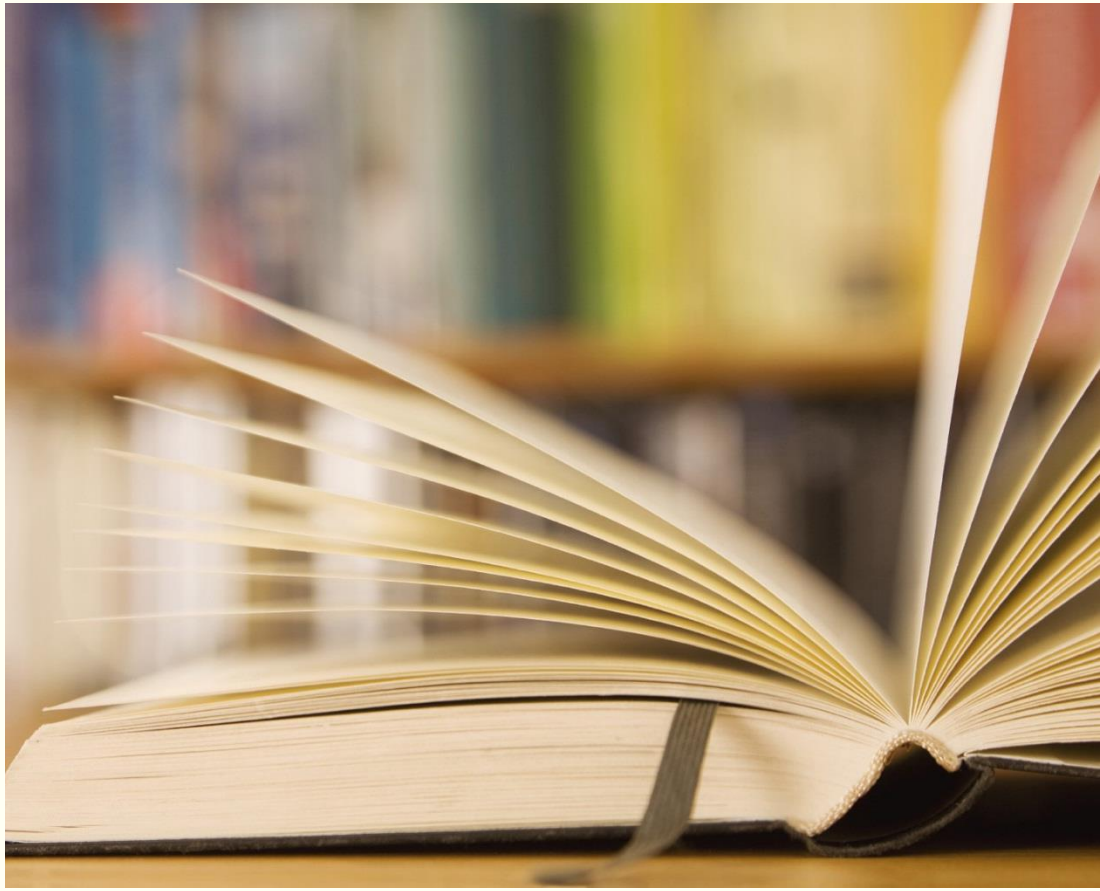
JAVASCRIPT

Từ Thị Xuân Hiền



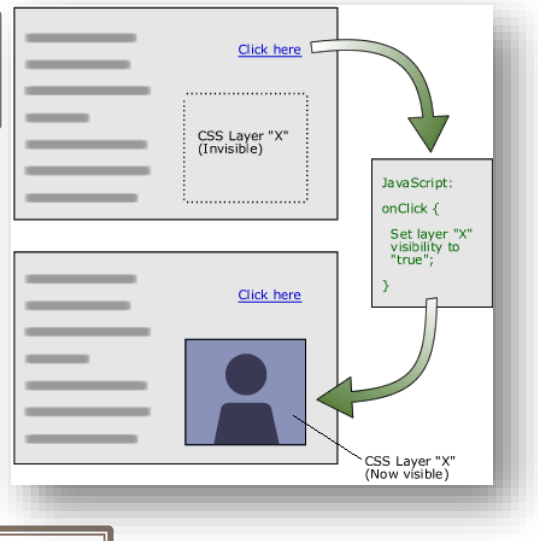
Nội dung

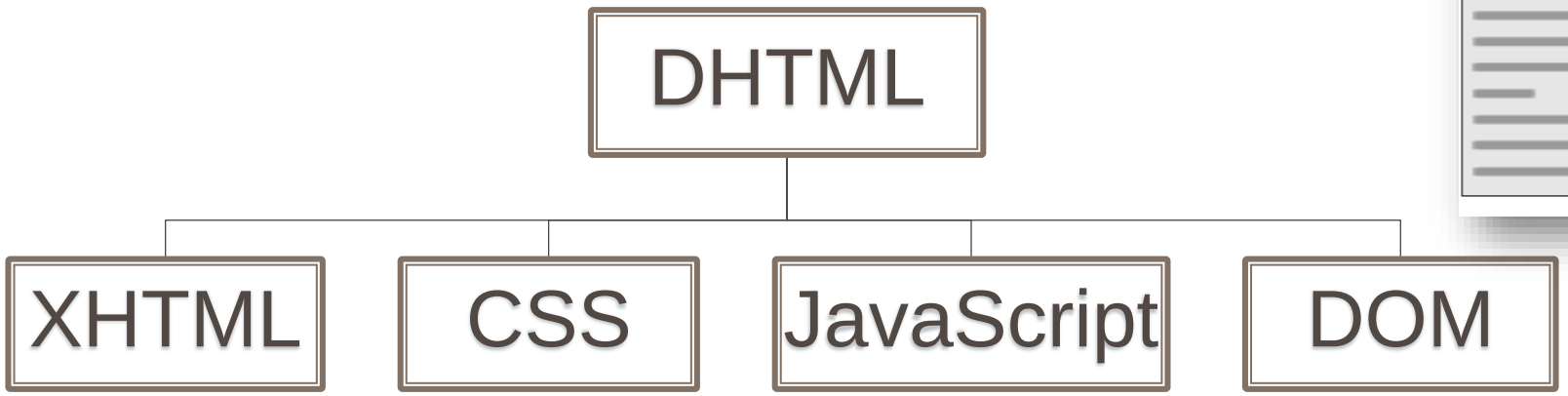
- Giới thiệu DHTML
- Công nghệ DHTML: XHTML, CSS, JavaScript, DOM
- Giới thiệu về JavaScript
- Cú pháp JavaScript
- Document Object Model
- Gỡ lỗi trong JavaScript

Giới thiệu về DHTML (*Dynamic HTML)*

- **Dynamic HTML (DHTML):** Làm cho một trang Web có thể phản ứng và thay đổi theo hành động của người dùng

DHTML = HTML + CSS + JavaScript





Giới thiệu về DHTML (*Dynamic HTML)*

- **HTML** xác định nội dung trang Web thông qua các thẻ ngữ nghĩa: headings, paragraphs, lists, …
- **CSS** xác định ‘rules' hoặc ‘Style' để trình bày mọi khía cạnh của tài liệu HTML:
  - Font (family, size, color, weight, etc.)
  - Background (color, image, position, repeat)
  - Position và layout: của bất kỳ đối tượng nào trên trang
- **JavaScript** xác định hành vi động tương tác với người dùng, để xử lý các sự kiện, v.v.

JavaScript

- **JavaScript** là một **ngôn ngữ kịch bản front-end** do **Netscape** phát triển cho nội dung động
  - Nhẹ, nhưng với khả năng hạn chế
  - Có thể được sử dụng như ngôn ngữ hướng đối tượng
  - Công nghệ Client-side
  - Được nhúng vào trang HTML
  - Được thông dịch bởi trình duyệt Web
  - Đơn giản và linh hoạt
  - Mạnh mẽ để thao tác DOM (Document Object Model)

JavaScript

- **Chức năng JavaScript**
  - Triển khai xác thực biểu mẫu - Form
  - Đáp trả các hành động của người dùng, ví dụ: xử lý các phím
  - Thay đổi hình ảnh khi di chuyển chuột qua nó
  - Các phần của trang xuất hiện và biến mất, nội dung tải và thay đổi động
  - Thực hiện các phép tính phức tạp
  - Các điều khiển HTML tùy chỉnh, ví dụ: bảng có thể cuộn
  - Triển khai chức năng AJAX

JavaScript

- **Hoạt động của JavaScript**
  - Xử lý các sự kiện
  - Đọc và ghi các phần tử HTML và sửa đổi cây DOM
  - Xác thực dữ liệu biểu mẫu
  - Truy cập hoặc sửa đổi cookie của trình duyệt
  - Phát hiện trình duyệt và hệ điều hành của người dùng
  - Được sử dụng như ngôn ngữ hướng đối tượng
  - Xử lý các trường hợp ngoại lệ
  - Thực hiện các cuộc gọi máy chủ không đồng bộ (AJAX)

Mã javascript trong HTML

- **Phần tử <script>:** Mã javascript đặt trong cặp thẻ **<script>**

```
<head>
        <Script language=”JavaScript”>
                Javascript Comments
        </script>
</head>
```

Mã javascript trong HTML

- **Phần tử <script>:** có thể đặt ở bất kỳ vị trí nào trong trang HTML, nhưng tốt nhất là đặt trong phần **<head>**

- **Ví dụ:**

```
<head>
    <script>
      var x = 5;
      var y = 10;
      var sum = x + y;
      document.write(sum); //in giá trị của biến
    </script>
 </head>
```

Tạo tập tin JavaScript

- **Tập tin Javascript**
  - Mã Javascript có thể đặt trong tập tin có phần mở rộng **.js**
  - Không chứa phần tử <script>
- **Cách truy cập tập tin JavaScript**
  - Trong phần <Head> của trang HTML, tạo liên kết đến tập tin Javascript

```
<script src=”fileJavascript.js” type="text/javscript" >
                JavaScript program
</script>
```

Thực thi mã Javascript

- **Mã JavaScript thực thi** trong quá trình **tải trang** hoặc khi **trình duyệt kích hoạt một sự kiện**
  - Tất cả các câu lệnh được thực thi khi tải trang
  - Một số câu lệnh chỉ xác định các hàm có thể được gọi sau này
  - Mã hoặc lệnh gọi hàm có thể được đính kèm dưới dạng "trình xử lý sự kiện" thông qua thuộc tính thẻ
  - Được thực thi khi **sự kiện** được **trình duyệt kích hoạt**

```
<img src="logo.gif" onclick="alert('clicked!')" />
```

Thực thi mã Javascript

- Ví dụ:

```
<html>
<head>
   <script type="text/javascript">
     function test (message) {
       alert(message);
     }
   </script>
</head>
<body>
  <img src="logo.gif“ onclick="test('clicked!')" />
</body>
</html>
```

Thực thi mã Javascript

- Ví dụ:

```
<html>
<head>
  <script src="sample.js type="text/javascript">
  </script>
</head>
<body>
  <button onclick="sample()" value="Call JavaScript
    function from sample.js" />
</body>
</html>
```

Tập tin js

```
function sample() {
  alert('Hello from sample.js!')
}
```

CÚ PHÁP CỦA JAVASCRIPT

Cú pháp trong Javascript

- **Cú pháp JavaScript tương tự như C # và Java**
  - Toán tử (+, *, =,! =, &&, ++,…)
  - Biến
  - Câu lệnh điều kiện (if, else)
  - Vòng lặp (for, while)
  - Mảng (my_array [ ]) và mảng kết hợp (my_array ['abc’])
  - Các hàm (có thể trả về giá trị)
  - Các biến hàm (like the C# delegates)

Kiểu dữ liệu

- **Kiểu dữ liệu trong JavaScript**:
  - **Numbers** (integer, floating-point)
  - **Boolean** (true / false)
  - **String type** – string of characters

```
var myName = "You can use both single or
double quotes for strings";
```

Kiểu dữ liệu

- **Kiểu dữ liệu trong JavaScript**:
  - **Arrays**

```
var my_array = [1, 5.3, "aaa"];
```

  - **Associative arrays** (hash tables)

```
var my_hash = {a:2, b:3, c:"text"};
```

Đối tượng trong Javascript

- **Đối tượng:** Mọi biến đều có thể được xem là đối tượng

  Ví dụ chuỗi và mảng có các **hàm thành viên**

```
var test = "some string";
alert(test[7]); // shows letter 'r'
alert(test.charAt(5)); // shows letter 's'
alert("test".charAt(1)); //shows letter 'e'
alert("test".substring(1,3)); //shows 'es'
```

```
var arr = [1,3,4];
alert (arr.length); // shows 3
arr.push(7); // appends 7 to end of array
alert (arr[3]); // shows 7
```

Hàm và toán tử của kiểu dữ liệu String

- **Toán tử + :** nối các chuỗi
  - **Ví dụ**:

```
string1 = "fat ";
string2 = "cats";
alert(string1 + string2);  // fat cats
alert("9" + 9);  // 99
```

- **Hàm chuyển đổi chuỗi thành số**:
  - `parseInt`
  - `parseFloat`

```
alert(parseInt("9") + 9);  // 18
```

Hàm và toán tử của kiểu dữ liệu String

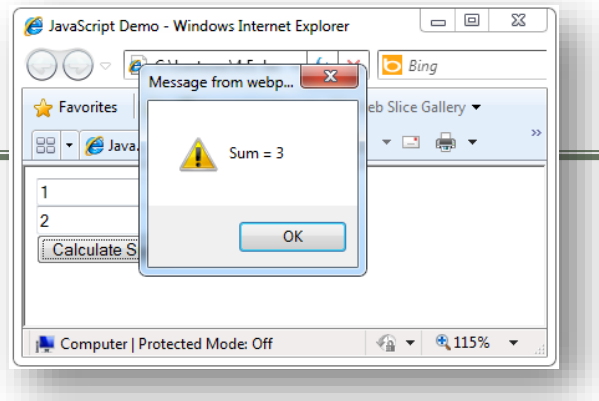
- Ví dụ:

```
<html>
<head>
  <title>JavaScript Demo</title>
  <script type="text/javascript">
    function calcSum() {
      value1 = parseInt(document.mainForm.textBox1.value);
      value2 = parseInt(document.mainForm.textBox2.value);
      sum = value1 + value2;
      document.mainForm.textBoxSum.value = sum;
    }
  </script>
</head>
```

Hàm và toán tử của kiểu dữ liệu String

- Ví dụ (tt):

```
<body>
  <form name="mainForm">
    <input type="text" name="textBox1" /> <br/>
    <input type="text" name="textBox2" /> <br/>
    <input type="button" value="Process"
      onclick="javascript: calcSum()" />
    <input type="text" name="textBoxSum“readonly="readonly"/>
  </form>
</body>
```



Đối tượng Array

- **Khai báo Array**
  - Khai báo mảng trống mới:

```
var arr = new Array();
```

  - Khai báo một mảng chứa một vài phần tử:

```
var arr = [1, 2, 3, 4, 5];
```

- **Thêm một phần tử / nhận phần tử cuối cùng:**

```
arr.push(3);
var element = arr.pop();
```

Đối tượng Array

- **Thao tác trên Array**
  - **Đọc số phần tử** (độ dài mảng):

```
arr.length;
```

  - **Tìm chỉ số của phần tử trong mảng**:

```
arr.indexOf(1);
```

Hộp thoại

- **Hộp Alert:** với văn bản và nút [OK]

```
alert(«Message");
```

- **Hộp Confirmation**: Chứa văn bản, nút [OK] và [Cancel]:

```
confirm(«Message");
```

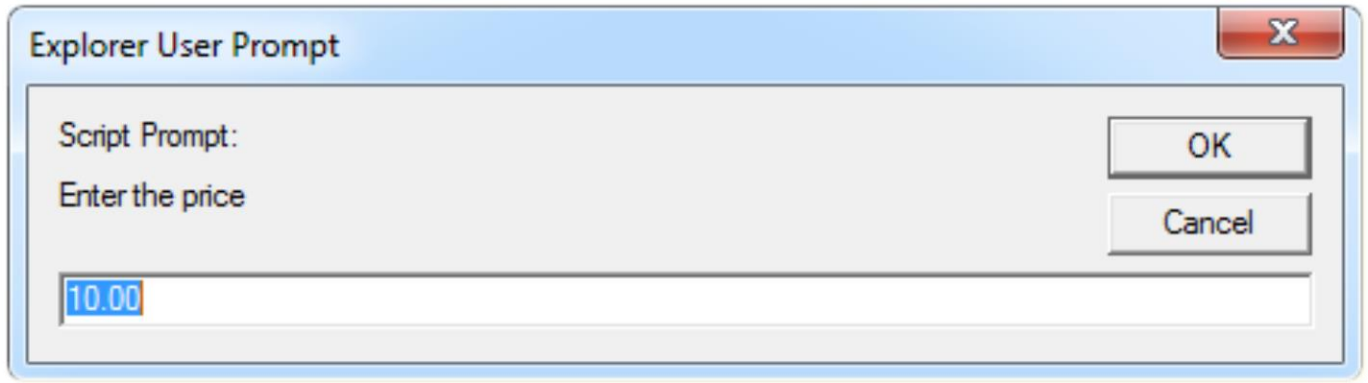
Hộp thoại

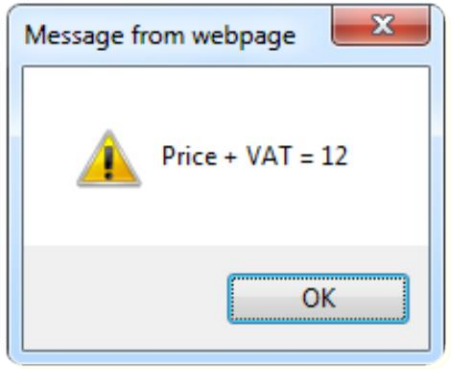
- **Hộp Prompt**: Chứa văn bản và ô nhập với giá trị mặc định

```
prompt («Message", default value);
```

- Ví dụ:

```
price = prompt("Enter the price", "10.00");
alert('Price + VAT = ' + price * 1.2);
```





CẤU TRÚC ĐIỀU KHIỂN TRONG JAVASCRIPT

Cấu trúc điều khiển

- **Cấu trúc điều khiển:** Bất kỳ một ngôn ngữ lập trình nào cũng có các câu lệnh điều khiển, dùng để thực thi các dòng code theo các yêu cầu được chỉ định. **Cấu trúc điều khiển** trong Javacript gồm các lệnh:
  - **Cấu trúc lựa chọn**:
    - **If**
    - **switch,**
  - **Cấu trúc lặp**:
    - **For**
    - **While**
    - **do while**

Cấu trúc điều khiển

- **Khối lệnh trong Javascript**
  - Nhiều câu lệnh Javascript có thể nhóm với nhau tạo ra khối lệnh
  - Các khối lệnh được nhóm bằng cặp dấu ngoặc nhọn { }, các khối thường sử dụng trong các câu lệnh điều khiển rẽ nhánh, vòng lặp

```
{
        //Các dòng lệnh trong khối
}
```

Câu lệnh if, if ... else

- **Câu lệnh if:** nếu **điều kiện** là **true** thì **thi hành các lệnh** trong khối, nếu **điều khiện** là **false** thì khối lệnh bị bỏ qua,
  - **Cú pháp**:

```
if (điều_kiện)
{
        //Khối lệnh
}
```

  - Ví dụ:

```
var n= 10;
if (n>0)
{
    alert(“n là số dương;
}
```

Câu lệnh if, if ... else

- **Câu lệnh if … else:** Nếu **biểu thức điều kiện** là **true** thi hành **khối lệnh 1**, nếu **false** thì thi hành **khối lệnh 2**
  - **Cú pháp**

```
if (biểu thức điều kiện)
{
      khối lệnh 1
}
 else
{
      khối lệnh 2
}
```

Câu lệnh if, if ... else

- **Câu lệnh if … else:**
  - Ví dụ:

```
var year = 2018; //Năm nhuận chia hết cho 400 hoặc 4
nhưng không chia hết cho 100
if((year % 400 == 0)||((year%100!=0) && (year%4==0)))
{
      alert(year + " là năm nhuận.");
}
Else
{
      alert(year + " không là năm nhuận.");
}
```

Câu lệnh if, if ... else

- **Câu lệnh else if:**
  - **else if** sẽ tạo ra câu lệnh điều kiện **if** mới nếu điều kiện trước đó **false**
  - **Ví dụ:**

```
var course = 1;
if (course == 1)
{
        document.write("HTML Tutorial");
}
else if (course == 2)
{
        document.write("CSS Tutorial");
}
 else
{
        document.write("JavaScript Tutorial");
}
```

Câu lệnh if, if ... else

- **Câu lệnh If… else rút gọn:** có thể rút gọn câu lệnh if...else bằng **toán tử 3 ngôi**
  - **Cú pháp:**

```
Điều kiện ?Lệnh 1: lệnh 2
```

  - **Ví dụ:**

```
a = 2;
a%2==0 ? Document.write("a là số chẵn"): Document.write("a là số lẻ")
```

Câu Lệnh if, if ... else

- **Lệnh else if**
  - **Ví dụ:** cho 3 cạnh của tam giác, kiểm tra 3 cạnh có tạo thành tam giác không
  - Nếu là tam giác, thì xuất ra loại tam giác
    - Nếu a=b=c: tam giác đều
    - Nếu a=b hoặc a=c hoặc b=c: tam giác cân
    - Ngược lại là tam giác thường

Câu Lệnh if, if ... else

- **Lệnh else if**
  - Khai báo và gán giá trị cho biến diemTB
  - Kiểm tra diemTB phải >=0 và diemTB<=10
  - Xét nếu diemTB >=8 thì xếp loại giỏi
  - Nếu diemTB>=6.5 thì xếp loại khá
  - Nếu diemTB>=5 thì xếp loại trung bình
  - Nếu diemTB<5 thì xếp loại yếu

Câu Lệnh switch

- **Lệnh switch**: Trong trường hợp có nhiều điều kiện khác nhau thì dùng câu lệnh switch.

- **Cú pháp:**

```
switch (biểu thức)
{
        case n1: thực thi lệnh 1 biểu thức= n; break;
        case n2: thực thi lệnh 1 biểu thức= n; break;
        default: thực thi lệnh mặc định nếu biểu thức không bằng giá
                 trị nào trong các case
}
```

Câu Lệnh switch

- **Lệnh switch**
  - Ví dụ

```
var diem = prompt("Hãy nhập điểm:")
switch (diem)
{
   case "A": alert(“Xuất sắc!");break;
   case "B": case "C": alert("Khá!");break;
   case "D": case "E": case "F": alert(“Trung bình”); break;
   default: alert("Giá trị nhập vào không hợp lệ.");
}
```

Câu Lệnh switch

- **Lệnh switch**
  - Bài tập: khai báo và gán giá trị cho 1 biến **thang**
  - Xuất ra số ngày trong tháng
    - Tháng 2: “tháng 2 có 28 ngày
    - …

Câu lệnh lặp For

- **Câu lệnh For**: lặp một đoạn code với số lần lặp xác định.
- **Cú pháp**:

```
for (khởitạo ; điềukiện ; bướctăng)
{
      //khối lệnh lặp
}
```

- Ví dụ:

```
for (i=1; i<=10; i++)
{
      document.write(5 +"*" + i+ "=" + (5*i) + "<br >");
}
```

Câu lệnh lặp For

- **Câu lệnh for ... In**: duyệt qua các khóa của đối tượng đếm được (không dùng cho mảng)
  - **Cú pháp**:

```
for(n in object)
{
    // khối lệnh
}
```

  - Ví dụ:

```
var fruits = ["Apple", "Banana", "Mango", "Orange", "Papaya"];
for(var i in fruits)
{
    document.write("<p>" + fruits[i] + "</p>");
}
```

Câu lệnh lặp **while**

- **Câu lệnh while**: thực hiện lặp đoạn code khi điều kiện đúng, lệnh while sẽ dừng khi điều kiện sai. Điều kiện được kiểm tra trước khi đoạn code được thực hiện.
  - **Cú pháp**

```
while (điềukiện )
{
        //khối lệnh
}
```

Câu lệnh lặp while

- **Câu lệnh while**
  - Ví dụ:

```
let text = "";
let i = 0;
while (i < 10)
{
  text += "<br>The number is " + i;
  i++;
}
```

The number is 0
The number is 1
The number is 2
The number is 3
The number is 4
The number is 5
The number is 6
The number is 7
The number is 8
The number is 9

Câu lệnh lặp do …while

- **Câu lệnh do … while**: tương tự câu lệnh while nhưng khối lệnh được thự thi trước, sau đó kiểm tra điều.
  - **Cú pháp**:

```
do
{
        //Khối lệnh
} while (condition);
```

- Ví dụ:

```
var i = 1;
do
{
   document.write("<p>The number is"+ i+ "</p>");
   i++;
} while(i <= 5);
```

The number is 1

The number is 2

The number is 3

The number is 4

The number is 5

HÀM TRONG JAVASCRIPT

Hàm trong JavaScript

- **Hàm** là một khối lệnh dùng để thực thi một chức năng cụ thể. Hàm được gọi thực thi nhiều lần.
- **Định nghĩa hàm trong Javascript**

```
function name_funtion()
{
        //khối lệnh của hàm
 }
```

Hàm trong JavaScript

- **Gọi hàm:** Để thực thi hàm, cần phải gọi hàm.
- **Cách gọi hàm**: **name_funtionn(Argument_list)**
  - **Ví dụ:** tạo hàm

```
function myFunction()
{
    alert("Calling a Function!");
}
```

  - **Gọi hàm**

```
myFunction(); //Hiện thông báo:  "Calling a Function!"
```

Hàm trong JavaScript

- **Tham số của hàm:** hoạt động như các biến giữ chỗ trong một hàm; khi gọi hàm, cần cung cấp **giá trị** (được gọi là đối số) cho các tham số để hàm thực thi.
- **Cú pháp**

```
function functionName(parameter1, parameter2, …)
{
    // khối lệnh của hàm
}
```

Hàm trong JavaScript

- **Tham số của hàm:**
  - Ví dụ:

```
function displaySum(num1, num2)
{
      var total = num1 + num2;
      alert(total);
}
// Calling function
displaySum(6, 20); // 0utputs: 26
displaySum(-5, 17); // 0utputs: 12
```

Hàm trong JavaScript

- **Giá trị mặc định cho các tham số của hàm**
  - Nếu không có đối số nào được cung cấp cho hàm khi nó được gọi thì các giá trị tham số mặc định này sẽ được sử dụng.
  - Ví dụ:

```
function sayHello(name = 'Guest’)
{
      alert('Hello, ' + name);
}
sayHello(); // 0utputs: Hello, Guest
sayHello('John'); // 0utputs: Hello, John
```

Hàm trong JavaScript

- **Giá trị trả về của hàm**
  - Một hàm có thể trả về một giá trị bằng cách sử dụng câu lệnh **return**.
  - Giá trị trả về có thể thuộc **kiểu dữ liệu bất kỳ**, bao gồm cả **mảng** và **đối tượng**.
  - Câu lệnh **return** thường được đặt ở dòng cuối cùng của hàm.

Hàm trong JavaScript

- **Giá trị trả về của hàm**
  - Ví dụ:

```
function getSum(num1, num2)
{
    var total = num1 + num2;
    return total;
}
```

  - Gọi hàm:

```
// Displaying returned value
alert(getSum(6, 20)); // 0utputs: 26
alert(getSum(-5, 17)); // 0utputs: 12
```

Hàm trong JavaScript

- **Biểu thức hàm**: Cú pháp tạo một biểu thức hàm tương tự khai báo hàm thông thường nhưng **khai báo bằng biểu thức.**
- **Biểu thức hàm** có thể **có tên** hoặc là **không có tên**, nếu không có tên thì biểu thức hàm này gọi là hàm ẩn danh (anonymous).

Hàm trong JavaScript

- **Biểu thức hàm:** Khi biểu thức hàm đã được lưu trữ trong một biến, biến đó có thể được sử dụng như một hàm
  - Ví dụ:

```
var getSum = function(num1, num2)
{
      var total = num1 + num2;
      return total;
};
alert(getSum(5, 10)); // 0utputs: 15
var sum = getSum(7, 25);
alert(sum); // 0utputs: 32
```

Hàm trong JavaScript

- **Biểu thức hàm:** Cú pháp của **khai báo hàm và biểu thức hàm** trông rất giống nhau, nhưng chúng khác nhau về cách chúng được đánh giá

- Ví dụ:

```
declaration(); // Outputs: Hi, I'm a function declaration!
function declaration() {
      alert("Hi, I'm a function declaration!");
}
expression();//Uncaught TypeError:undefined is not a function
var expression = function() {
       alert("Hi, I'm a function expression!");
};
```

Phạm vi của biến

- **Phạm vi của biến:** nơi biến có thể được sử dụng hoặc truy cập. Biến có thể khai báo ở bất kỳ đâu trong chương trình. Tuy nhiên, **vị trí của khai báo** xác định mức độ sẵn có của một biến trong chương trình JavaScript
- **Biến cục bộ**: các biến được khai báo trong một hàm, và không thể truy cập từ bên ngoài hàm đó.

Phạm vi của biến

- Ví dụ:

```
function greetWorld()
{
      var greet = "Hello World!";
      alert(greet);
}
greetWorld(); // Outputs: Hello World!
alert(greet); // lỗi: greet không tìm thấy
```

Phạm vi của biến

- **Biến toàn cục:**
  - Biến được khai báo trong một chương trình **bên ngoài** các hàm đều có **phạm vi toàn cục**, nó sẽ có sẵn cho tất cả các tập lệnh.

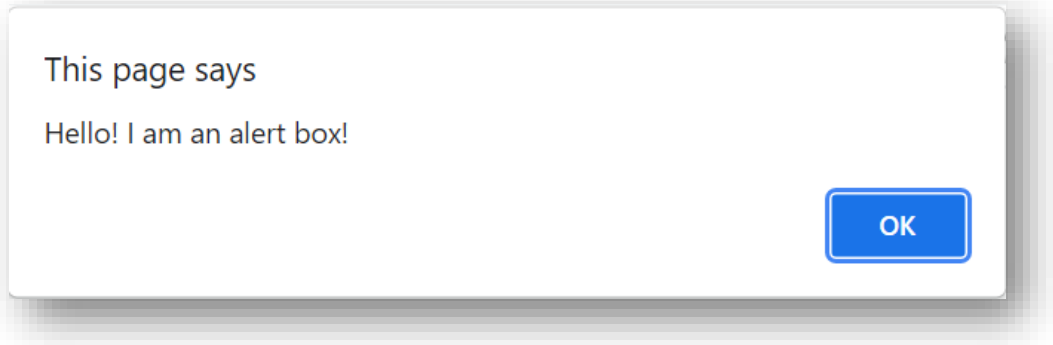
Ví dụ:

```
var greet = "Hello World!"; //biến toàn cục
// Defining function
function greetWorld()
{
      alert(greet);
}
greetWorld(); // Outputs: Hello World!
alert(greet); // Outputs: Hello World!
```

Các hàm trong Javascript

- **Hàm alert():** hiển thị hộp thông báo, một tham số truyền vào là nội dung hiển thị trên hộp thông báo.
- Ví dụ:

```
<script>
    alert("Hello! I am an alert box!");
</script>
```

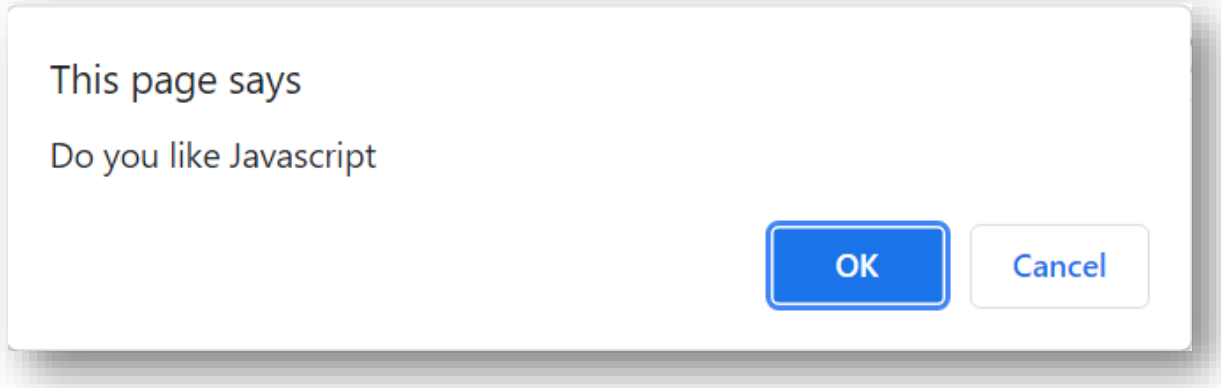


Các hàm trong Javascript

- **Hàm confirm()** hiển thị hộp **thông báo**, có hai lựa chọn là Yes và No, nếu người dùng chọn Yes thì hàm trả về TRUE và nếu chọn NO thì hàm sẽ trả về FALSE. Hàm có một tham số truyền vào là nội dung thông báo.

- Ví dụ:

```
<script language="javascript">
    confirm("Do you like Javascript");
</script>
```



Các hàm trong Javascript

- **Hàm prompt():** lấy thông tin từ người dùng, hàm gồm có hai tham số truyền vào là nội dung thông báo và giá trị ban đầu. Nếu người dùng không nhập vào thì giá trị trả về là NULL

- Ví dụ:

```
<script language="javascript">
    var t = prompt("Nhập tên của bạn", '');
    alert(t);
</script>
```

DOCUMENT OBJECT MODEL (DOM)

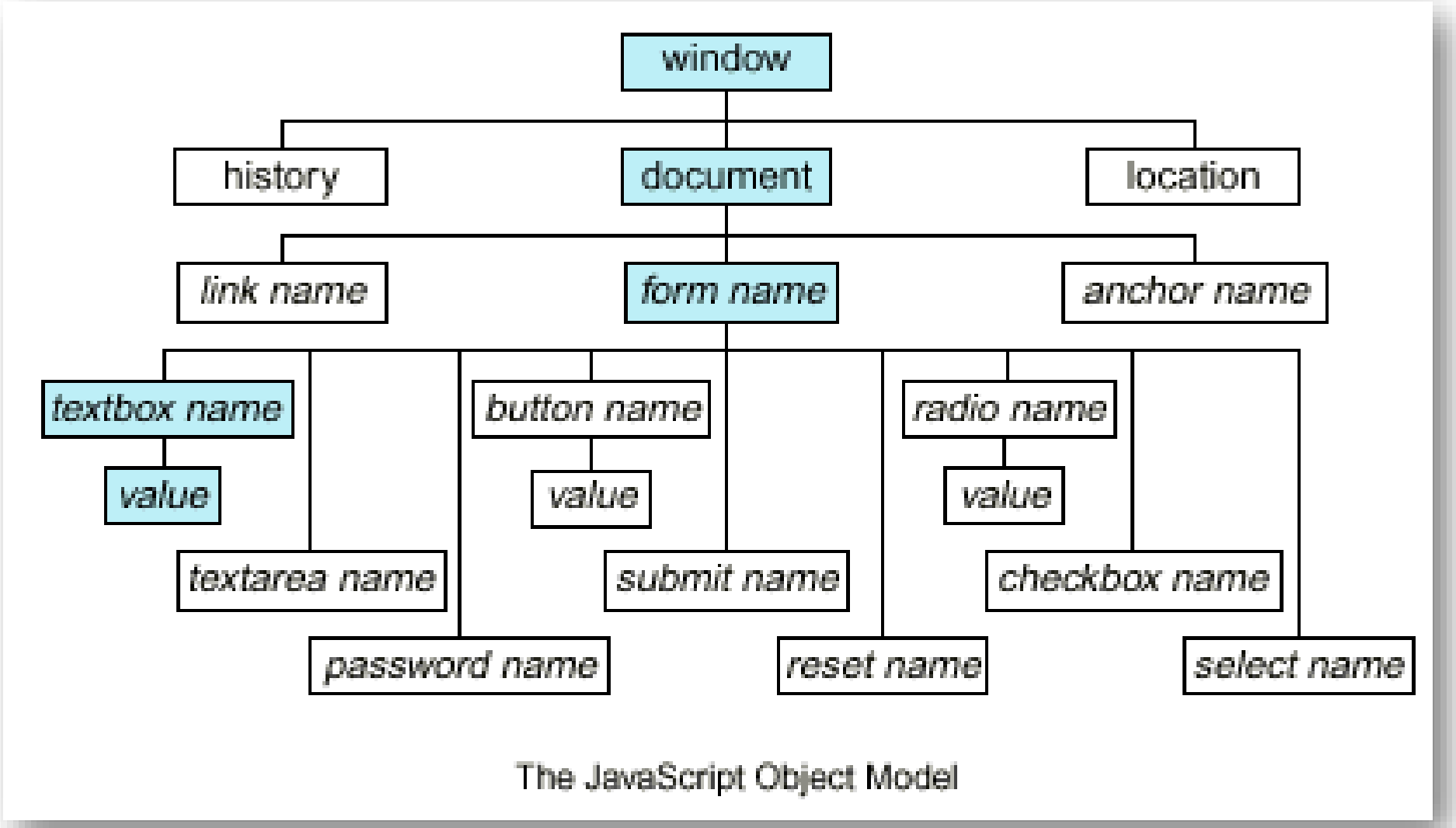
Tổng quan về DOM

- **Mỗi trang web** hiển thị trên cửa sổ trình duyệt được coi là một đối tượng.
- **Đối tượng Document:** đại diện cho **tài liệu HTML** được hiển thị trong cửa sổ trình duyệt. Đối tượng Document có nhiều thuộc tính tham chiếu đến các đối tượng khác cho phép truy cập và sửa đổi nội dung tài liệu.
- Cách mà nội dung tài liệu được truy cập và sửa đổi được gọi là **Mô hình Đối tượng Tài liệu - DOM**. Các đối tượng được tổ chức theo một hệ thống phân cấp. Cấu trúc phân cấp này áp dụng cho việc tổ chức các đối tượng trong tài liệu Web.

Tổng quan về DOM

- **Mô hình Đối tượng Tài liệu - DOM**
  - **Window object**: Trên cùng của hệ thống phân cấp, là phần tử ngoài cùng của hệ thống phân cấp đối tượng.
  - **Document object**: tài liệu HTML được tải trên cửa sổ trình duyệt là một đối tượng tài liệu. Tài liệu chứa nội dung của trang.
  - **Form object: <form> ... </form>:** chứa tất cả các phần tử được xác định cho đối tượng đó như ext fields, buttons, radio buttons, and checkboxes.

Tổng quan về DOM



Tổng quan về DOM

- **Tính chất của DOM**:
  - Mọi phần tử HTML đều có thể truy cập được qua JavaScript DOM API
  - Hầu hết các đối tượng DOM có thể được điều khiển bởi lập trình viên
  - Mô hình sự kiện cho phép một tài liệu phản ứng khi người dùng làm điều gì đó trên trang
- **Thuận lợi**
  - Tạo các trang tương tác
  - Cập nhật các đối tượng của một trang mà không cần tải lại nó

Truy cập các phần tử của DOM

- **Truy cập các phần tử thông qua thuộc tính ID**

```
var elem = document.getElementById("some_id")
```

- **Qua thuộc tính name**

```
var arr = document.getElementsByName("some_name")
```

- **Qua tên thẻ**

```
var imgTags = el.getElementsByTagName("img")
```

Thao tác với các phần tử trên DOM

- Khi truy cập vào một phần tử trong DOM, thì có thể lấy giá trị hoặc gán giá trị cho các thuộc tính của phần tử đó
- **Thuộc tính của các phần tử của DOM** đều bắt nguồn từ các thuộc tính của thẻ HTML:
  - **id, name, href, alt, title, src,** etc…
  - **thuộc tính style:** hiệu chỉnh CSS của phần tử, chính là inline Style.
  - Một số thuộc tính được nhúng hoặc bên ngoài: `style.width, style.marginTop, style.backgroundImage`

Thao tác với các phần tử trên DOM

- Ví dụ:

```
function change(state) {
  var lampImg = document.getElementById("lamp");
  lampImg.src = "lamp_" + state + ".png";
  var statusDiv = document.getElementById("statusDiv");
  statusDiv.innerHTML = "The lamp is " + state";
}
…
<img src="test_on.gif" onmouseover="change('off')"
  onmouseout="change('on')" />
```

Thao tác với các phần tử trên DOM

- **class = className:** chỉ định các phần tử HTML thuộc 1 lớp.
- **innerHTML:** giữ tất cả toàn bộ mã HTML bên trong phần tử
- **Read-only:** trạng thái của phần tử
  - `tagName`, `offsetWidth`, `offsetHeight`, `scrollHeight`, `scrollTop`, `nodeType`, etc…

Thao tác với các phần tử trên DOM

- **Truy cập các phần tử thông qua cấu trúc cây DOM:** có thể truy cập các phần tử trong DOM thông qua một số thuộc tính thao tác trên cấu trúc của mô hình DOM
  - `element.childNodes`
  - `element.parentNode`
  - `element.nextSibling`
  - `element.previousSibling`
  - `element.firstChild`
  - `element.lastChild`

Thao tác với các phần tử trên DOM

- **Truy cập các phần tử thông qua cấu trúc cây DOM:**

  Ví dụ:

```
var el = document.getElementById('div_tag');
alert (el.childNodes[0].value);
alert (el.childNodes[1].getElementsByTagName('span').id);
…
<div id="div_tag">
  <input type="text" value="test text" />
  <div>
    <span id="test">test span</span>
  </div>
</div>
```

HTML DOM EVENT

HTML DOM Event

- **HTML DOM event** cho phép JavaScript thiết lập các trình xử lý sự kiện khác nhau trên các phần tử trong tài liệu HTML.
  - Sự kiện được trình duyệt kích hoạt và được gửi đến hàm xử lý sự kiện JavaScript được chỉ định
  - Có thể được đặt bằng các thuộc tính HTML
  - Có thể được truy cập thông qua DOM

```
<img src="test.gif" onclick="imageClicked()" />
```

```
var img = document.getElementById("myImage");
img.onclick = imageClicked;
```

HTML DOM Event

- **Tham số của các trình xử lý sự kiện**: cung cấp thông tin về sự kiện gồm
  - Loại sự kiện (nhấp chuột, nhấn phím, v.v.)
  - Dữ liệu về vị trí nơi sự kiện đã được kích hoạt (ví dụ: tọa độ chuột)
  - Giữ một tham chiếu đến người gửi sự kiện

    Ví dụ. nút đã được nhấp
  - Lưu trữ thông tin về trạng thái của các phím [Alt], [Ctrl] và [Shift]. Một số trình duyệt không gửi đối tượng này, nhưng đặt nó trong **document.event**

Các sự kiện thông dụng

- **Mouse events:**
  - `onclick`, `onmousedown`, `onmouseup`
  - `onmouseover`, `onmouseout`, `onmousemove`
- **Key events:**
  - `onkeypress`, `onkeydown`, `onkeyup`
  - Only for input fields
- **Interface events:**
  - `onblur`, `onfocus`
  - `onscroll`

Các sự kiện thông dụng

- **Form events**
  - onchange – for input fields
  - onsubmit
    - Cho phép hủy gửi biểu mẫu
    - Hữu ích cho việc xác thực biểu mẫu
- **Miscellaneous events**
  - onload, onunload
    - Chỉ được phép cho phần tử <body>
    - Kích hoạt khi tất cả nội dung trên trang đượcloaded / unloaded

Các sự kiện thông dụng

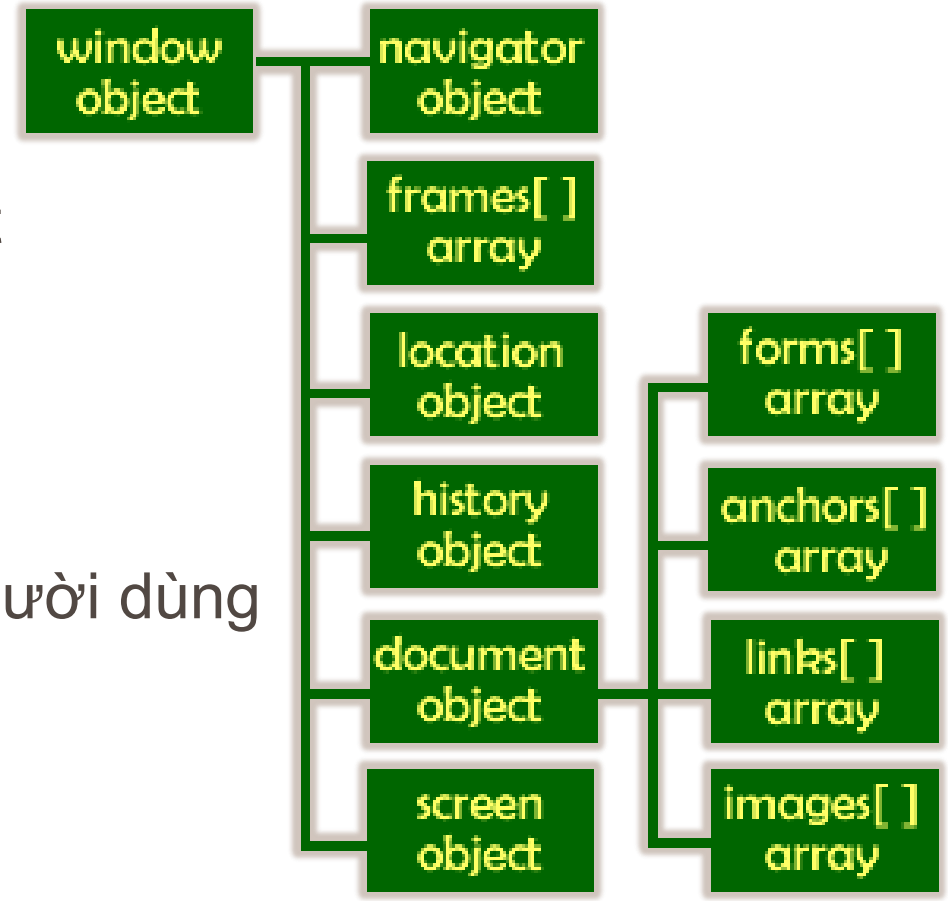
- Ví dụ:

```
<html>

<head>
  <script type="text/javascript">
    function greet() {
      alert("Loaded.");
    }
  </script>
</head>

<body onload="greet()" >
</body>

</html>
```

THE BUILT-IN BROWSER OBJECTS

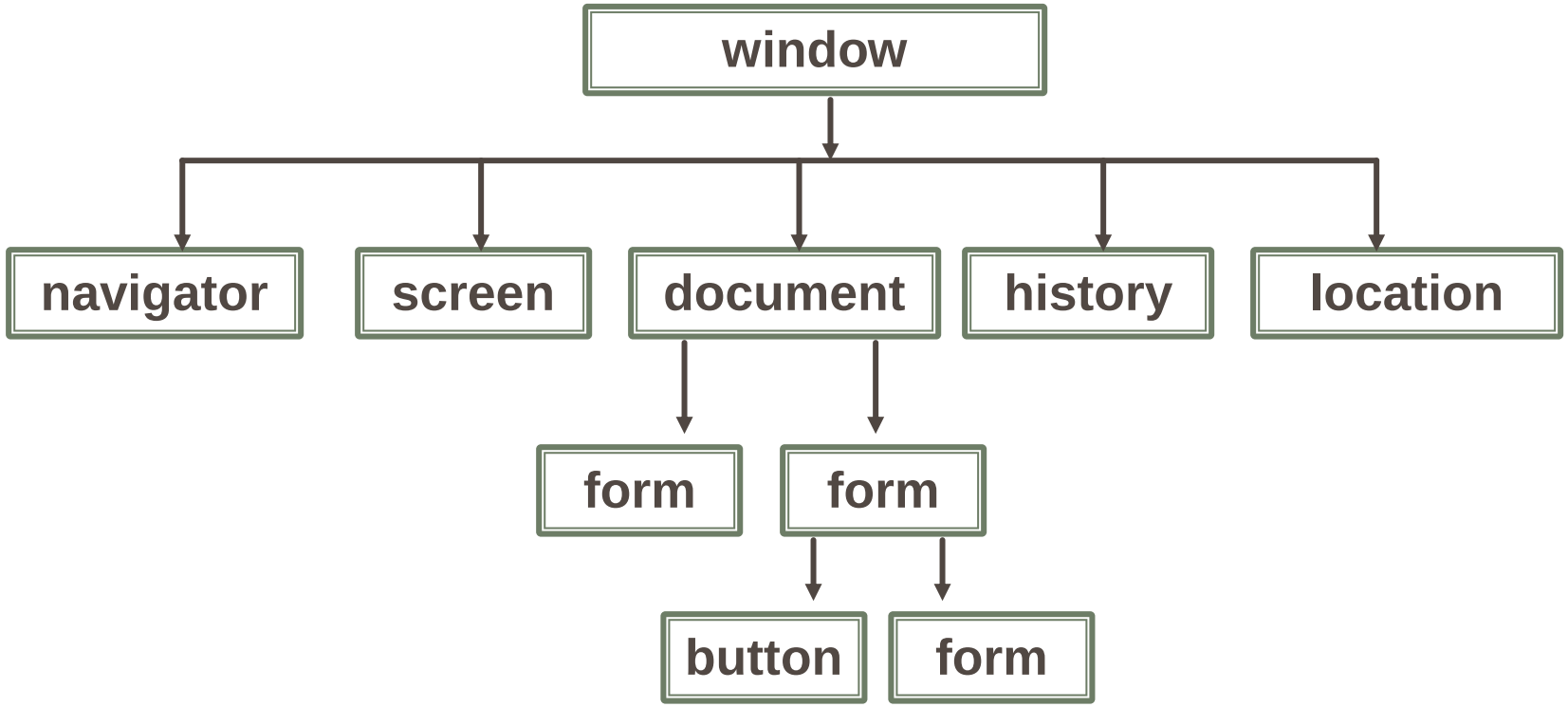
Built-in Browser Objects

- **Trình duyệt cung cấp một số dữ liệu chỉ đọc qua**:
  - **window**
    - Nút trên cùng của DOM
    - Đại diện cho cửa sổ của trình duyệt
  - **document**
    - Chứa thông tin về tài liệu được tải
  - **screen**
    - Chứa các thuộc tính hiển thị của người dùng
  - **browser**
    - Chứa thông tin của trình duyệt



Built-in Browser Objects

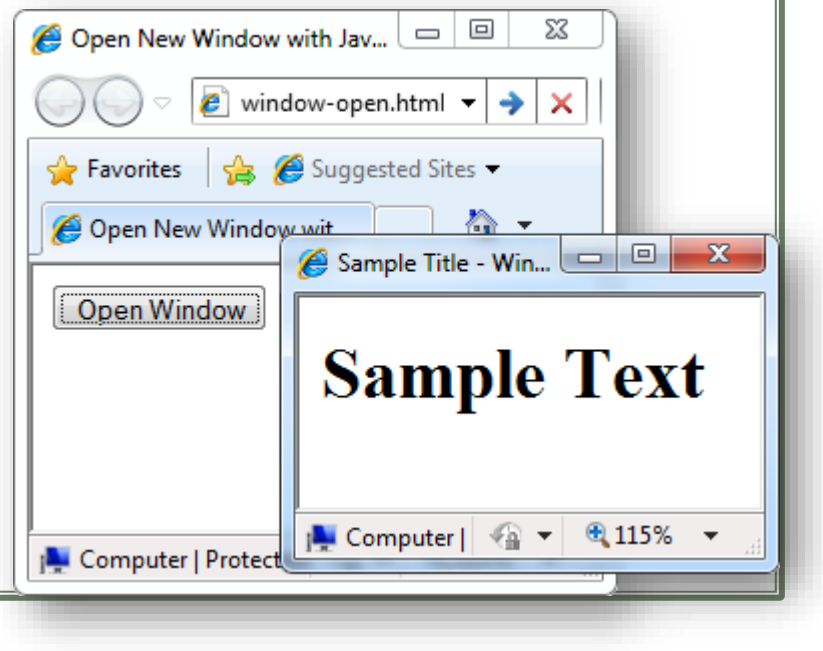
- Ví dụ: **DOM Hierarchy**



Đối tượng trình duyệt tích hợp

- Ví dụ: **Window.open()**

```
var newWindow = window.open("", "sampleWindow",
  "width=300, height=100, menubar=yes, status=yes,
resizable=yes");
newWindow.document.write(
  "<html><head><title>
  Sample Title</title>
  </head><body><h1>Sample
  Text</h1></body>");
newWindow.status =
  "Hello folks";
```



Navigator Object

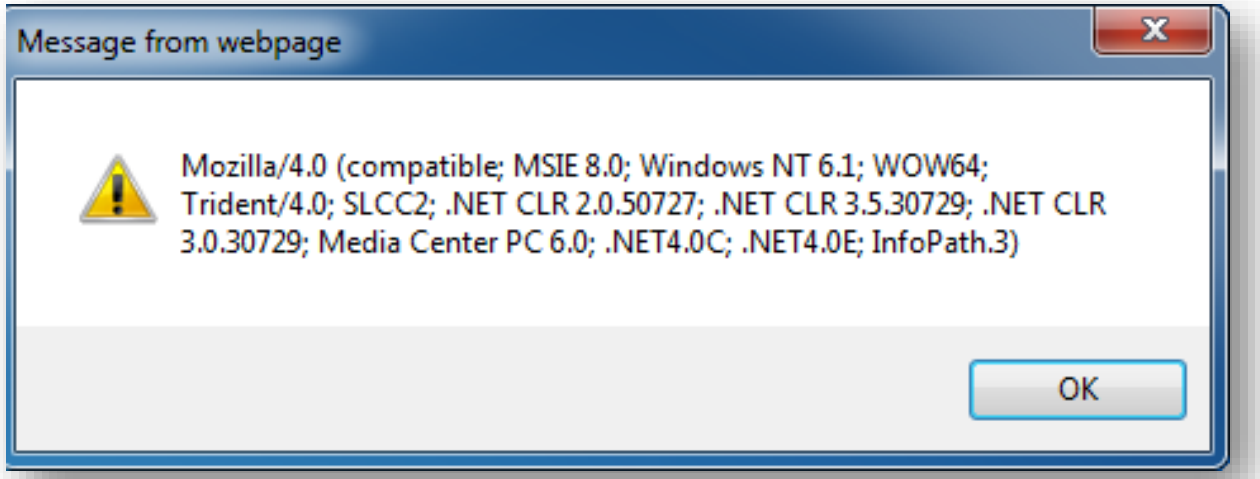
- **Đối tượng navigator** chứa thông tin về trình duyệt.

```
alert(window.navigator.userAgent);
```

The browser window

The navigator in the browser window

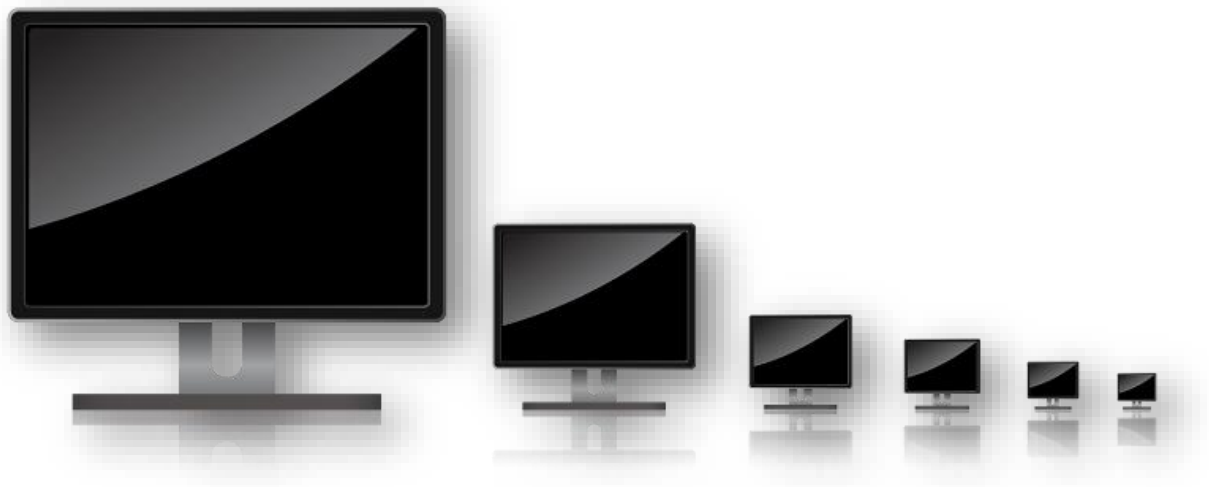
The userAgent (browser ID)



Screen Object

- **Đối tượng screen** chứa thông tin về màn hình
- Ví dụ:

```
window.moveTo(0, 0);
x = screen.availWidth;
y = screen.availHeight;
window.resizeTo(x, y);
```



Document and Location

- **Đối tượng document**
  - Cung cấp một số built-in arrays của các đối tượng cụ thể trên trang Web hiện đang được tải
  - **Ví dụ**:

```
document.links[0].href = "yahoo.com";
document.write("This is some <b>bold text</b>");
```

- **document.location**
  - Truy cập **URL** hiện đang mở hoặc chuyển hướng trình duyệt

```
document.location = "http://www.yahoo.com/";
```

Form Validation

- **Việc xác thực HTML form** được thực hiện bằng JavaScript.
  - Nếu **form field (fname)** trống, hàm sẽ cảnh báo một thông báo và trả về false để ngăn form được **submit**
- **Ví dụ**:

Form Validation

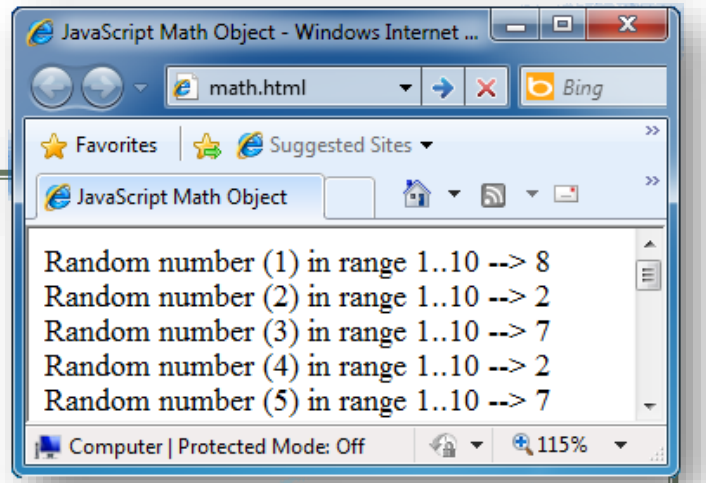
```
function checkForm()
{
  var valid = true;
  if (document.mainForm.firstName.value == "") {
    alert("Please type in your first name!");
    document.getElementById("firstNameError").
      style.display = "inline";
    valid = false;
  }
  return valid;
}
…
<form name="mainForm" onsubmit="return checkForm()">
  <input type="text" name="firstName" />
  …
</form>
```

Math Object

- **Đối tượng Math** cung cấp một số hàm toán học

Ví dụ:

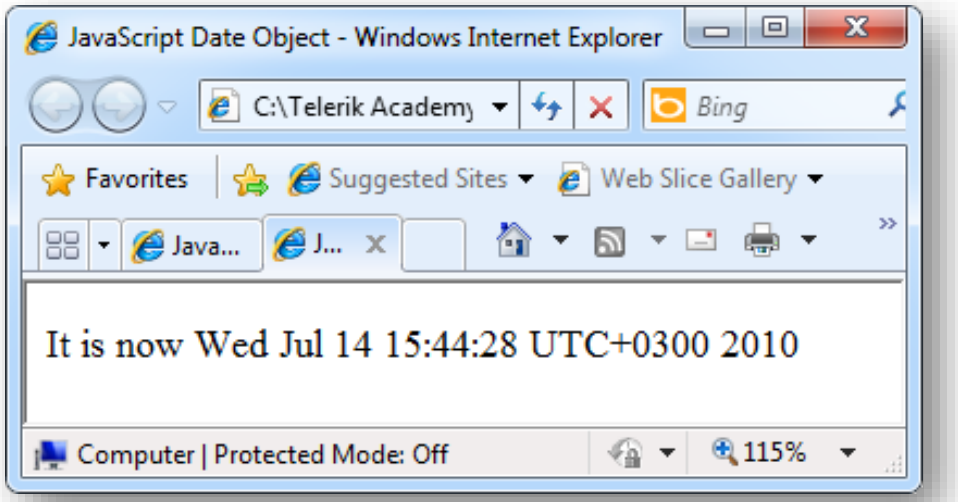
```
for (i=1; i<=20; i++) {
  var x = Math.random();
  x = 10*x + 1;
  x = Math.floor(x);
  document.write("Random number("+ i+ ") in range " +
    "1..10 --> " + x + "<br/>");
}
```



The Date Object

- **Đối tượng `Date`** cung cấp các hàm date / calendar

```
var now = new Date();
var result = "It is now " + now;
document.getElementById("timeField").innerText = result;
...
<p id="timeField"></p>
```



Timers - Timing Events

- **Đối tượng window** cho phép thực thi mã trong các khoảng thời gian xác định. Những khoảng thời gian này được gọi là **timing events**. Gồm hai phương thức:
- **setTimeout():** Thực thi hàm, sau số mili giây được chỉ định

```
window.setTimeout(function, milliseconds);
```

  - ***function***: hàm được thực thi.
  - ***milliseconds:*** số mili giây trước khi thực thi.

Timers - Timing Events

- **setTimeout():**

  Ví dụ:

```
var timer = setTimeout('bang()', 5000);
```

5 seconds after this statement executes, this function is called

```
clearTimeout(timer);
```

Cancels the timer

Timers - Timing Events

- **setInterval():** Tương tự như **setTimeout ()**, nhưng lặp lại việc thực thi hàm liên tục.
- Ví dụ:

```
var timer = setInterval('clock()', 1000);
```

This function is called continuously per 1 second.

```
clearInterval(timer);
```

Stop the timer.

Timers - Timing Events

- Ví dụ:

```
<script type="text/javascript">
  function timerFunc() {
    var now = new Date();
    var hour = now.getHours();
    var min = now.getMinutes();
    var sec = now.getSeconds();
    document.getElementById("clock").value =
      "" + hour + ":" + min + ":" + sec;
  }

  setInterval('timerFunc()', 1000);
</script>

<input type="text" id="clock" />
```