

TRƯỜNG ĐÀO TẠO LẬP TRÌNH VIÊN VÀ QUẢN TRỊ MẠNG QUỐC TẾ BACHKHOA-APTECH

# BÀI 15: HÀM VÀ ĐỐI TƯỢNG

# Mục tiêu

- Giải thích về hàm
- Giải thích về tham số của hàm
- Giải thích về câu lệnh return
- Mô tả các đối tượng
- Giải thích sự khác nhau của các đối tượng giữa các trình duyệt
- Mô tả Document Object Model (DOM)

# Giới thiệu

Để tạo cho code hướng nhiệm vụ hơn và dễ quản lý, JavaScript cho phép nhóm câu lệnh lại trước khi chúng thực sự gọi.

Điều này có thể đạt được bằng cách sử dụng các chức năng.

Một chức năng là một khối mã có khả năng tái sử dụng được thực thi trên sự xuất hiện của một sự kiện.

Sự kiện có thể là một hành động của người dùng trên các trang hoặc một lời gọi kịch bản.

# Các hàm

Là một khối mã độc lập có thể tái sử dụng và nó thực hiện một số thao tác trên các biến và các biểu thức để hoàn thành một nhiệm vụ nào đó.

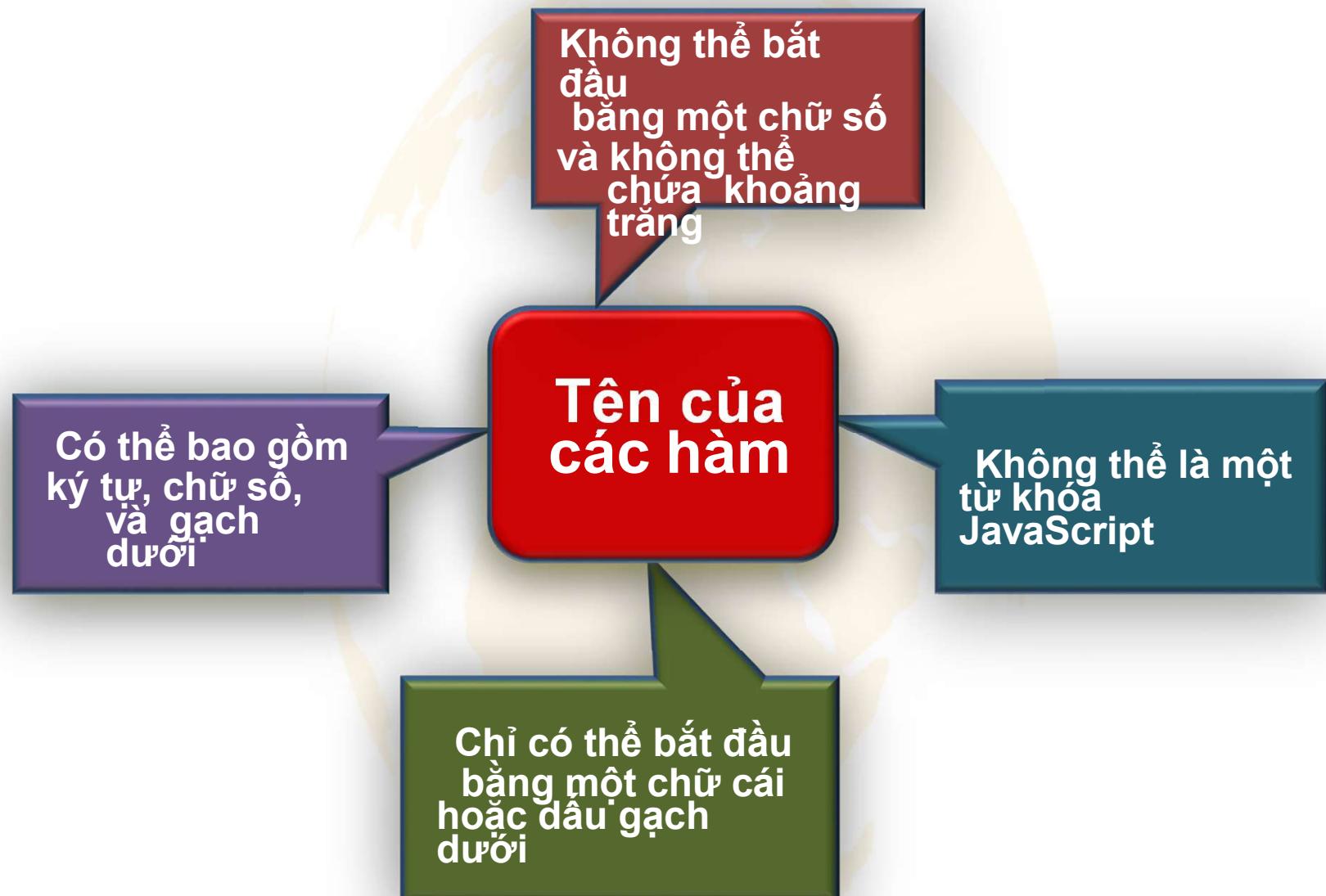
Có thể chấp nhận tham số là các biến hoặc các giá trị.

Có thể trả về giá trị kết quả để hiển thị trong trình duyệt sau khi các hoạt động đã được thực hiện.

Hàm trong JavaScript luôn được tạo trong phần tử script

JavaScript hỗ trợ cả hai loại là: hàm do người dùng định nghĩa và hàm có sẵn trong hệ thống.

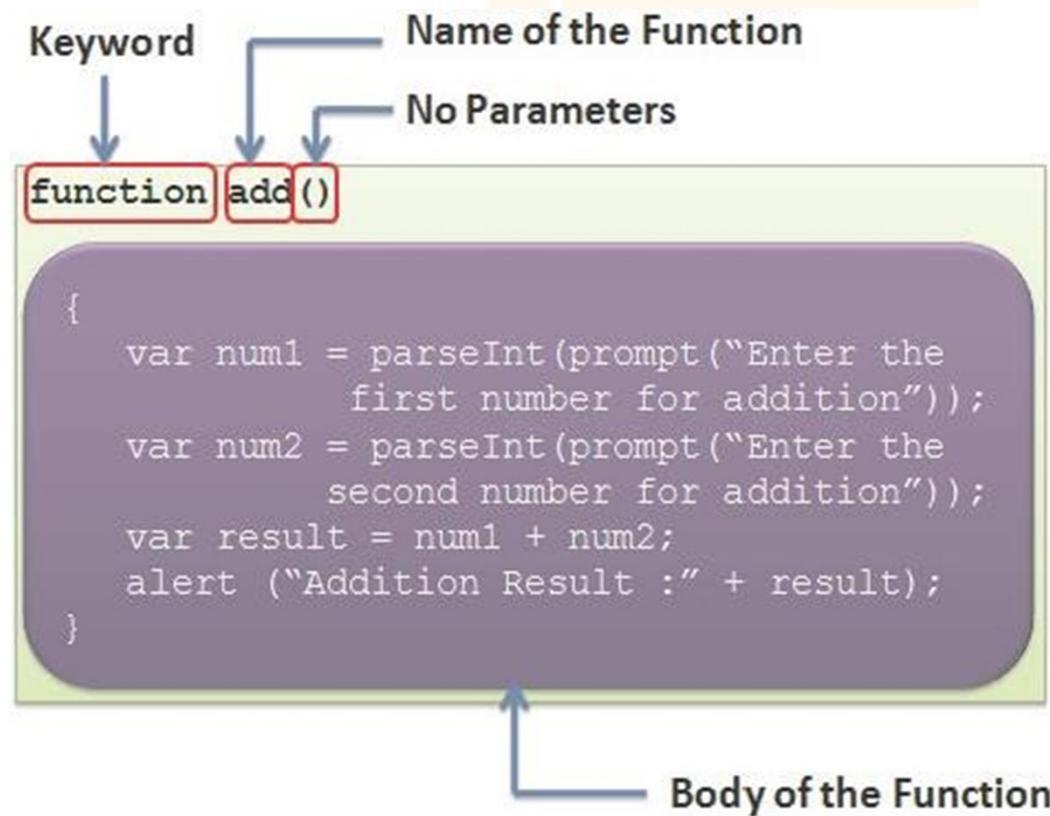
# Mô tả và định nghĩa các hàm 1-2



# Mô tả và định nghĩa các hàm 2-2

- Cú pháp định nghĩa hàm

```
function function_name(list of  
parameters)  
{  
// Body of the function  
}
```



# Gọi các hàm 1-2

Một hàm cần phải được gọi hoặc được gọi là để thực hiện nó trong trình duyệt.

Để gọi một hàm, chỉ ra tên hàm tiếp theo là cặp ngoặc () và các tham số bên trong nếu có.

Một hàm có thể được định nghĩa và gọi trong một tập tin JavaScript bên ngoài.

Một hàm có thể được gọi từ một hàm khác trong JavaScript.

Một hàm để gọi một hàm khác được gọi là hàm gọi.

Các hàm cung cấp các tiện lợi bằng cách cho phép người sử dụng để gọi một hàm nhiều lần.

## Gọi các hàm 2-2

```
<script>
    function add()
    {
        var num1 = parseInt(prompt("Enter the
            first number for addition"));
        var num2 = parseInt(prompt("Enter the
            second number for addition"));
        var result = num1 + num2;
        alert ("Addition Result :" + result);
    }

```

```
function calling_add()
{
    add();
}

calling_add();
</script>
```

Invoking the  
calling\_add() Function

Called Function

Calling Function

# Các tham số của hàm

Các hàm trong javascript có thể chấp nhận tham số

## Các tham số của hàm

Có thể được tạo ra khi có nhu cầu nhận các giá trị từ người sử dụng

Các tham số chứa các giá trị mà trên đó các hàm cần thực hiện các thao tác

Takes two numbers and stores it in val1 and val2

```
<script>  
    var val1 = parseInt(prompt("Enter the first  
        number for addition"));  
    var val2 = parseInt(prompt("Enter the second  
        number for addition"));
```

add(val1, val2);

```
function add(num1, num2)  
{  
    var result = num1 * num2;  
    alert("Addition Result: " + result);  
}  
</script>
```

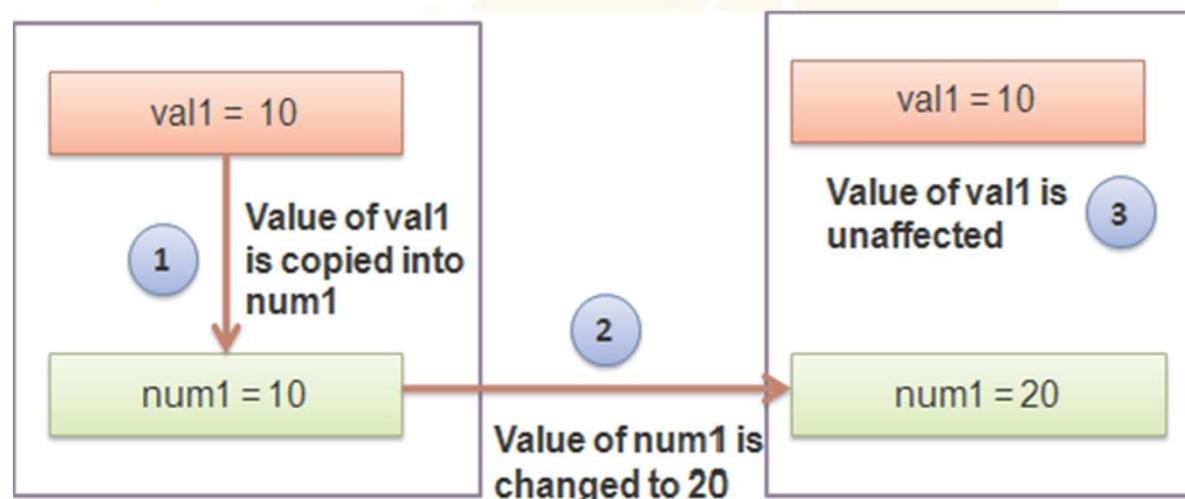
Invokes the add()  
function by passing  
val1 and val2

Holds the value of  
val1 and val2

# Các cách truyền tham số 1-3

Có hai cách truyền đối số cho hàm cụ thể là, truyền giá trị và truyền tham số.

**Truyền giá trị**- Tức là chuyển các biến như là các tham số tới một hàm. Hàm được gọi không thay đổi các giá trị của tham số được truyền tới nó khi nó được gọi.

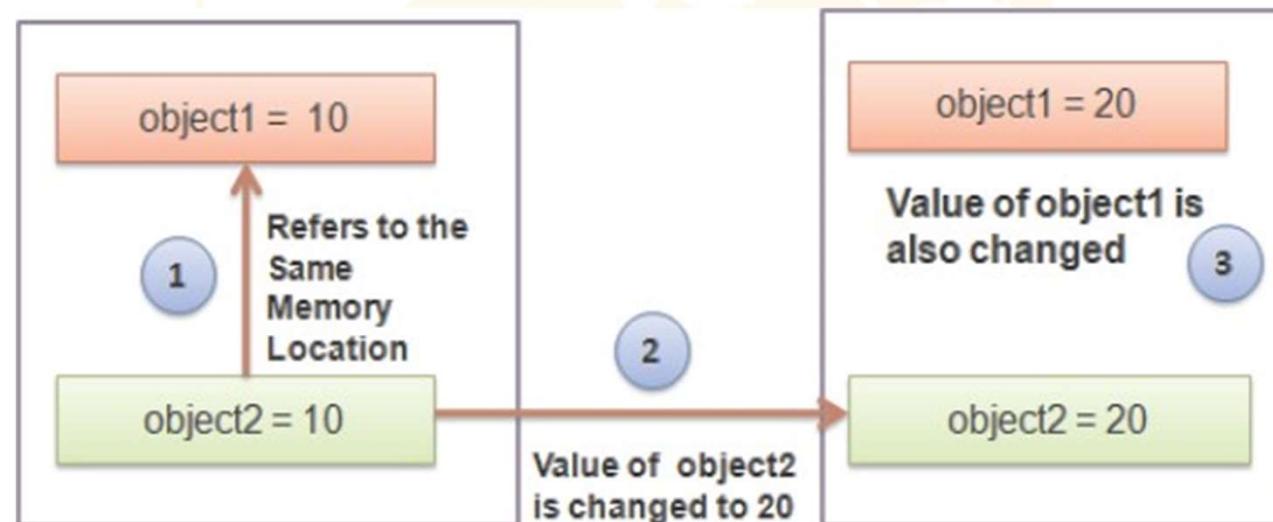


# Các cách truyền tham số 2-3

**Truyền tham chiếu** – Tức là truyền đối tượng như là hàm số.

Hàm được gọi thay đổi giá trị của các thông số được truyền cho nó từ hàm đang gọi

Sự thay đổi này được phản ánh khi lời gọi hàm kết thúc.



# Các cách truyền tham số 3-3

- Ví dụ

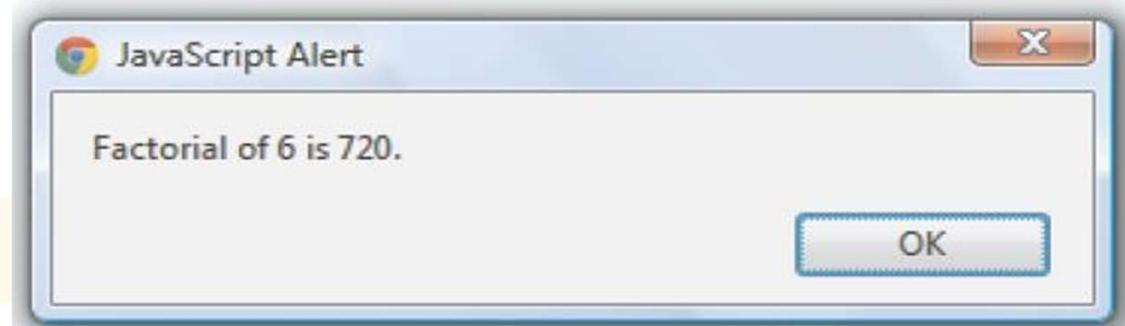
```
<script>
    var names =new Array('James', 'Kevin', 'Brad');
    function change_names(names) {
        names[0]= 'Stuart';
    }
    function display_names() {
        document.writeln('<H3> List of Student Names:</H3>');
        document.write('<UL>');
        for(vari=0; i<names.length; i++) {
            document.write('<LI>' + names[i]+ '</LI>');
        }
        document.write('</UL>');
        change_names(names);
        document.write('<H3> List of Changed Students Names:</H3>');
        document.write('<UL>');
        for(vari=0; i<names.length; i++) {
            document.write('<LI>' + names[i]+ '</LI>');
        }
        document.write('</UL>');
    }
    display_names();
</script>
```

# Câu lệnh return

Trả về kết quả cho một hàm

- Ví dụ

```
<script>  
function factorial (num) {  
    if (num==0)  
        return 0;  elseif (num==1)  
    return 1;  else  
        {  
            var result = num;  
            while (num>1)  
            {  
                result = result * (num-1);  num--;  
            }  
            return result;  
        }  
}  
var num=prompt('Enter number:', '' );  var result =  
factorial(num);  
alert('Factorial of ' +num+ ' is ' + result + '.');  
</script>
```



# Đối tượng trong JavaScript 1-2

Là những thực thể với các thuộc tính và phương thức và giống như các đối tượng trong thế giới thực.

Các thuộc tính là các đặc tính hoặc đặc điểm của một đối tượng.

Phương thức xác định hành vi của một đối tượng.



<b>Properties</b>	Make - ford Color - green Wheels – four
<b>Methods</b>	run() stop()

Object: Bird



<b>Properties</b>	Type - pigeon Color - gray Wings - two
<b>Methods</b>	eat() fly()

## Đối tượng trong JavaScript 2-2

- JavaScript cung cấp các đối tượng được xây dựng sẵn và các đối tượng do người dùng định nghĩa.

Built-in Objects – là những đối tượng được định nghĩa sẵn trong javascript.

Custom Objects – là những đối tượng do người dùng định nghĩa.

# Tạo đối tượng người dùng 1-3

Object là đối tượng cha của tất cả các đối tượng trong JavaScript.

Đối tượng do người dùng tạo có thể được bắt nguồn từ đối tượng Object bằng cách sử dụng các từ khóa new.

Hai phương pháp chính để tạo ra một đối tượng cụ thể là, phương pháp trực tiếp và phương pháp dùng mẫu và khởi tạo nó với các từ khóa new.

# Tạo đối tượng người dùng 2-3

- Cú pháp
- Cách tạo trực tiếp

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

- Cách tạo bằng mẫu
- Cú pháp tạo constructor

```
function object_type(list of parameters)
{
    // Body specifying properties and
    methods
}
```

# Tạo đối tượng người dùng 3-3

- Cú pháp tạo đối tượng mới bằng từ khóa new

```
object_name = new object_type(optional list of arguments);
```

- Ví dụ

```
<script>
  //create an object using direct method
  var doctor_details=new Object();
  //create an object using new keyword
  studOne = new student_info ('James', '23', 'New
  Jersey');
</script>
```

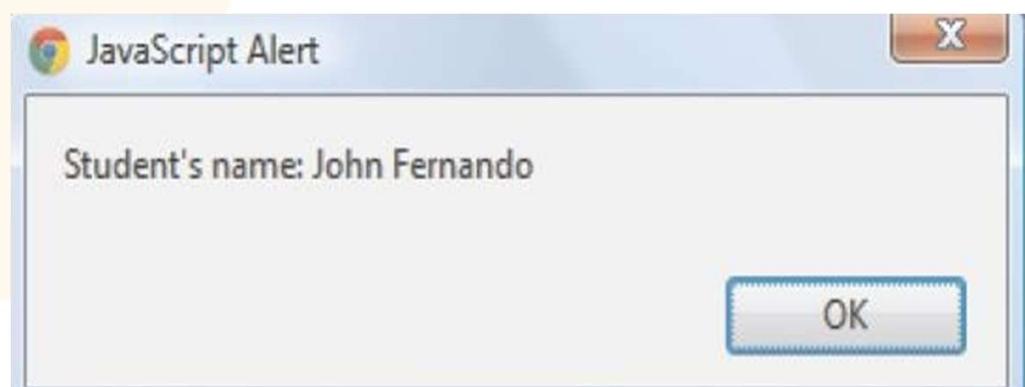
# Tạo thuộc tính cho các đối tượng người dùng 1-2

Các thuộc tính là các đặc điểm của đối tượng

Để truy cập thuộc tính chúng ta sử dụng tên đối tượng tiếp theo là dấu chấm và cuối cùng là tên thuộc tính.

- Ví dụ.

```
<script>  
var student_details=new Object();  
student_details.first_name= 'John';  
student_details.last_name= 'Fernando';  
student_details.age= '15';  
alert('Student\'s name: ' +student_details.first_name+ ' '  
+student_details.last_name);  
</script>
```



# Tạo thuộc tính cho các đối tượng người dùng 2-2

- Ví dụ tạo đối tượng và thêm thuộc tính cho đối tượng

```
<script>
    // To define the object type
    function employee_info(name, age, experience)
    {
        this.name =
            name;  this.age=
            age;
        this.experience= experience;
    }
    // Creates an object using new keyword
    empMary=newemployee_info('Mary', '34', '5 years');
    alert("Name: "+empMary.name + '\n' +"Age: "+empMary.age+
        '\n'
    +"Experience: "+empMary.experience);
</script>
```



# Tạo phương thức cho các đối tượng người dùng

Phương thức tương tự như các hàm trong JavaScript

Một phương thức được gắn với một đối tượng và thực bởi đối tượng đó trong khi hàm không gắn với đối tượng nào và thực thi độc lập.

Các phương thức sẽ được chỉ ra sau khi đối tượng được tạo hoặc trong khi tạo đối tượng mẫu

Để gọi phương thức chúng ta chỉ ra tên đối tượng tiếp theo là dấu chấm rồi đến tên phương thức kèm theo cặp ngoặc (), bên trong ngoặc có thể có tham số.

- Ví dụ.

```
<script>
    var square =new Object();
    square.length=parseInt("5");
    square.cal_area=function() {
        var area =(parseInt(square.length)*parseInt("4"));
        return area;
    }
    alert("Area: "+square.cal_area());
</script>
```

# Các đối tượng xây dựng sẵn

Mô hình Object của ngôn ngữ JavaScript hình thành nền tảng của ngôn ngữ.

Các đối tượng này cung cấp các chức năng tùy chỉnh trong script.

JavaScript xử lý các kiểu dữ liệu nguyên thủy như các đối tượng và cung cấp đối tượng tương đương cho chúng.

Các đối tượng JavaScript bao gồm: built-in objects, browser objects, và HTML objects.

Built-in objects là các đối tượng tĩnh có thể mở rộng thêm các tính năng trong script

Browser objects như window, history, và navigator được sử dụng để làm việc với cửa sổ trình duyệt

HTML objects, như form, anchor, .. được sử dụng để truy cập các phần tử trên trang web.

# Đối tượng String 1-3

Strings trong JavaScript là tập các ký tự được đặt trong cặp ngoặc kép hoặc cặp ngoặc đơn.

Đối tượng String cho phép thực hiện các thao tác trên dữ liệu dạng văn bản

- Ví dụ:

```
var object_name = new String("Set of characters")
;
```

- Các thuộc tính của String

Thuộc tính	Mô tả
length	Lấy số ký tự trong một chuỗi.
prototype	Bổ sung thêm thuộc tính và phương thức người dùng định nghĩa cho thể hiện của String.

# Đối tượng String 2-3

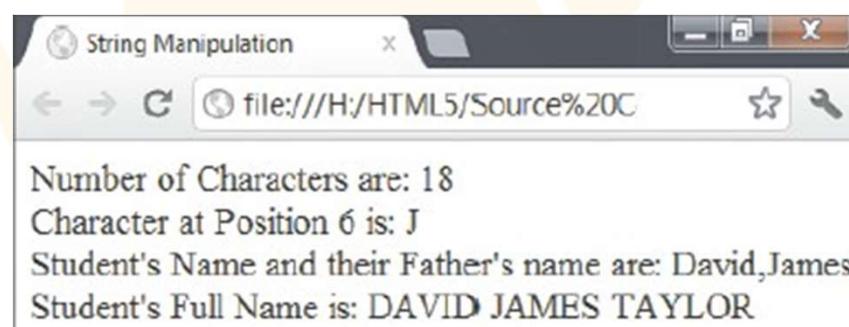
- Các phương thức của đối tượng String.

Phương thức	Mô tả
charAt()	Lấy một ký tự từ một vị trí cụ thể trong một chuỗi.
concat()	Nối các ký tự từ một chuỗi với các ký tự từ một chuỗi khác và trả ra một chuỗi mới duy nhất.
indexOf()	Lấy vị trí mà tại đó chuỗi giá trị tìm thấy đầu tiên xuất hiện trong chuỗi.
lastIndexOf()	Lấy vị trí mà tại đó chuỗi giá trị tìm thấy cuối cùng xuất hiện trong chuỗi.
replace()	Thay thế các mẫu tìm thấy trong chuỗi bằng một chuỗi khác

# Đối tượng String 3-3

## ● Ví dụ

```
<script>
var full_name=new String('David James Taylor');
document.write('Number of Characters are: ' +full_name.length+
'<BR/>');
document.write('Character at Position 6 is: ' +
+full_name.charAt(6)+ '<BR/>');
document.write('Student\'s Name and their Father\'s name are:
' +full_name.split(' ',2)+ '<BR/>');
document.write('Student\'s Full Name is: ' +
+full_name.toUpperCase());
</script>
```



# Đối tượng Math 1-2

Đối tượng Math cho phép người dùng thực hiện các phép toán trên các giá trị số.

Đối tượng Math là một đối tượng được xác định trước mà cung cấp các thuộc tính tinh và phương thức để thực hiện các hoạt động toán học.

Thuộc tính và phương thức được khai báo là tinh, vì vậy chúng có thể được gọi trực tiếp với tên đối tượng.

- Cú pháp gọi thuộc tính:

```
var variable_name = Math.PropertyName;
```

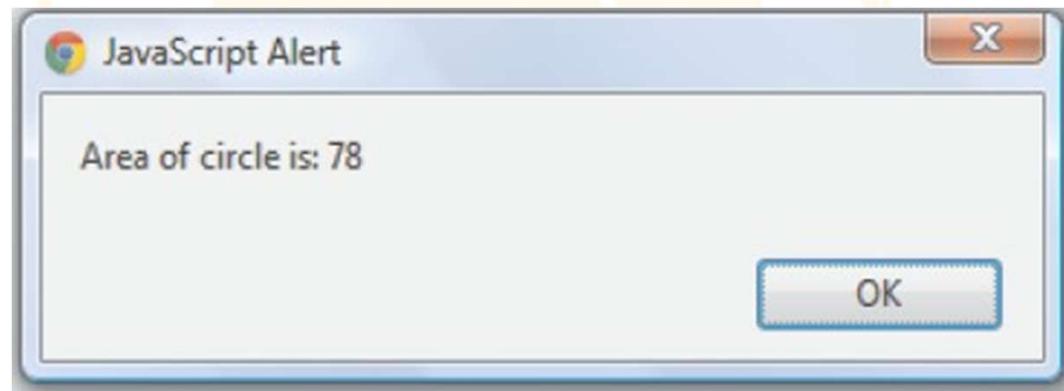
- Cú pháp gọi phương thức:

```
var variable_name = Math.MethodName(optional list of  
parameters);
```

# Đối tượng Math 2-2

- Ví dụ

```
<script>
    var full_name=new String('David James Taylor');
    document.write('Number of Characters are: '
+full_name.length+ '<BR/>');
    var area
    =Math.floor(tempArea); return
area;
}
alert('Area of circle is: ' +area_circle(5));
</script>
```



# Đối tượng Date 1-3

Đối tượng Date cho phép bạn định nghĩa và thao tác các giá trị ngày và thời gian lập trình.

Nó hỗ trợ cả hai Universal Coordinated Time (UTC) và Greenwich Mean Time (GMT).

Đối tượng Date tính toán ngày tháng trong phần nghìn giây từ 01 Tháng 1 năm 1970.

- Tạo đối tượng Date:

```
var object_name = new Date();
var object_name = new Date(milliseconds);
var object_name = new Date(year,           day,   hour,   minutes,
                           month,  seconds, milliseconds);
var object_name = new Date("dateString");
```

# Đối tượng Date 2-3

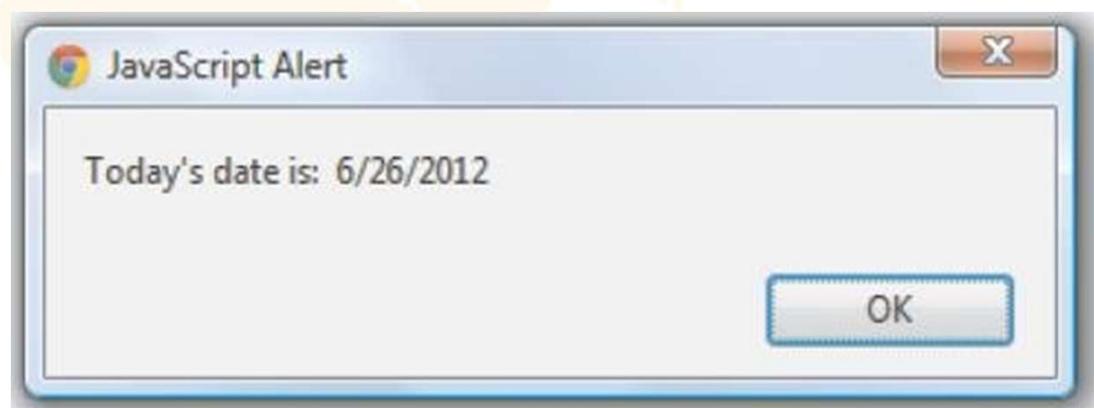
- Các phương thức của Date

Phương thức	Mô tả
getDate()	Lấy một giá trị số từ 1 đến 31, trong đó cho biết ngày của tháng.
getDay()	Lấy một giá trị số từ 0 đến 6, cho biết ngày trong tuần.
getTime()	Lấy một giá trị số, trong đó cho biết thời gian trôi qua trong mili giây tính từ nửa đêm 01/01/1970.
getFullYear()	Lấy bốn chữ số giá trị số, biểu thị năm trong ngày nhất định.

# Đối tượng Date 3-3

- Ví dụ

```
<script>
function display_date()
{
var today =new Date();
var date =today.getDate();
var month =today.getMonth();
month++;
var year =today.getFullYear();
alert('Today\'s date is: ' + month + '/' + date + '/' +
year);
}
display_date();
</script>
```



# Câu lệnh with

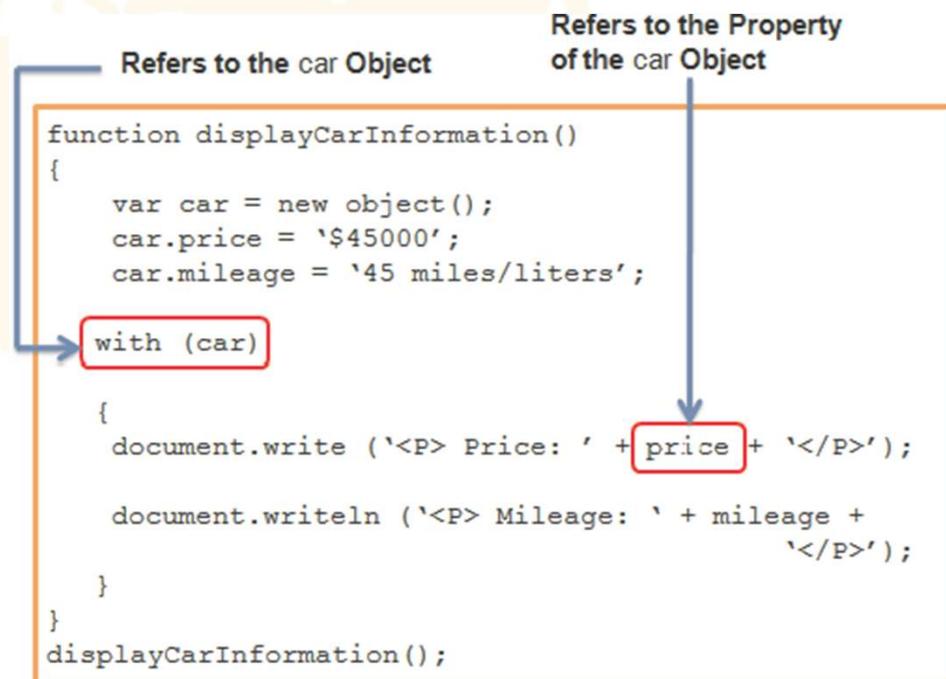
câu lệnh **with** cho phép để loại bỏ các tham chiếu đối tượng cho mỗi câu JavaScript.

câu lệnh **with** bắt đầu với từ khóa with sau là dấu ngoặc mở và đóng, nắm giữ các câu lệnh đề cập đến một đối tượng chung.

câu lệnh **with** tăng khả năng đọc mã và cũng làm giảm thời gian cần thiết khi viết từng đối tượng tham chiếu trong mỗi câu lệnh liên quan.

- Ví dụ:

```
with(object_name)
{
//Statements
}
```

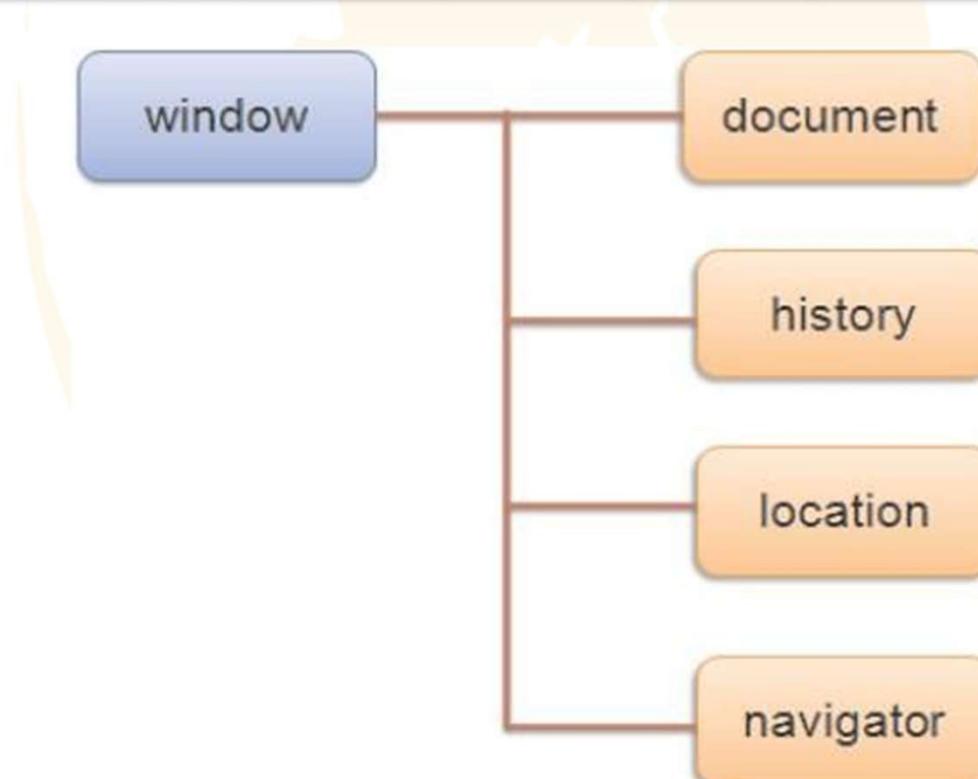


# Các đối tượng Browser

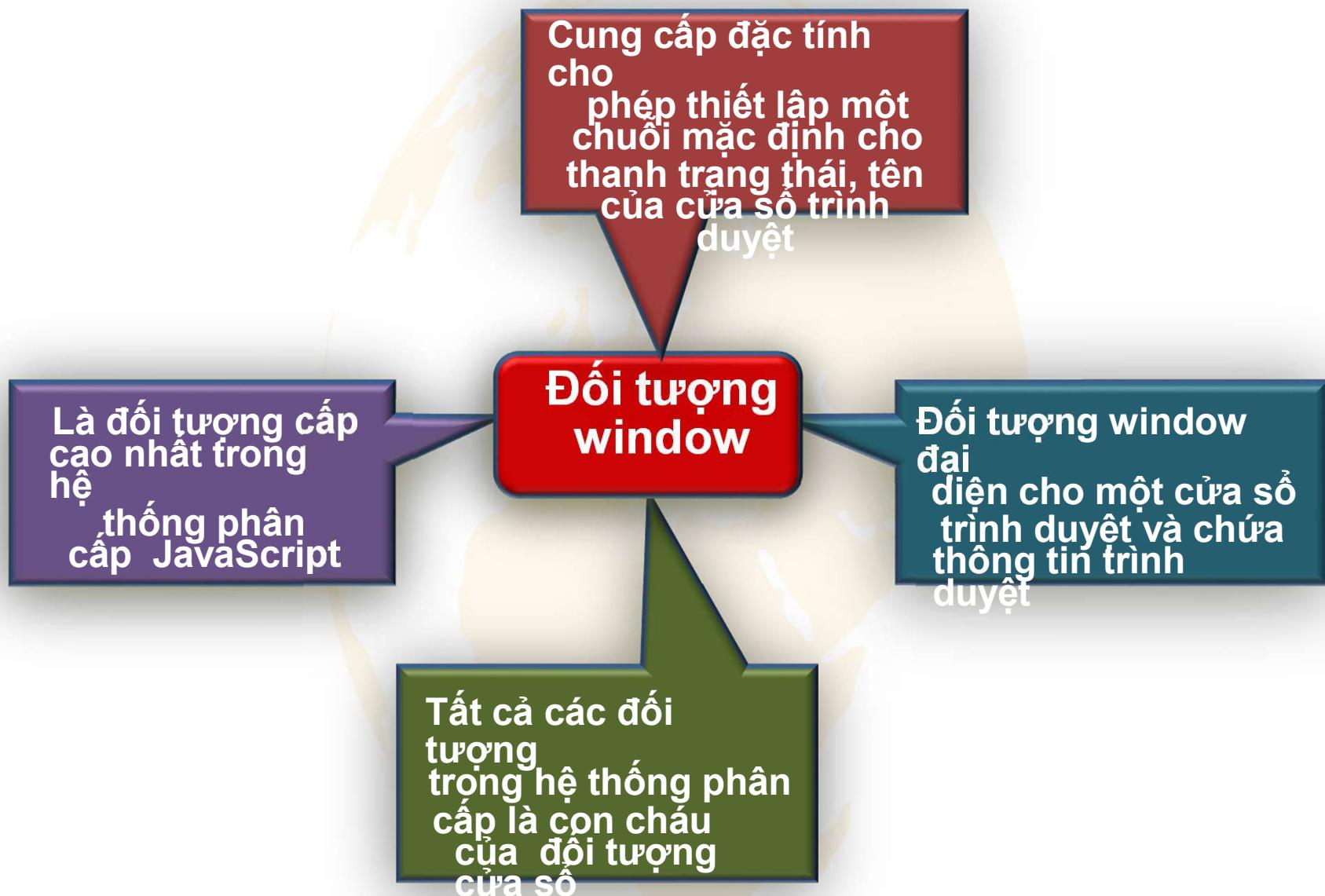
JavaScript cũng cung cấp các đối tượng để truy cập và thao tác các khía cạnh khác nhau của trình duyệt Web.

Các đối tượng này được gọi là **đối tượng trình duyệt**.

Chúng tồn tại trên tất cả các trang hiển thị trong trình duyệt và tương ứng với các yếu tố của một trang.



# Đối tượng Window 1-4



# Đối tượng Window 2-4

- Bảng sau liệt kê các thuộc tính của đối tượng window

Thuộc tính	Mô tả
defaultStatus	Quy định cụ thể hoặc lấy chuỗi mặc định được hiển thị trên thanh trạng thái của cửa sổ trình duyệt.
document	Đại diện cho một tài liệu HTML có chứa các phần tử khác nhau.
history	Chứa lịch sử của Uniform Resource Locators (URL).
location	Chứa nội dung của URL được chỉ định.

# Đối tượng Window 3-4

- Bảng sau liệt kê các phương thức của đối tượng window

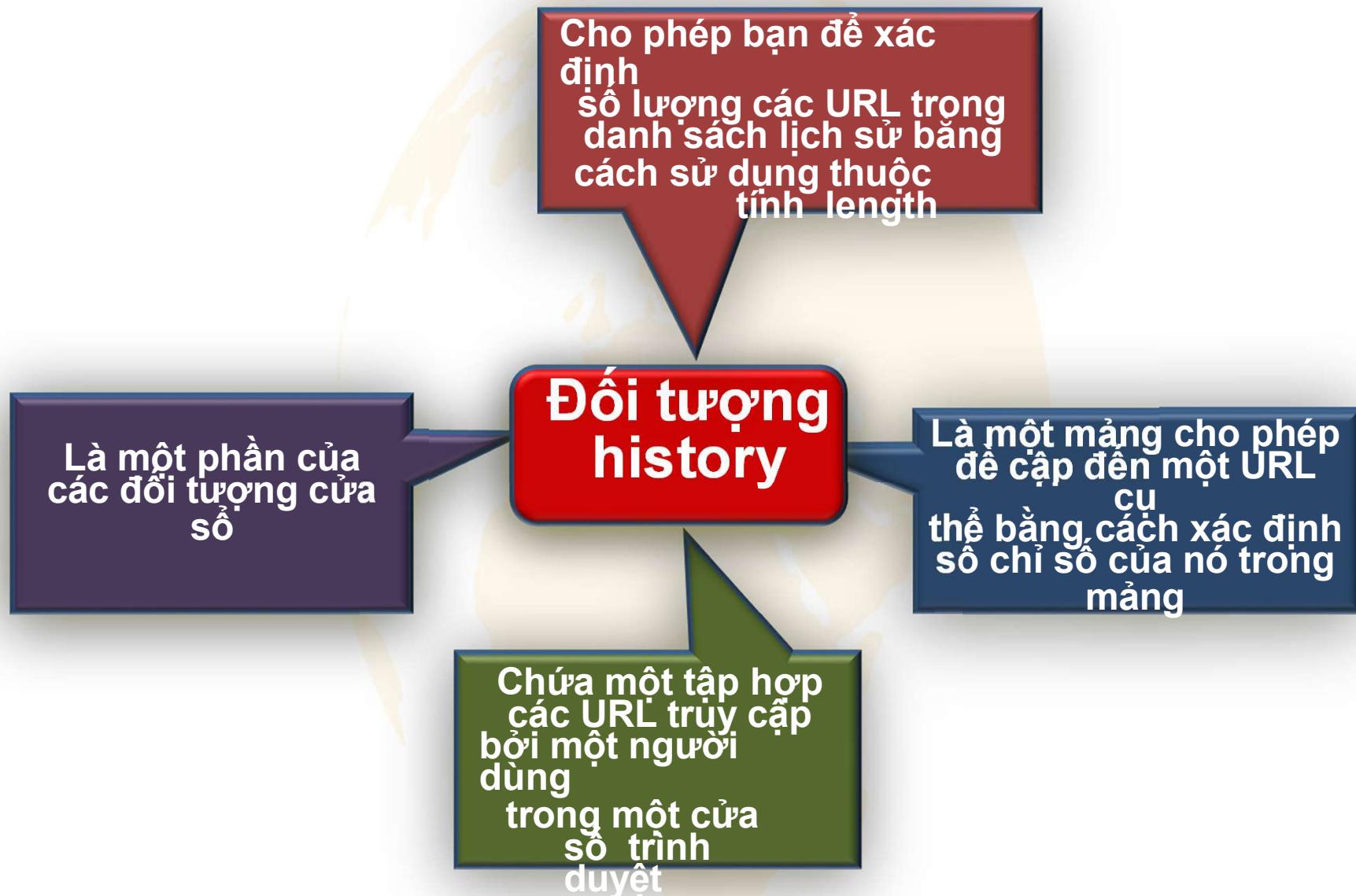
Phương thức	Mô tả
alert()	Hiển thị một hộp cảnh báo rằng tình trạng tin nhắn và một nút OK.
confirm()	Nhắc nhở một hộp thoại hiển thị thông báo với các nút OK và Cancel.
createPopup()	Tạo ra một cửa sổ pop-up.
focus()	Tập trung các cửa sổ hiện hành.
open()	Mở tập tin xác định trong một cửa sổ trình duyệt mới.
prompt()	Nhắc nhở một hộp thoại mà chấp nhận đầu vào từ người sử dụng.

# Đối tượng Window 4-4

- Ví dụ.

```
<!DOCTYPE html>
<head>
<title> window Object </title>
<script>
function new_window() {
if(confirm('Do you want to open a new page?')) {
window.open('http://www.aptech-education.com/pages/about-
us/about-aptech.html', '_parent');
}
else {
window.alert('In the Current Window');
}
}
</script>
</head>
<body>
<input type="button" value="Click to move to the next
page" onClick="new_window();"/>
</body>
</html>
```

# Đối tượng History 1-2



# Đối tượng history 2-2

- Các phương thức của đối tượng history.

Phương thức	Mô tả
back()	Lấy và hiển thị các URL trước đó từ danh sách lịch sử.
forward()	Lấy và hiển thị URL tiếp theo trong danh sách lịch sử.
go()	Hiển thị URL được chỉ định. Nó chấp nhận một tham số, mà một trong hai có thể là một chuỗi hoặc một số để đi đến trang cụ thể.

# Đối tượng navigator 1-2

## Đối tượng navigator

Chứa thông tin về các trình duyệt được sử dụng bởi máy khách

Cho phép người sử dụng để lấy thông tin, chẳng hạn như tên, số phiên bản, ..

- Bảng sau liệt kê thuộc tính của navigator.

Thuộc tính	Mô tả
appName	Lấy tên của trình duyệt.
appVersion	Lấy số phiên bản và nền tảng của trình duyệt.
browserLanguage	Lấy ngôn ngữ của trình duyệt.
cookieEnabled	Xác định xem các tập tin cookie được kích hoạt trong trình duyệt.
platform	Lấy các loại máy như Win32, của trình duyệt của máy khách.

# Đối tượng navigator 2-2

- Ví dụ

```
<!DOCTYPE html>
<head>
    <title> navigator Object </title>
    <script>
        function display_browser() {
            document.write('Browser name: ' +navigator.appName+ '<BR/>');
            document.write('Browser version: ' +navigator.appVersion+
'<BR/>');
            document.write('Browser language: ' +navigator.browserLanguage+
'<BR/>');
            document.write('Platform: ' +navigator.platform+ '<BR/>');
            if(navigator.cookieEnabled) { document.write('Cookie
is enabled in the browser.');
        }
    }
</script>
</head>
<body>
    <input type="button" value="Browser Information"
onclick="display_browser()"/>
</body>
</html>
```

# Đối tượng location

## Đối tượng location

Cho phép truy cập thông tin đầy đủ của URL tải trong cửa sổ trình duyệt

Là một phần của đối tượng Window

- Bảng sau liệt kê phương thức và thuộc tính của đối tượng location

Phương thức/Thuộc tính	Mô tả
host	Lấy tên máy và số cổng URL.
href	Quy định cụ thể hoặc lấy toàn bộ URL.
pathname	Quy định cụ thể hoặc lấy tên đường dẫn của URL.
assign()	Tải một tài liệu mới với các URL được chỉ định.
reload()	Đổi lại các tài liệu hiện tại bằng cách một lần nữa gửi yêu cầu đến máy chủ.
replace()	Ghi đè lịch sử URL cho tài liệu hiện tại với tài liệu mới.

# DOM-Mô hình đối tượng tài liệu 1-3

Một trang web có chứa phần tử khác nhau, chẳng hạn như nút, hộp văn bản, hộp kiểm tra, ..

Những phần tử này tồn tại trong một hệ thống phân cấp và tổng thể đại diện cho một tài liệu HTML.

JavaScript cho phép người dùng truy cập các phần tử HTML và cũng thay đổi cấu trúc hiện có của một trang HTML bằng cách sử dụng Document Object Model (DOM) đặc điểm kỹ thuật.

DOM là một Application Programming Interface (API) xác định cấu trúc đối tượng để truy cập và thao tác các phần tử HTML.

Được sử dụng với JavaScript để thêm, chỉnh sửa, hoặc xóa, các nội dung trên trang web.

Thông số kỹ thuật DOM được đặt bởi World Wide Web Consortium (W3C) và được thực hiện bởi tất cả các trình duyệt để khắc phục các vấn đề không tương thích.

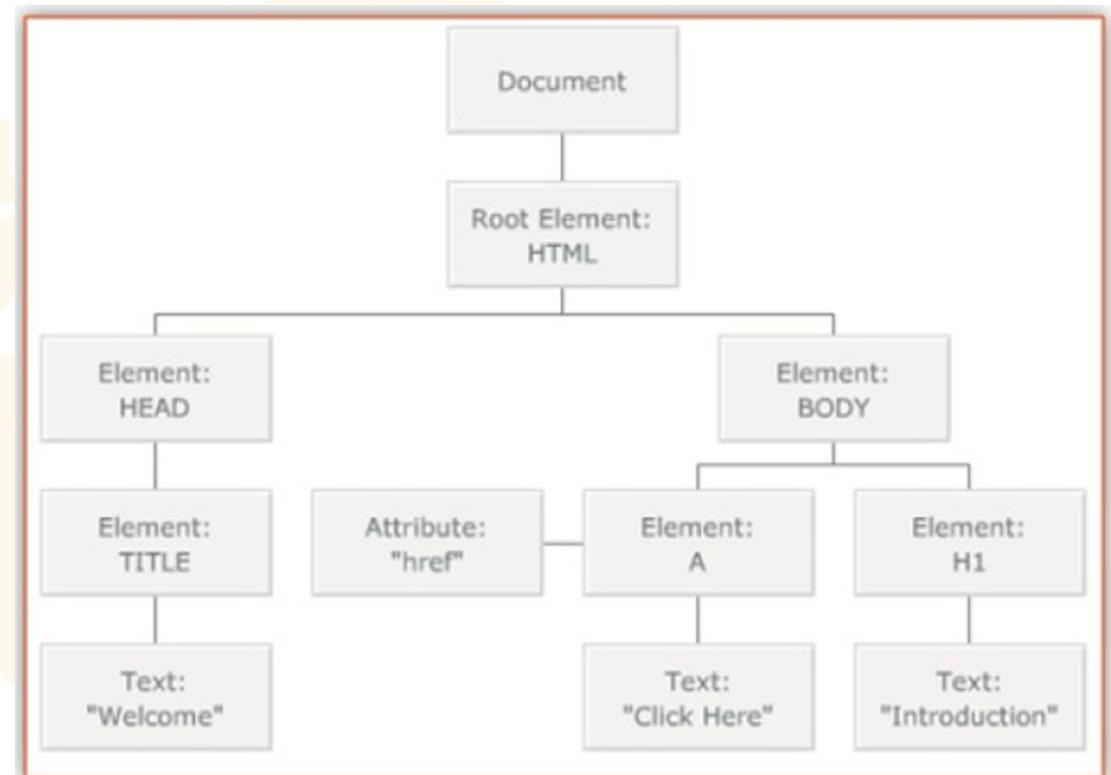
DOM đọc tất cả các phần tử có trong một trang HTML và xử lý các phần tử HTML là các nút.

Toàn bộ tài liệu HTML đại diện cho một nút tài liệu. Nút tài liệu này bao gồm các nút phần tử, các nút thuộc tính, và các nút văn bản. Nút tài liệu là nút cấp cao nhất và các nút văn bản là những người thấp nhất.

# DOM-Mô hình đối tượng tài liệu 2-3

- Ví dụ

```
<!DOCTYPE html>
<html lang="en">
<head>
  <title>Welcome</title>
</head>
<body>
  <h1> Introduction </h1>
  <a href="#">Click
    Here</a>
</body>
</html>
```



# DOM-Mô hình đối tượng tài liệu 3-3

- Tất cả các nút hiện tại trong hệ thống nút chứa các thuộc tính nhất định.
- Các thuộc tính này cung cấp thông tin về nút. Các thuộc tính nút khác nhau như sau:

nodeName - Đại diện cho tên của nút. Nó chứa các tên thẻ của phần tử HTML trong trường hợp trên.

nodeValue - Đại diện văn bản chứa trong các nút. Thuộc tính này chỉ có sẵn cho các nút thuộc tính và không cho tài liệu và phần tử nút.

nodeType – Đại diện cho các loại nút. Ví dụ, nút tài liệu, nút phần tử,..

- HTML DOM cung cấp đối tượng tiêu chuẩn cho các tài liệu HTML và một số các đối tượng này như sau:
  - Document object
  - Form object
  - Link object
  - Table object

# Đối tượng Document 1-3

Được sử dụng trong Javascript để truy cập tất cả các phần tử HTML được trình bày trên trang.

Đại diện cho toàn bộ tài liệu HTML và cung cấp truy cập cho các phần tử khác, chẳng hạn như liên kết, neo, ..

Chỉ có một đối tượng tài liệu được tạo ra khi phần tử BODY được tải trên trang web.

Cũng là một phần của đối tượng cửa sổ và được truy cập như window.document.

Cung cấp thuộc tính cho phép người dùng chỉ định hoặc lấy các thông tin về các phần tử và nội dung của nó.

## Đối tượng Document 2-3

- Bảng sau liệt kê thuộc tính của đối tượng document

Phương thức	Mô tả
close()	Đóng một dòng dữ liệu và hiển thị các dữ liệu thu thập bằng cách sử dụng phương thức open().
getElementById()	Lấy một tập hợp các phần tử HTML bằng cách sử dụng ID.

# Đối tượng Document 3-3

- Ví dụ

```
<!DOCTYPE html>
<head> <title> Document Object </title>
<script>
function change_image() {
    var imgText=document.getElementById('myImg').alt;
    if(imgText=="ford") {
        document.getElementById('myImg').src="ferrari.jpg";
        document.getElementById('myImg').alt ="ferrari";
        document.getElementById('mytext').value ="Ferrari Car";
    } else { document.getElementById('myImg').src="ford.jpg";
        document.getElementById('myImg').alt ="ford";
        document.getElementById('mytext').value ="Ford Car";
    }
}</script> </head>
<body>
<imgid="myImg" src="ford.jpg" width="300" height="300" alt="ford"/> <br/>
Model:      <input type="text" id="mytext" value="Ford Car"
readonly="readonly"/> <br/><br/>
<input type="button" value="Change Image" onclick="change_image()"/>
</body>
```

# Đối tượng Form 1-2

Chấp nhận đầu vào từ người sử dụng và gửi dữ liệu người dùng đã được xác nhận.

Một tài liệu HTML có thể chứa nhiều form

Đặc điểm kỹ thuật DOM cung cấp một đối tượng form đại diện cho một form HTML được tạo ra cho mỗi thẻ <form> trong một tài liệu HTML.

# Đối tượng Form 2-2

- Ví dụ

```
<!DOCTYPE html>
<html>
  <head>
    <title> Form Object </title>
    <script>
      function display_length() {
        var count = document.getElementById("form1").length;
        alert('Number of controls on the form: ' + count);
      }
    </script>
  </head>
  <body>
    <form id="form1" action="welcome.php">
      First name: <input type="text" name="firstname" value="John"/><br />
      Last name: <input type="text" name="lastname" value="Smith" /><br/>
      Age : <input type="text" name="age" value="40" /><br />
      <input type="button" value = "Controls" onClick="display_length()" />
    </form>
  </body>
</html>
```

# Tổng kết

- Một hàm là phần tái sử dụng mã, thực hiện tính toán trên các tham số và các biến khác.
- Các câu lệnh return trả kết quả đầu ra sau khi gọi hàm.
- Đối tượng là thực thể có thuộc tính và các phương thức và giống như các đối tượng thực tế đời sống.
- Có hai cách để tạo ra một đối tượng tùy biến cụ thể là, bằng cách trực tiếp từ đối tượng Object đối tượng hoặc bằng cách tạo ra một hàm tự xây dựng.
- JavaScript cung cấp các đối tượng khác nhau được xây dựng sẵn, chẳng hạn như String, Math, và Date.
- JavaScript cũng cung cấp các đối tượng Browser, chẳng hạn như window, history, location, và navigation.
- DOM là một kỹ thuật tiêu chuẩn cho việc truy cập và thao tác các phần tử HTML. DOM cung cấp một đối tượng tài document được sử dụng trong Javascript để truy cập tất cả các phần tử HTML được trình bày trên trang.

TRƯỜNG ĐÀO TẠO LẬP TRÌNH VIÊN VÀ QUẢN TRỊ MẠNG QUỐC TẾ BACHKHOA-APTECH

---

THANK FOR WATCH !

