĐỐI TƯỢNG (OBJECT)

- Trong lập trình: đặc tính được gọi là thuộc tính, hành động được gọi là phương thức

**Đặc tính:**

Cân nặng: 2,4kg
Màu lông: nâu
Tuổi: 3
Giống: Chihuahua

Phương thức: Sữa, Cắn

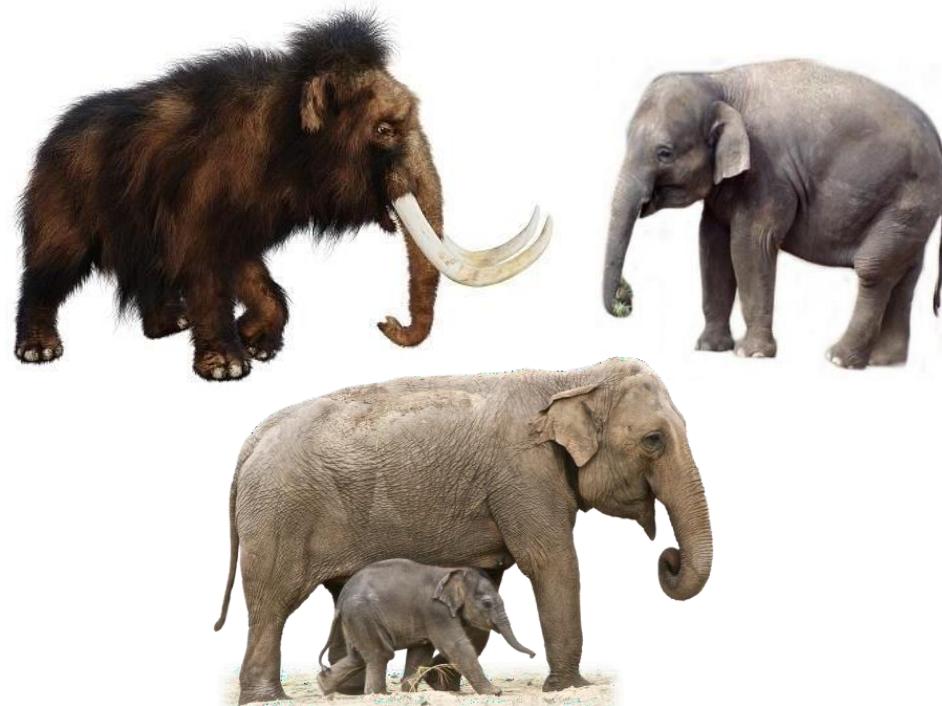
**Đặc tính:**

Cân nặng: 2 tấn
Màu da: nâu
Vòi: 1m

Phương thức: Phun nước, Ăn cỏ

Lớp (Class)

- Các đối tượng có cùng thuộc tính và phương thức được gom lại thành 1 lớp (Class)
- Hay: Lớp (Class) định nghĩa tập hợp các đối tượng có cùng thuộc tính và phương thức



ĐỐI TƯỢNG (OBJECT)

- Tạo đối tượng: sử dụng cặp ngoặc nhọn { }
- Ví dụ

```
let dog = {  
    dogName: "JavaScript",  
    weight: 2.4,  
    color: "brown",  
    breed: "chihuahua",  
    age: 3  
};
```

ĐỐI TƯỢNG (OBJECT)

- Truy xuất thuộc tính của đối tượng

```
let dogColor = dog["color"];
```

- hoặc

```
let dogColor = dog.color;
```

- Cập nhật giá trị của thuộc tính của đối tượng

```
dog["color"] = "black";
```

```
dog.weight = 2.3;
```

ĐỐI TƯỢNG (OBJECT)

- Phương thức

```
let dog = {  
    dogName: "JavaScript",  
    weight: 2.4,  
    color: "brown",  
    breed: "chihuahua",  
    age: 3,  
    sua: function () { //tenphuongthuc: function () {}  
        return "Gau Gau";  
    }  
};
```

- Truy cập phương thức

```
dog.sua() //tendoituong.tenphuongthuc()
```

ĐỐI TƯỢNG (OBJECT)

- Vấn đề nảy sinh

```
let dogtype1 = {  
    weight: 2.4,  
    color: "white",  
    breed: "chihuahua",  
    type: "short hair",  
};
```

```
let dogtype2 = {  
    weight: 2.4,  
    color: "white",  
    breed: "chihuahua",  
    type: "long hair",  
};
```

```
let dogtype3 = {  
    weight: 2.3,  
    color: "brown",  
    breed: "chihuahua",  
    type: "short hair",  
};
```

```
};
```

```
let dogtype4 = {  
    weight: 2.3,  
    color: "black and white",  
    breed: "chihuahua",  
    type: "short hair",  
};
```

ĐỐI TƯỢNG (OBJECT)

- Vân đề nảy sinh

```
let dogtype1 = {  
    weight: 2.4,  
    color: "white",  
    breed: "chihuahua",  
    type: "short hair",  
};
```



```
let dogtype2 = {  
    weight: 2.4,  
    color: "white",  
    breed: "chihuahua",  
    type: "long hair",  
};
```

```
let dogtype3 = {  
    weight: 2.3, color:  
    "brown",  
    breed: "chihuahua", type:  
    "short hair",  
};
```

```
let dogtype4 = {  
    weight: 2.3,  
    color: "black and white",  
    breed: "chihuahua",  
    type: "short hair",  
};
```

- => Tạo một khuôn mẫu chung (lớp)

LỚP (CLASS)

- Tạo object bằng cách sử dụng **special function** kết hợp với từ khoá **this**

```
function Dog(dogName, weight, color, breed, type) {  
    this.dogName = dogName;  
    this.weight = weight;  
    this.color = color;  
    this.breed = breed;  
    this.type = type;  
    this.sua = function () {  
        return "Gau Gau";  
    };  
}  
let dog1 = new Dog("Jacky", 0.8, "brown", "chihuahua teacup", "short hair");  
let dog2 = new Dog("Javascript", 1, "brown", "chihuahua mini", "long hair");
```

OBJECT VÀ ARRAY

- Mảng các object (objects in arrays)

```
Dog[0] = new Dog("Jacky", 0.8, "brown", "chihuahua teacup", "short hair" );
Dog[1] = new Dog("Javascript", 1, "brown", "chihuahua mini", "long hair" );
Dog[2] = new Dog("Jscript", 1, "white", "chihuahua", "long hair" );
Dog[3] = new Dog("Jscript", 1, "black & white", "chihuahua", "short hair" );
```

- Truy xuất đến thuộc tính và phương thức của đối tượng trong mảng đối tượng

```
Dog [1] .sua ()
Dog [1] .dogName ;
```

- Lặp qua các đối tượng (dùng for in)

```
for (var x in Dog) {
    console.log(Dog [x] .dogName) ;
}
```

OBJECT VÀ ARRAY

- Mảng trong object (arrays in objects)

```
company = {  
    companyName: "Healthy Candy",  
    activities: ["food manufacturing",  
                "improving kids' health",  
                "manufacturing toys"],  
    yearOfEstablishment: 2021  
};
```

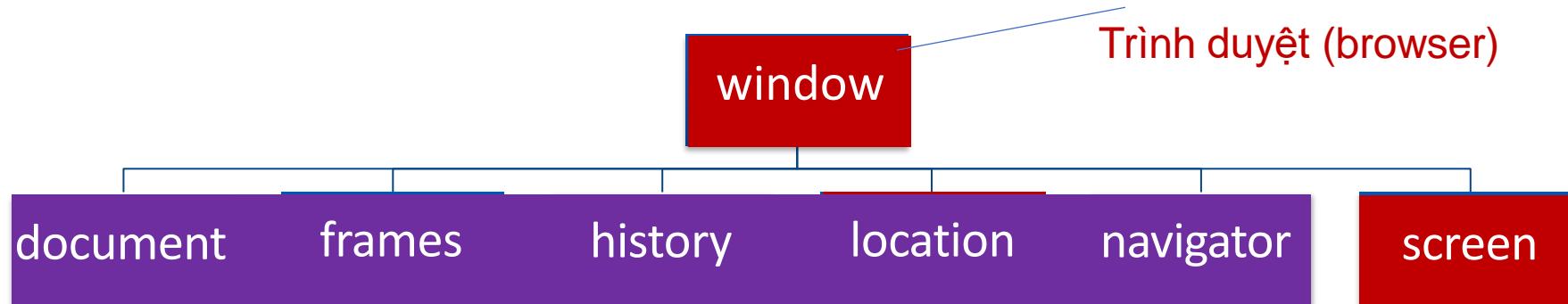
- Truy xuất

```
let activity = company.activities[1];
```

2. BROWSER OBJECT MODEL (BOM)

BROWSER OBJECT MODEL (BOM)

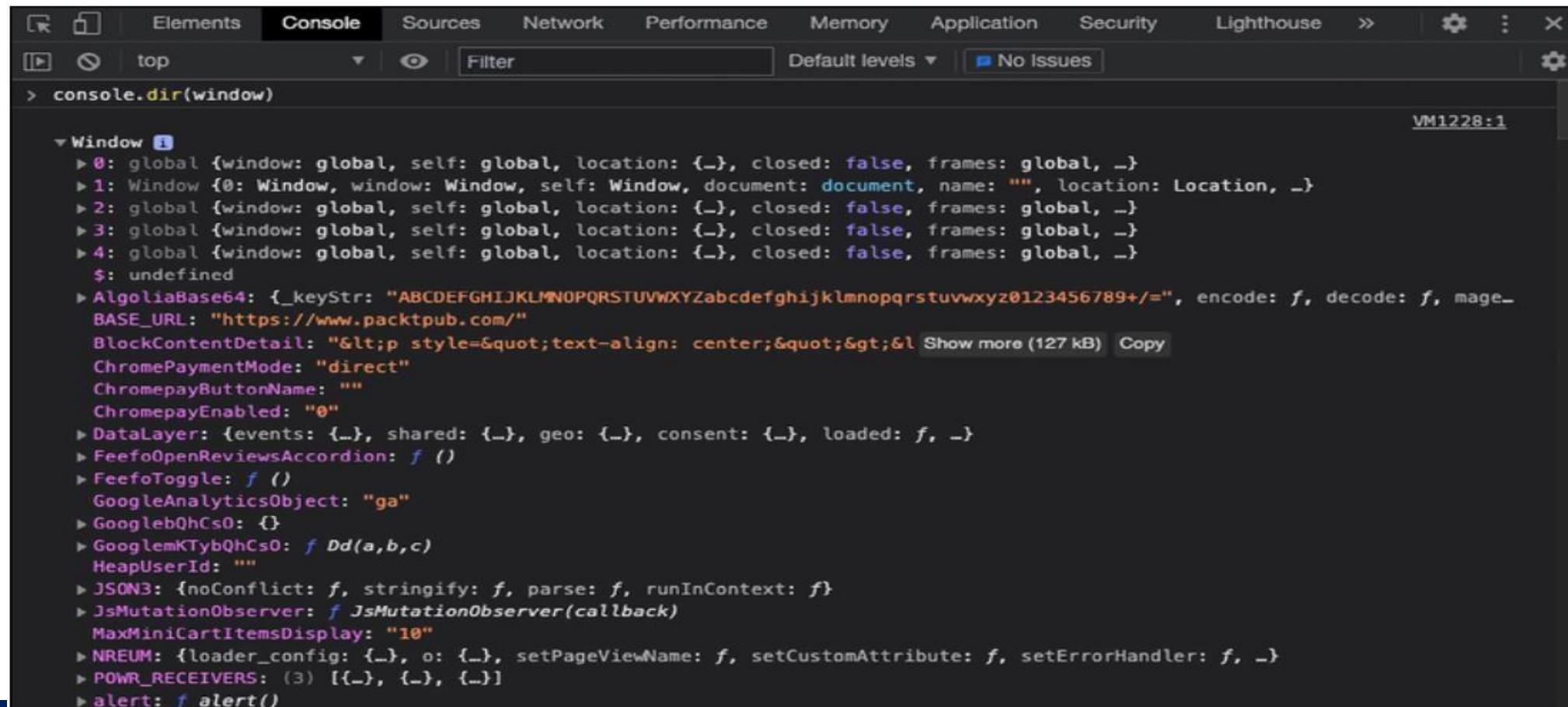
- BOM là một hệ thống phân cấp hình cây gồm các đối tượng



- Các đối tượng cung cấp thuộc tính và phương thức cho lập trình viên Javascript
- Đối với mỗi đối tượng, mỗi trình duyệt hỗ trợ các thuộc tính và phương thức khác nhau
 - Hiểu môi trường mà trình duyệt cung cấp để viết mã Javascript chạy ổn định trên nhiều trình duyệt

BROWSER OBJECT MODEL (BOM)

- `Console.dir()`: hiển thị danh sách tất cả các thuộc tính của đối tượng đặc biệt



The screenshot shows the Chrome DevTools interface with the "Console" tab selected. In the main pane, the command `> console.dir(window)` is entered at the top. Below it, the output of the `dir` method is displayed as a hierarchical tree of properties for the `window` object. The tree includes properties like `0`, `1`, `global`, `AlgoliaBase64`, `DataLayer`, `FeefoOpenReviewsAccordion`, `FeefoToggle`, `GoogleAnalyticsObject`, `GooglebQhCs0`, `GooglemKTybQhCs0`, `JSON3`, `JsMutationObserver`, `NREUM`, `POWR_RECEIVERS`, and `alert`. Some properties like `AlgoliaBase64` have a "Show more" link and a "Copy" button.

```
> console.dir(window)
VM1228:1
Window
▶ 0: global {window: global, self: global, location: {}, closed: false, frames: global, ...}
▶ 1: Window {0: Window, window: Window, self: Window, document: document, name: "", location: Location, ...}
▶ 2: global {window: global, self: global, location: {}, closed: false, frames: global, ...}
▶ 3: global {window: global, self: global, location: {}, closed: false, frames: global, ...}
▶ 4: global {window: global, self: global, location: {}, closed: false, frames: global, ...}
$: undefined
▶ AlgoliaBase64: {_keyStr: "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789+/", encode: f, decode: f, mage_
  BASE_URL: "https://www.packtpub.com/"
  BlockContentDetail: "<p style="text-align: center;">" Show more (127 kB) Copy
  ChromePaymentMode: "direct"
  ChromepayButtonName: ""
  ChromepayEnabled: "0"
  DataLayer: {events: {}, shared: {}, geo: {}, consent: {}, loaded: f, ...}
  FeefoOpenReviewsAccordion: f()
  FeefoToggle: f()
  GoogleAnalyticsObject: "ga"
  GooglebQhCs0: {}
  GooglemKTybQhCs0: f Dd(a,b,c)
  HeapUserId: ""
  JSON3: {noConflict: f, stringify: f, parse: f, runInContext: f}
  JsMutationObserver: f JsMutationObserver(callback)
  MaxMiniCartItemsDisplay: "10"
  NREUM: {loader_config: {}, o: {}, setPageViewName: f, setCustomAttribute: f, setErrorHandler: f, ...}
  POWR_RECEIVERS: (3) [{}, {}, {}]
  alert: f alert()
```

ĐỐI TƯỢNG WINDOW

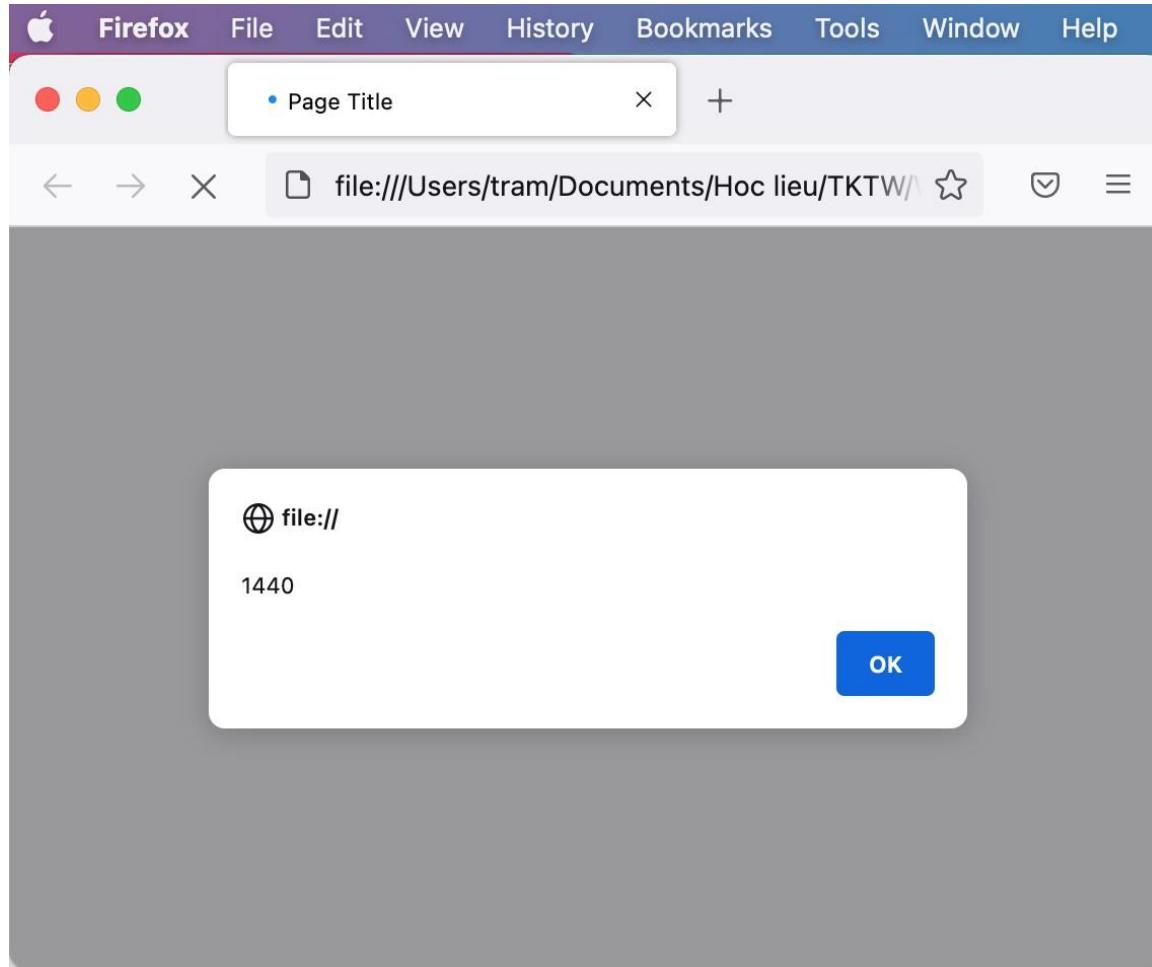
- Window là đối tượng thể hiện cửa sổ hiển thị hiện tại của trình duyệt
- Một số phương thức của đối tượng window đã được sử dụng: alert(), prompt(), confirm()
- Các thuộc tính và phương thức của window có thể gọi trực tiếp hoặc thông qua window

```
alert("Hello");  
  
window.alert("Hello"  
);
```

CÁC THUỘC TÍNH CỦA WINDOW

Thuộc tính	Giải thích
closed	Có giá trị là True khi cửa sổ được đóng
defaultStatus	Thiết lập văn bản mặc định trên thanh trạng thái của trình duyệt
name	Thiết lập hoặc trả về tên của cửa sổ
opener	Tham chiếu đến cửa sổ tạo ra cửa sổ hiện tại
status	Thông tin xuất hiện trên thanh trạng thái
innerHeight	Thiết lập hoặc trả về chiều cao phần nội dung của cửa sổ
document	Trả về đối tượng document của cửa sổ

CÁC THUỘC TÍNH CỦA WINDOW



```
alert(window.innerWidth);
```

CÁC PHƯƠNG THỨC CỦA WINDOW

Phương thức	Giải thích
focus ()	Chuyển focus đến cửa sổ
blur ()	Bỏ focus đến cửa sổ
close ()	Đóng cửa sổ
open ()	Mở cửa sổ
print ()	Thực hiện chức năng in
moveTo ()	Sử dụng để chuyển cửa sổ về vị trí xác định
resizeTo ()	Thay đổi kích thước cửa cửa sổ về vị trí xác định

CÁC PHƯƠNG THỨC CỦA WINDOW

- Window.open: sử dụng để mở một cửa sổ từ cửa sổ hiện thời
window.open(url, ten, dactinh)

- url: url của trang web
- ten: tên của cửa sổ sẽ mở
- dactinh: các đặc tính mà cửa sổ được mở sẽ có (mỗi trình duyệt sẽ hỗ trợ một tập các đặc tính riêng)

```
window.open("http://www.google.com.vn/", "timkiem", "menubar = yes, width = 800, height = 600")
```

Chú ý:

- Chỉ nên sử dụng cách này khi thật cần thiết vì trình duyệt có thể bị disable javascript
- Có thể sử dụng thẻ <a> để thay thế

WINDOW SCREEN OBJECT (ĐỐI TƯỢNG SCREEN)

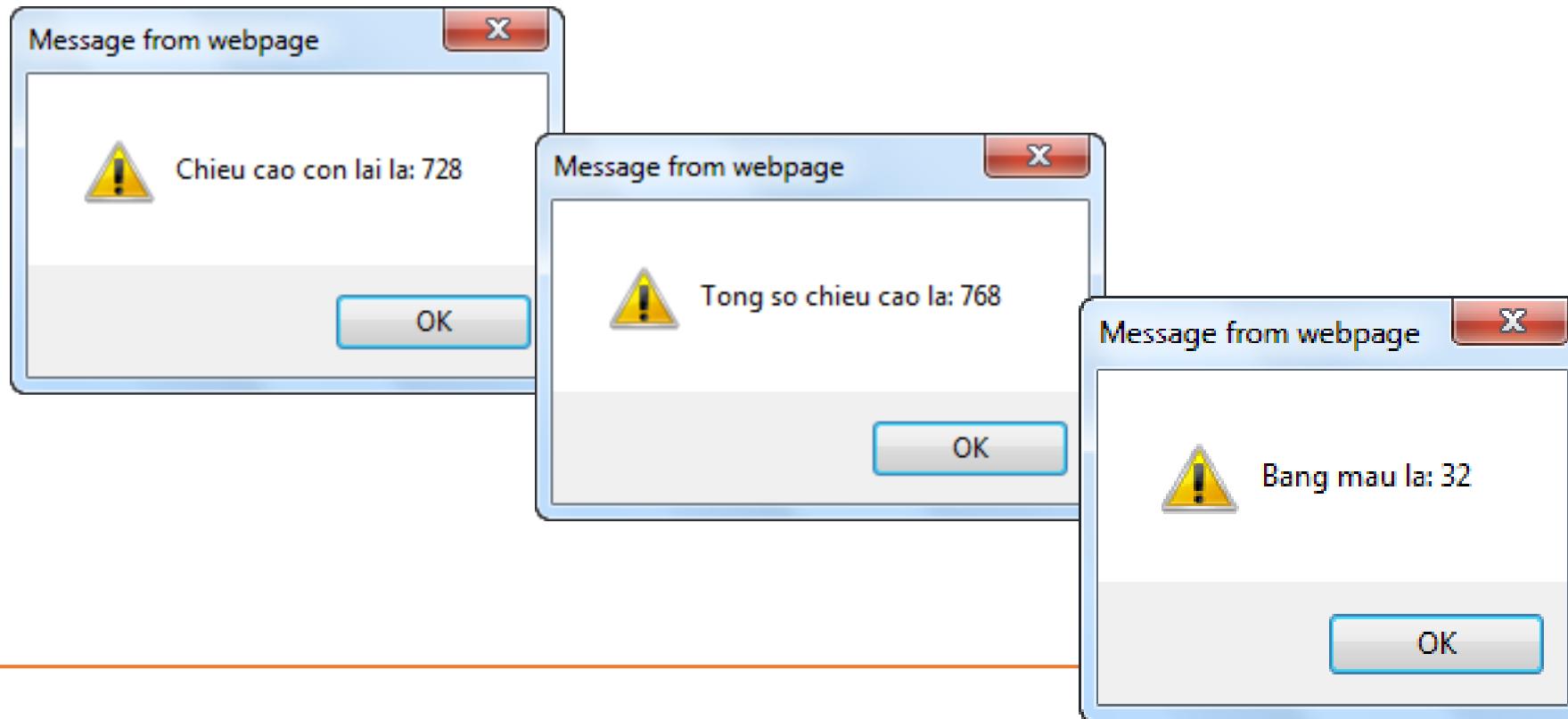
- Mỗi người truy cập sử dụng màn hình có độ phân giải khác nhau, kích thước khác nhau, dải màu khác nhau...
→ Người lập trình phải nắm được thông tin này để hiển thị ảnh phù hợp, hiển thị trang web có kích thước phù hợp...
- Đối tượng screen cung cấp thuộc tính để lấy thông tin về màn hình của người truy cập

WINDOW SCREEN OBJECT (ĐỐI TƯỢNG SCREEN)

Thuộc tính	Giải thích
availHeight	Trả về chiều dài của màn hình (trừ kích thước của window taskbar)
availWidth	Trả về chiều rộng của màn hình (trừ kích thước của window taskbar)
height	Trả về chiều dài của màn hình
width	Trả về chiều rộng của màn hình
pixelDepth	Trả về độ phân giải của màn hình
colorDepth	Trả về bảng màu để hiển thị ảnh

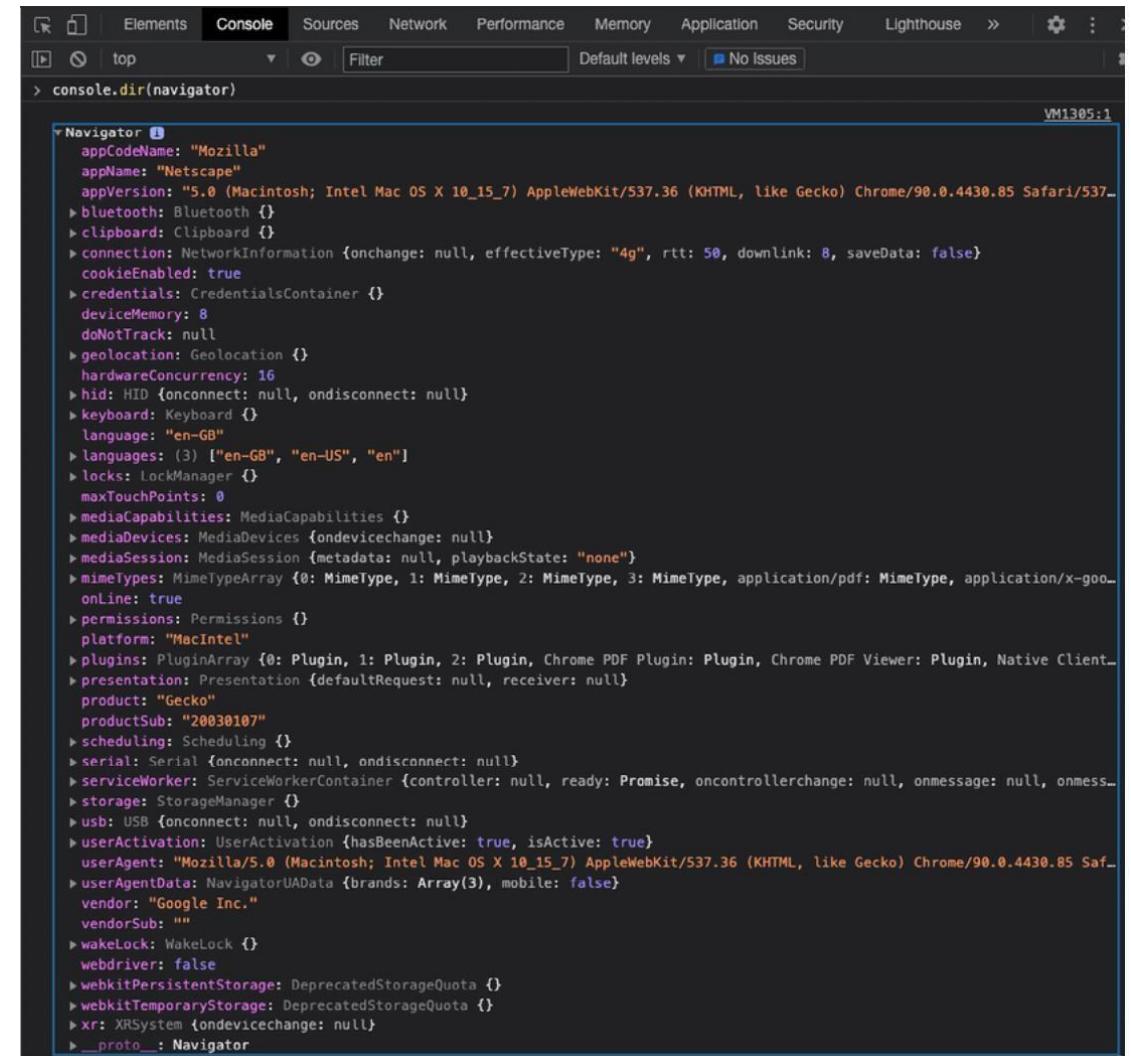
DEMO

```
alert("Chieu cao con lai la: " + screen.availHeight);  
alert("Tong so chieu cao la: " + screen.height);  
alert("Bang mau la: " + screen.colorDepth);
```



WINDOW NAVIGATOR OBJECT (ĐỐI TƯỢNG NAVIGATOR)

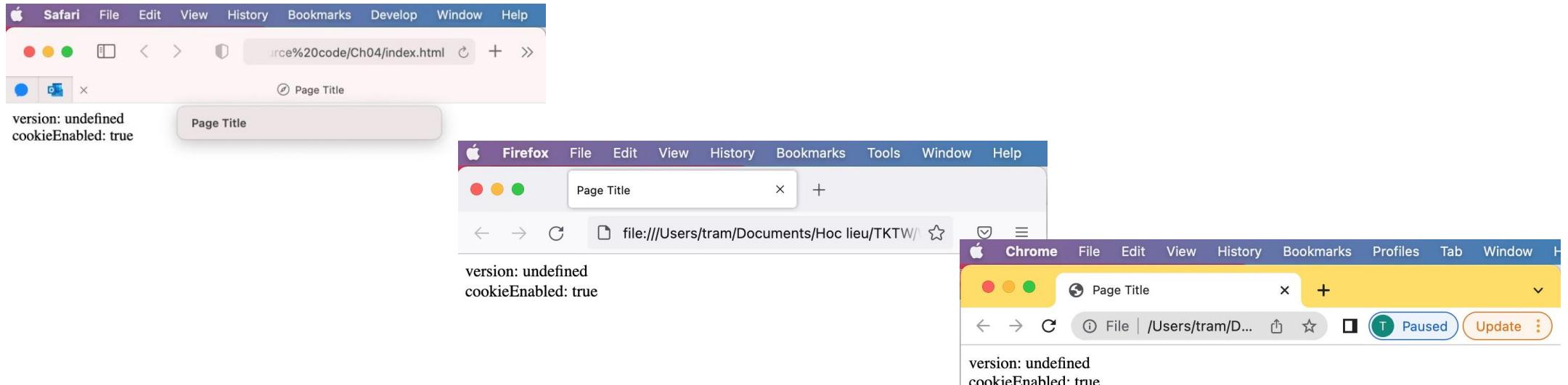
- Mỗi trình duyệt có cách thức thi hành mã Javascript riêng
- Có thể cùng thực hiện một chức năng, nhưng đối với từng trình duyệt, cần phải viết các đoạn mã khác nhau.
⇒ Cần biết thông tin về trình duyệt để viết mã Javascript phù hợp
- Đối tượng Navigator cung cấp các thông tin về trình duyệt đang sử dụng (version, hệ điều hành mà trình duyệt đang chạy)



console.dir(window.navigator); HOẶC console.dir(navigator);

DEMO

```
document.write("version: " + navigator.version + "<br>");  
document.write("cookieEnabled: " + navigator.cookieEnabled);
```

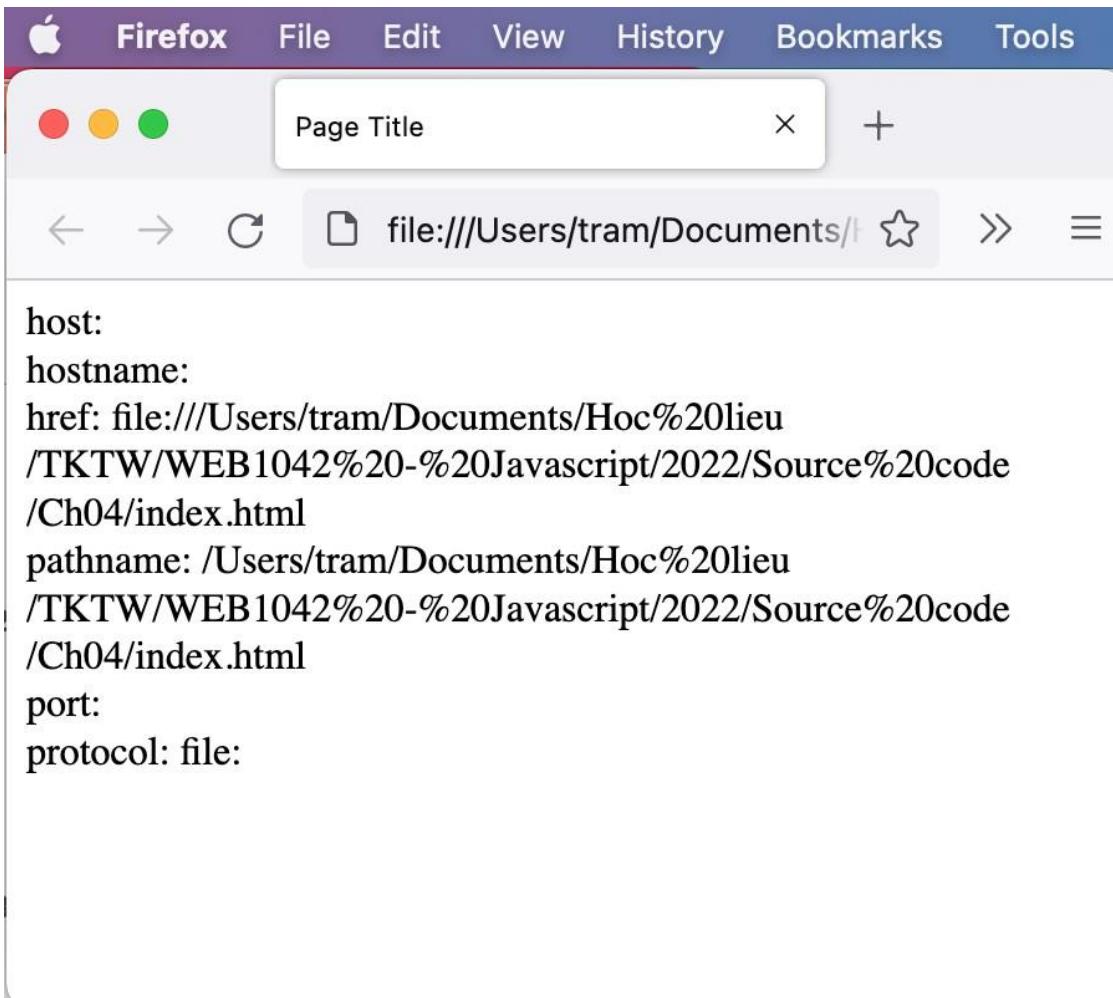


WINDOW LOCATION OBJECT (ĐỐI TƯỢNG LOCATION)

- Quản lý thông tin về URL hiện tại

Thuộc tính	Giải thích
host	Trả về tên host và cổng của URL
hostname	Trả về tên host
href	Trả về toàn bộ URL
pathname	Trả về tên đường dẫn của URL
port	Trả về cổng mà server sử dụng cho URL
protocol	Trả về protocol của URL
Phương thức	Giải thích
assign ()	Load document mới
reload ()	Load lại document hiện tại

DEMO



```
document.write("host: " + location.host +  
"<br>");  
document.write("hostname: " +  
location.hostname + "<br>");  
document.write("href: " + location.href +  
"<br>");  
document.write("pathname: " +  
location.pathname + "<br>");  
document.write("port: " + location.port +  
"<br>");  
document.write("protocol: " +  
location.protocol + "<br>");
```

WINDOW HISTORY OBJECT (ĐỐI TƯỢNG HISTORY)

- Chứa thông tin về các URL được người dùng truy cập

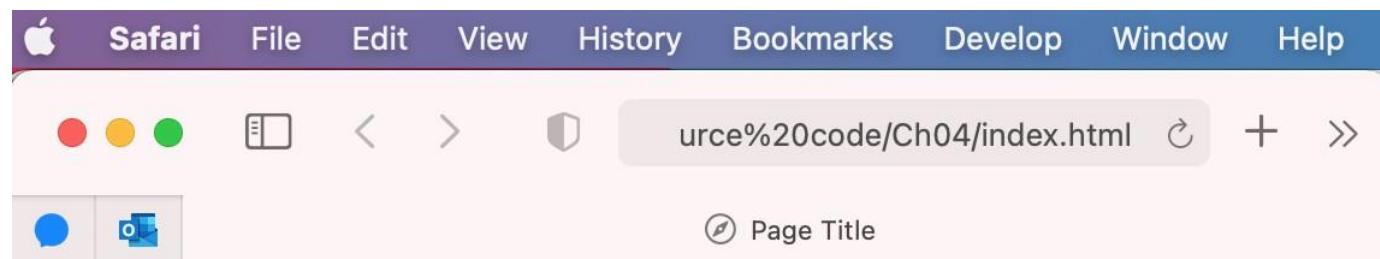
Thuộc tính	Giải thích
length	Trả về số lượng URL trong danh sách History
Phương thức	Giải thích
back ()	Load URL trước đó trong danh sách History
forward ()	Load URL sau đó trong danh sách History
go ()	Load URL cụ thể từ History

DEMO

- Định nghĩa hàm trong thẻ Javascript

```
function goBack() {  
    history.back();  
}
```

```
function goNext() {  
    history.forward();  
}
```



- Gọi hàm

```
<p><a href = "#"  
onclick="goBack()">Back</a></p>  
<p><a href = "#"  
onclick="goNext()">Next</a></p>
```

TIMER

- Javascript cung cấp các phương thức để xử lý sự kiện thời gian
- Các phương thức này thuộc đối tượng window
- Một số phương thức quan trọng

Phương thức	Giải thích
setTimeout	Thực hiện công việc sau một khoảng thời gian trong tương lai
clearTimeout	Hủy bỏ setTimeout trước đó
setInterval	Thực hiện lặp lại công việc sau một khoảng thời gian
clearInterval	Hủy bỏ setInterval

TIMER

- Cú pháp

```
var t=setTimeout("Lệnh_javascript",số_mili_giây );
```

- Lệnh_javascript: mã thực thi hoặc lời gọi hàm
- Mili_giây: sau thời gian này mã sẽ được thực hiện
- setTimeout() trả về giá trị, giá trị được lưu trong biến t. Nếu muốn huỷ bỏ setTimeout, sử dụng hàm clearTimeout và truyền vào đối số t
- Cú pháp tương tự đối với setInterval()

DEMO setTimeout

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Page Title</title>
    <script type="text/javascript">
        function onClickEvent() {
            var t = setTimeout("alert('Hi')", 1000);
        }
    </script>
</head>
<body>
    <button onclick="onClickEvent();"> Click here!</button>
</body>
</html>
```

DEMO setInterval

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <title>Page Title</title>
    <script type="text/javascript">
        function onClickEvent() {
            var t = setInterval("alert('Hi')", 1000);
        }
    </script>
</head>
<body>
    <button onclick="onClickEvent();"> Click here!</button>
</body>
</html>
```

TỔNG KẾT BOM

- Có rất nhiều phương thức lập trình. Mỗi phương thức phù hợp cho một mục đích riêng. Phương thức lập trình hướng đối tượng được phát triển rộng rãi nhất.
- Mỗi đối tượng có các thuộc tính và phương thức riêng
- Các đối tượng có các thuộc tính và phương thức giống nhau thuộc cùng một lớp
- Browser Object Model (BOM) là tập hợp các đối tượng được xây dựng sẵn giúp lập trình viên thao tác với trình duyệt

TỔNG KẾT

- Trình duyệt được biểu diễn bằng đối tượng window
- Đối tượng window có các đối tượng con là document, frames, history, location, navigator, screen
- Đối tượng document đại diện cho nội dung trang web
- Đối tượng history chứa thông tin về các url được người dùng truy cập
- Đối tượng location chứa thông tin về url hiện tại
- Đối tượng navigator chứa thông tin về trình duyệt
- Đối tượng screen chứa thông tin về màn hình

3.3 DOCUMENT OBJECT MODEL (DOM)

3.3.1 DOM CƠ BẢN

**3.3.2 TRUY XUẤT CÁC
ELEMENT TRONG DOM**

3.3.1 DOM CƠ BẢN

DOCUMENT OBJECT MODEL

- DOM là một chuẩn được định nghĩa bởi **W3C** (World Wide Web Consortium) để có thể **truy cập và thao tác** với các tài liệu như **html** hay **xml** bằng các **ngôn ngữ lập trình** như Javascript, VB...

The Document Object Model is a platform- and language-neutral interface that will allow programs and scripts to dynamically access and update the content, structure and style of documents (Định nghĩa bởi W3C)

DOM

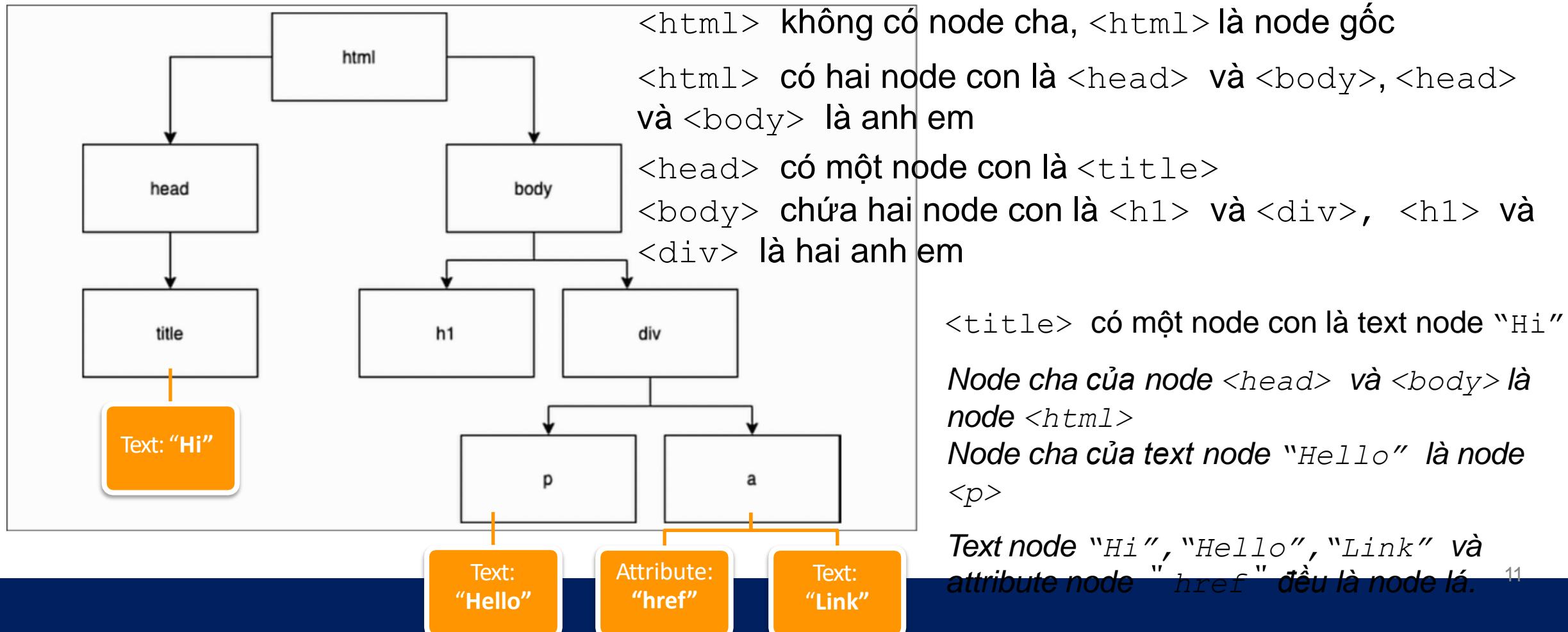
- DOM được chia thành 3 mức
 - Core DOM: Tiêu chuẩn cho bất kỳ tài liệu có cấu trúc nào
 - XML DOM: Tiêu chuẩn cho tài liệu XML
 - HTML DOM: Tiêu chuẩn cho tài liệu HTML
- HTML DOM định nghĩa các **đối tượng** và **thuộc tính** của tất cả các thành phần HTML và **phương thức** để truy cập đến chúng.
- Hay **HTML DOM là chuẩn để lấy, thay đổi, thêm, xoá các thành phần HTML**

NODE TRONG DOM

- Tất cả các thành phần trong tài liệu HTML đều được biểu diễn bằng đối tượng node
 - Toàn bộ tài liệu là **document node**
 - Tất cả các thành phần của HTML đều là **element node**
 - Văn bản trong thành phần HTML là **text node**
 - Tất cả các thuộc tính là **attribute node**
 - Chú thích là **comment node**

CẤU TRÚC HÌNH CÂY DOM

- DOM trình bày tài liệu HTML theo cấu trúc hình cây

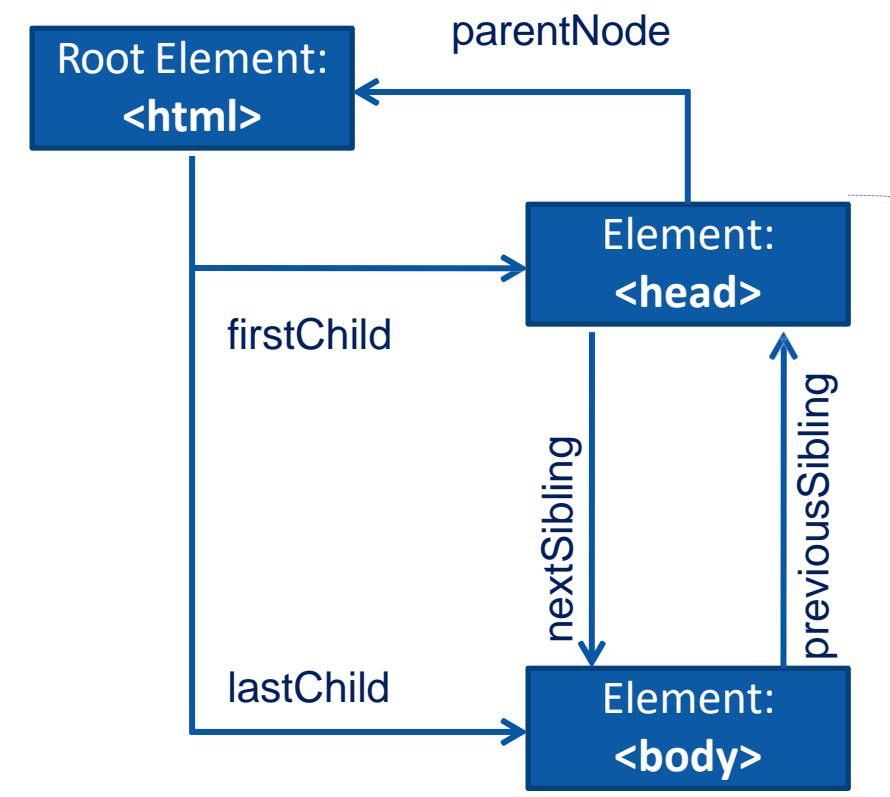


QUAN HỆ GIỮA CÁC NODE

- Giữa các node có mối quan hệ **cha, con** (parent, children) và **anh em** (siblings)
 - Node cha có các node con, các node con cùng cấp bậc gọi là anh em
- Trên cây, node đầu tiên được gọi là **root** (gốc)
- Tất cả các node, ngoại trừ root **chỉ có một node cha**
- Một node có thể có **nhiều node con** hoặc **không có node con nào**
- **Node lá** (leaf) là node **không có node con**
- **Những node anh em** (siblings) với nhau là node có cùng một node cha

CON CẢ, CON ÚT

- <head> và <body> là anh em, trong đó <head> là con cả của <html> còn <body> là con út của <html>
- <h1> và <div> là con cả của <body>, trong đó <h1> là con cả của <body>, còn <div> là con út của <body>



3.3.2 TRUY XUẤT CÁC ELEMENT TRONG DOM

THUỘC TÍNH VÀ PHƯƠNG THỨC CỦA NODE

- DOM định nghĩa các thuộc tính và các phương thức cho các node để hỗ trợ cho việc lập trình
- Thuộc tính định nghĩa các đặc tính cho node
- Phương thức để thực hiện các thao tác với node
 - Truy cập đến node
 - Thêm node con cho node
 - Xoá node con

Nhắc lại về CSS

- CSS định nghĩa màu sắc, font, layout...cho trang web
- CSS bao gồm một tập các thuộc tính, mỗi thuộc tính có một tập giá trị nhất định
 - Font-family: arial
- Selector cho biết thành phần nào sẽ được áp dụng
 - Selector { property:value; }
- Có 3 loại selector
 - Thẻ h1{font-family:arial;}
 - Lớp .tenClass{font-family:arial;}
 - ID .tenId{font-family:arial;}

CÁC THUỘC TÍNH CỦA NODE

Thuộc tính	Giải thích
<i>x là đối tượng node</i>	
<code>x.innerHTML</code>	Giá trị văn bản của x
<code>x.nodeName</code>	Tên của x
<code>x.nodeValue</code>	Giá trị của x
<code>x.nodeType</code>	Kiểu của Node
<code>x.parentNode</code>	Node cha của x
<code>x.childNodes</code>	Các node con của x
<code>x.attributes</code>	Các node thuộc tính của x

TRUY XUẤT CÁC ELEMENT TRONG DOM

- Duyệt DOM bằng cách sử dụng document (document chứa tất cả các HTML và là đại diện cho trang web)
- Có thể truy cập đến các element (node) bằng cách sau
 - Truy xuất các element (node) **bằng ID**: `getElementById()`
 - Truy xuất các element (node) **bằng tên thẻ (Tag name)**:
`getElementsByName()`
 - Truy xuất các element (node) **bằng tên class (Class name)**:
`getElementsByClassName()`
 - Truy xuất các element (node) với **CSS selector**: `querySelector()`,
`querySelectorAll()`

TRUY XUẤT CÁC ELEMENT TRONG DOM

- Truy xuất các element (node) **bằng ID**: getElementById()

```
<html>
  <body>
    <h1 style="color: pink;">Just an example</h1>
    <div id="one" class="example">Hi!</div>
    <div id="two" class="example">Hi!</div>
    <div id="three" class="something">Hi!</div>
  </body>
  <script>
    console.log(document.getElementById("two"));
  </script>
</html>
```

TRUY XUẤT CÁC ELEMENT TRONG DOM

- Truy xuất các element (node) với **Tag name**: getElementByTagName ()

```
<html>
  <body>
    <h1 style="color: □pink;">Just an example</h1>
    <div id="one" class="example">Hi!</div>
    <div id="two" class="example">Hi!</div>
    <div id="three" class="something">Hi!</div>
  </body>
  <script>
    console.log(document.getElementsByTagName("div").item(1));
    console.log(document.getElementsByTagName("div").namedItem("one"));
  </script>
</html>
```

TRUY XUẤT CÁC ELEMENT TRONG DOM

- Truy xuất các element (node) với **class name**: getElementByClassName()

```
<html>
  <body>
    <h1 style="color: pink;">Just an example</h1>
    <div id="one" class="example">Hi!</div>
    <div id="two" class="example">Hi!</div>
    <div id="three" class="something">Hi!</div>
  </body>
  <script>
    console.log(document.getElementsByClassName("example"));
  </script>
</html>
```

Truy xuất các element trong DOM

- Truy xuất các element (node) với **CSS selector**

```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
    <div id="hoa">
      <p>Hoa Hong</p>
      <p>Hoa Lan</p>
      <p>Hoa Dao</p>

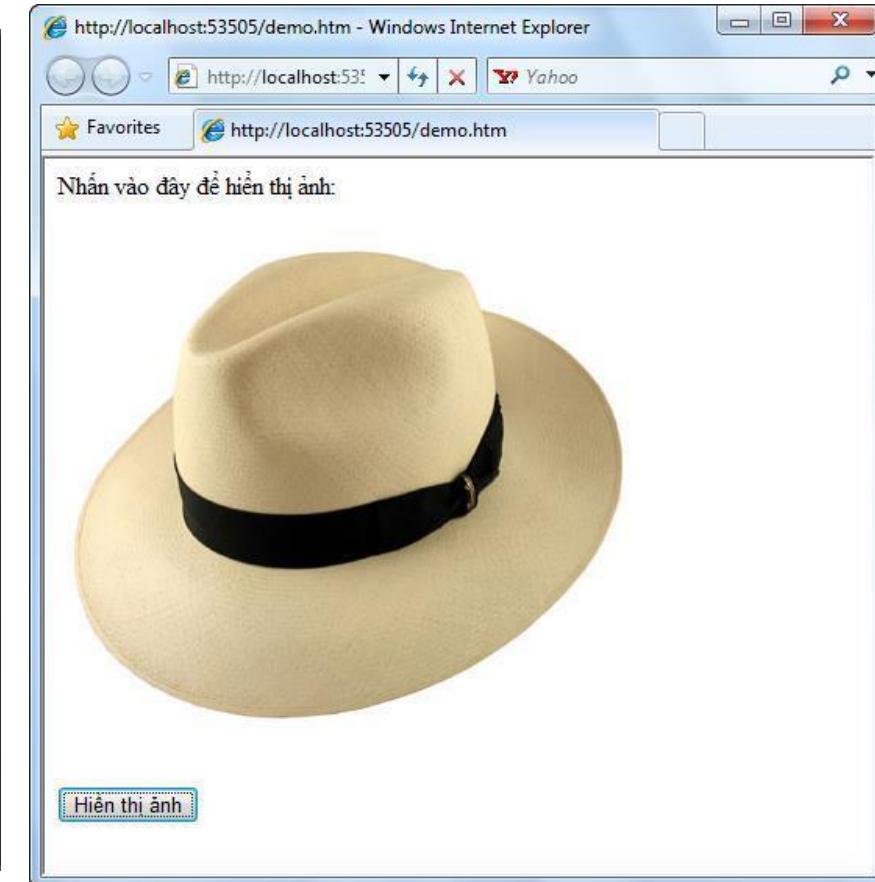
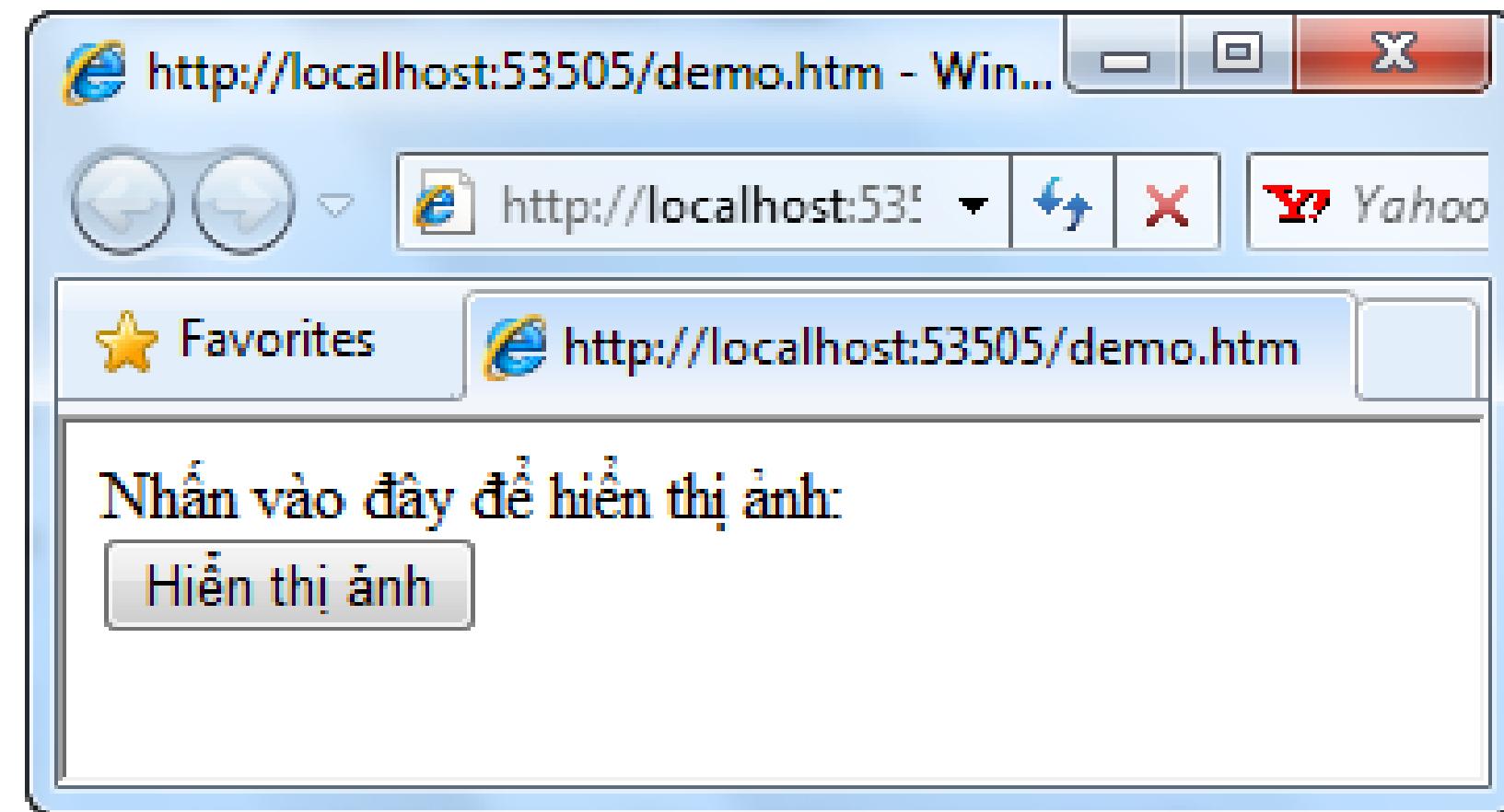
      <div class="example">Hi</div>
      <div class="example">Hello</div>
    </div>
    <div id="three" class="something">Hi!</div>
    <script type="text/javascript">
      console.log(document.querySelector("div"));
      console.log(document.querySelector(".something"));
      console.log(document.querySelectorAll("div.example"));
    </script>
  </body>
</html>
```

DEMO THUỘC TÍNH INNERHTML

- Demo thuộc tính **innerHTML**

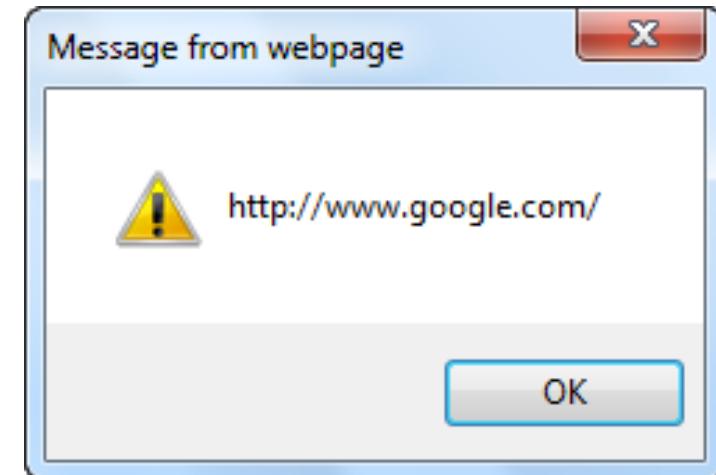
```
<html>
  <head>
    <script type="text/javascript">
      function hienThi() {
        var node = document.getElementById("anhDiv");
        node.innerHTML = "<img src='mu.jpg'>";
      }
    </script>
  </head>
  <body>
    Nhấn vào đây để hiển thị ảnh:
    <div id="anhDiv"></div>
    <input type="button" value="Hiển thị ảnh" onclick="hienThi()" />
  </body>
</html>
```

DEMO THUỘC TÍNH INNERHTML



DEMO LẤY ATTRIBUTE CỦA NODE (ELEMENT)

```
<html>
    <head>
        <title>Hi</title>
    </head>
    <body>
        <p>Hello</p>
        <a id = "link" href = "http://www.google.com">Link</a>
        <script type ="text/javascript" >
            var linkNode = document.getElementById("link");
            alert(linkNode.href);
        </script>
    </body>
</html>
```



DEMO LẤY ATTRIBUTE CỦA NODE (ELEMENT)

- Giả sử đoạn mã được viết lại như sau

```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
    <p>Hello</p>
    <script type ="text/javascript" >
      var linkNode = document.getElementById("link");
      alert(linkNode.href);
    </script>
    <a id = "link" href = "http://www.google.com">Link</a>
  </body>
</html>
```

Tại sao không hiển thị hộp thoại???

DEMO LẤY ATTRIBUTE CỦA NODE (ELEMENT)

- **Trả lời:** Tại vì browser làm việc theo cơ chế thông dịch. Tức là dịch từng dòng một, khi đến lệnh Javascript `document.getElementById("link")` thì chưa có Id nào tên là "Link" nên không có node nào trả về cho biến `linkNode`

DEMO ĐIỀU HƯỚNG QUA CÁC NODE

```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
    <div id="hoa">
      <p>Hoa Hong</p>
      <p>Hoa Lan</p>
      <p>Hoa Dao</p>
    </div>
    <script type ="text/javascript">
      var divHoa = document.getElementById("hoa");
      var pHoaHong = divHoa.firstChild;
      alert(pHoaHong.nextSibling.childNodes[0].nodeValue);
    </script>
  </body>
</html>
```



THÊM NODE VÀO ELEMENT

- Sử dụng phương thức **createElement** và **appendChild** để thêm node element vào tài liệu

```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
    <div id="hoa">
      <p>Hoa Hong</p>
      <p>Hoa Lan</p>
      <p>Hoa Dao</p>
    </div>
    <script type="text/javascript">
      var newElement = document.createElement("p");
      document.body.appendChild(newElement);
      var text = document.createTextNode("Hello World");
      newElement.appendChild(text );
    </script>
  </body>
</html>
```

XOÁ NODE ELEMENT

- Sử dụng phương thức **removeChild(nodeId)** của node để xoá các node element của node

```
<html>
  <head>
    <title>Hi</title>
  </head>
  <body>
    <p id="pHello">Hello</p>
    <p id="pHi">Hi</p>

    <script type="text/javascript">
      document.body.removeChild(pHi);
      //var pHi = document.getElementById("pHi");
      //document.body.removeChild(pHi);
    </script>
  </body>
</html>
```

ELEMENT CLICK HANDLER (XỬ LÝ SỰ KIỆN)

- HTML elements có thể làm điều gì đó khi bạn click chuột
- Ví dụ

```
<!DOCTYPE html>
<html>
  <body>
    <div id="one" onclick="alert('Ouch! Stop it!')">Don't click here!</div>
  </body>
</html>
```

- Hoặc

```
<!DOCTYPE html>
<html>
  <body>
    <script>
      function stop(){
        alert("Ouch! Stop it!");
      }
    </script>
    <div id="one" onclick="stop()">Don't click here!</div>
  </body>
</html>
```

ELEMENT CLICK HANDLER (XỬ LÝ SỰ KIỆN)

- HTML elements có thể làm điều gì đó khi bạn click chuột
- Sử dụng DOM

```
<!DOCTYPE html>
<html>
  <body>
    <div id="one">Don't click here!</div>
  </body>
  <script type="text/javascript">
    function e_click() {
      alert("Auch! Stop!");
    }
    document.getElementById("one").onclick = eclick();
  </script>
</html>
```

This và DOM

- This keyword có ý nghĩa tương đối và phụ thuộc vào ngữ cảnh sử dụng nó
- Trong DOM, this đề cập đến DOM element mà nó thuộc về

```
<!DOCTYPE html>
<html>
  <body>
    <script>
      function reveal(el){
        console.log(el);
      }
    </script>
    <button onclick="reveal(this)">Click here!</button>
  </body>
</html>
```

LẤY THÔNG TIN BROWSER

- Vấn đề nảy sinh:
 - Có rất nhiều trình duyệt
 - Mỗi trình duyệt lại có nhiều phiên bản
 - Mỗi trình duyệt lại hỗ trợ ở các mức khác nhau
 - ⇒ Để kiểm tra tất cả các trình duyệt, phiên bản của trình duyệt là điều không thể
 - ⇒ Thuộc tính userAgent ([navigator.userAgent](#)) cũng có thể trả về thông tin sai
- Sử dụng cách thử để biết trình duyệt hỗ trợ phương thức gì

```
if (typeof document.body.firstChild != "undefined") {  
    alert("Browser ho tro phuong thuc firstElementChild");  
} else {  
    alert("Browser khong ho tro phuong thuc firstElementChild");  
}
```

ĐỐI MẶT VỚI CÁC BROWSER CŨ

- Tạo mã Javascript chạy tốt trên tất cả các version của tất cả các trình duyệt là điều không thể
- Thiết lập một giới hạn hợp lý các trình duyệt và version để hỗ trợ
- Giới hạn càng hẹp thì càng ít khách hàng truy cập được vào website
- Đối với browser không hỗ trợ Javascript hoặc bị disable Javascript
 - Dùng thẻ <noscript></noscript>
 - Dùng thẻ <!--><!-->

TỔNG KẾT DOM

- DOM là một chuẩn được định nghĩa bởi W3C (World Wide Web Consortium) để có thể truy cập và thao tác với các tài liệu như html hay xml bằng các ngôn ngữ lập trình như Javascript, VB...
- DOM được chia làm 3 mức CoreDOM, HTML DOM và XML DOM
- HTML DOM định nghĩa các đối tượng và thuộc tính của tất cả các thành phần HTML và phương thức để truy cập đến chúng.
- Tất cả các thành phần trong tài liệu HTML đều được biểu diễn bằng đối tượng node
- DOM trình bày tài liệu HTML theo cấu trúc hình cây
- Thuộc tính định nghĩa các đặc tính node như nodeName, nodeValue, innerText, childNodes, parentNode...

TỔNG KẾT DOM

- Phương thức để thực hiện các thao tác với node như truy cập đến node, thêm node con và xoá node con.
- Có thể truy cập đến node với Id, tag name, class name, css selector
- Có rất nhiều trình duyệt, mỗi trình duyệt lại định nghĩa cấu trúc DOM và các thuộc tính, phương thức cho mỗi Node khác nhau => Sử dụng cách thử để biết trình duyệt hỗ trợ phương thức nào.