P.N.H

Second Edition

JavaScript

cho người mới bắt đầu

Lời nói đầu:

Các bạn đang cầm trong tay cuốn sách "JavaScript cho người mới bắt đầu",nếu như bạn mới làm quen với ngôn ngữ JavaScript thì cuốn sách quả thực rất có ích cho các bạn bởi lối viết đơn giản,dễ hiểu,cộng thêm vào đó là các ví dụ minh họa sát với thực tế.Đây là ấn bản đầu tiên cho nên khó có thể khẳng định rằng nó không mắc một sai sót nào,mọi ý kiến đóng góp xin các bạn vui lòng gửi về hòm thư hieupn89@gmail.com.

Các bạn có thể thông qua hòm thư trên để gửi bài viết,các ví dụ hay về JavaScript cho tác giả,góp phần hoàn thiện cuốn sách hơn nữa trong lần ấn bản sau (các bạn cũng lưu ý rằng cuốn sách này là cuốn sách hoàn toàn phi lợi nhuận – cho nên P.N.H sẽ rất cảm ơn các bạn đóng góp bài viết cho cuốn sách).

Sau cùng P.N.H mong chờ và cảm ơn những góp ý của các bạn,chúc các bạn luôn luôn hạnh phúc và luôn luôn lạc quan yêu đời bạn nhé.Hẹn gặp lại các bạn ở ấn bản lần thứ 3 của cuốn sách.

P.N.H

Mục lục

Chương 1: JS BASIC

Mở đầu về JavaScrip 5
Đặt code JavaScript ở đâu ? 8
Câu lệnh JavaScript ? 10
Sử dụng Comment trong JS ? 11
Biến trong JavaScript. 12
Các phép toán trong JavaScript 13
Các phép toán so sánh và Logic trong JavaScript 15
Cấu trúc IF… ELSE 16
Cấu trúc lựa chọn Switch 19
Các hàm vào ra thông dụng nhất 20
Sử dụng hàm eval() trong JavaScript 23
</html> 23
Hàm(function) trong JavaScript 26
Vòng lặp trong JavaScript 29
Vai trò lệnh Break và Continue trong vòng lặp ? 32
Các lệnh thao tác trên đối tượng 34
Tạo đối tượng trong JavaScript 37
Sự kiện(event) trong JavaScript 42
Try – Catch 44
Lệnh throw 47
Các ký tự đặc biệt trong JavaScript 48
Một số chú ý khi sử dụng JavaScript 49
Objects trong JavaScript 51
String Objects 52
getElementById 58
Làm việc với thuộc tính innerHTML 58
Làm việc với ngày giờ (Date) 60
Tạo đồng hồ bấm giờ bằng JavaScrip 67
Tạo chiếc đồng hồ điện tử cho chính bạn 69
Mảng (Array) 70
Boolean Objects 77
Math Objects 79
Tạo số ngẫu nhiên trong JavaScript 82
So mẫu (RegExp) trong JavaScript 82
Bài thực hành 1: 87
Bài thực hành 2: 92
bài thực hành 3 95
Bài thực hành 4 96
Sự phân cấp đối tượng trong JS 98
JavaScript Navigator 98
javaScript window 100
Location Object 107
Frame Object 112
Bài thực hành với Frame: 116
Document Object 119
Document Object Properties 120

Document Object Methods 120
Image Object 125
Bài thực hành với image: 126
Bài thực hành tạo dao diện Yahoo Hỏi đáp: 131
Kiểm tra tính hợp lệ của giá trị trong form với JavaScript 134
JavaScript Cookie 146
JavaScript Form Validation 155
Một số kỹ thuật hướng đối tượng làm nền tảng cho các Javascript framework 159
JavaScript Animation 166
JavaScript Image map 167
JavaScript SetTimeOut và CleartimeOut 168
JavaScript và lập trình hướng đối tượng- phần 1 173
JavaScript và lập trình hướng đối tượng- phần 2 180
Tóm tắt những điều đã học 187
Mở đầu về DOM HTML 188
HTML DOM nodes 189
HTML DOM node tree 190
HTML DOM Methods 192
Truy cập vào nút trong DOM 194
Sử dụng DOM để xác định thông tin node 200
Bài thực hành về DOM 201
HTML DOM – event 203
BÀI TẬP: 204
LỜI GIẢI: 204
</html> 205

Chương 1

JS BASIC

Mở đầu về JavaScrip

JavaScript là ngôn ngữ kịch bản của web.

Nó ngôn ngữ hướng đối tượng

Nó được sử dụng trên hàng triệu trang web

Để học nó tốt nhất là bạn phải có kiến thức căn bản về HTML

JavaScript là một ngôn ngữ lập trình dựa trên nguyên mẫu với cú pháp phát triển từ C. Giống như C, JavaScript có khái niệm từ khóa, do đó, JavaScript gần như không thể được mở rộng.

Cũng giống như C, JavaScript không có bộ xử lý xuất/nhập (*input/output*) riêng. Trong khi C sử dụng thư viện xuất/nhập chuẩn, JavaScript dựa vào phần mềm ngôn ngữ được gắn vào để thực hiện xuất/nhập.

Trên trình duyệt, rất nhiều trang web sử dụng JavaScript để thiết kế trang web động và một số hiệu ứng hình ảnh thông qua DOM. JavaScript được dùng để thực hiện một số tác vụ không thể thực hiện được với chỉ HTML như kiểm tra thông tin nhập vào, tự động thay đổi hình ảnh,... Ở Việt Nam,

JavaScript còn được ứng dụng để làm bộ gõ tiếng Việt giống như bộ gõ hiện đang sử dụng trên trang Wikipedia tiếng Việt. Tuy nhiên, mỗi trình duyệt áp dụng JavaScript khác nhau và không tuân theo chuẩn W3C DOM, do đó trong rất nhiều trường hợp lập trình viên phải viết nhiều phiên bản của cùng một đoạn mã nguồn để có thể hoạt động trên nhiều trình duyệt. Một số công nghệ nổi bật dòng JavaScript để tương tác với DOM bao gồm DHTML, Ajax và SPA.

Bên ngoài trình duyệt, JavaScript có thể được sử dụng trong tập tin PDF của Adobe Acrobat và Adobe Reader. Điều khiển Dashboard trên hệ điều hành Mac OS X phiên bản 10.4 cũng có sử dụng JavaScript. Công nghệ kịch bản linh động (*active scripting*) của Microsoft có hỗ trợ ngôn ngữ JScript làm một ngôn ngữ kịch bản dùng cho hệ điều hành. JScript .NET là một ngôn ngữ tương thích với CLI gần giống JScript nhưng có thêm nhiều tính năng lập trình hướng đối tượng.

Mỗi ứng dụng này đều cung cấp mô hình đối tượng riêng cho phép tương tác với môi trường chủ, với phần lõi là ngôn ngữ lập trình JavaScript gần như giống nhau.

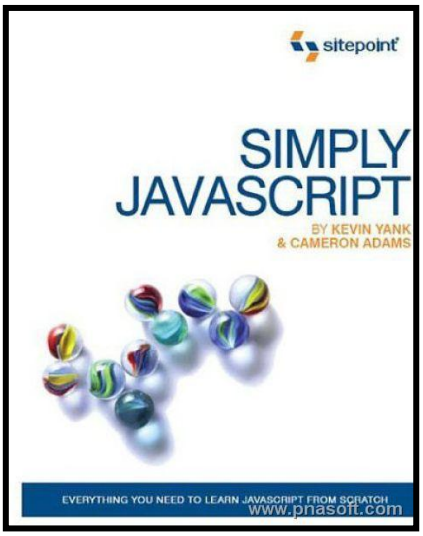
Ví dụ về JavaScrip:

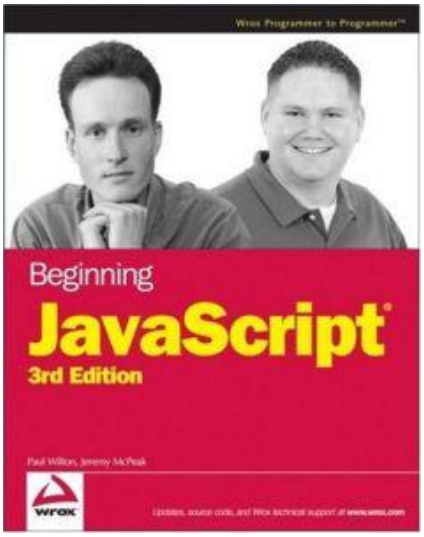
```
<html>
<body>
<script type="text/javascript">
document.write("This is my first JavaScript!");
</script>
</body>
</html>
```

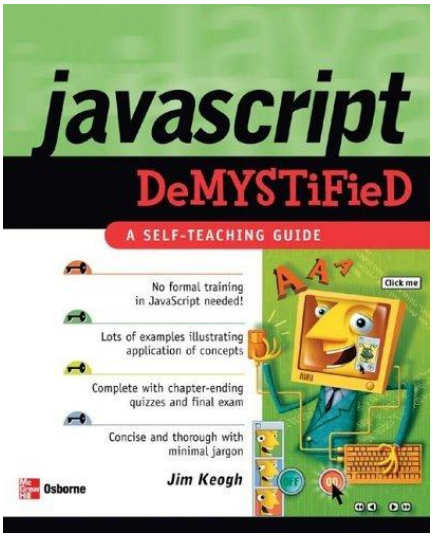
ở ví dụ này trên trang web sẽ hiện lên dòng chữ “ this is my first JavaScript !”.

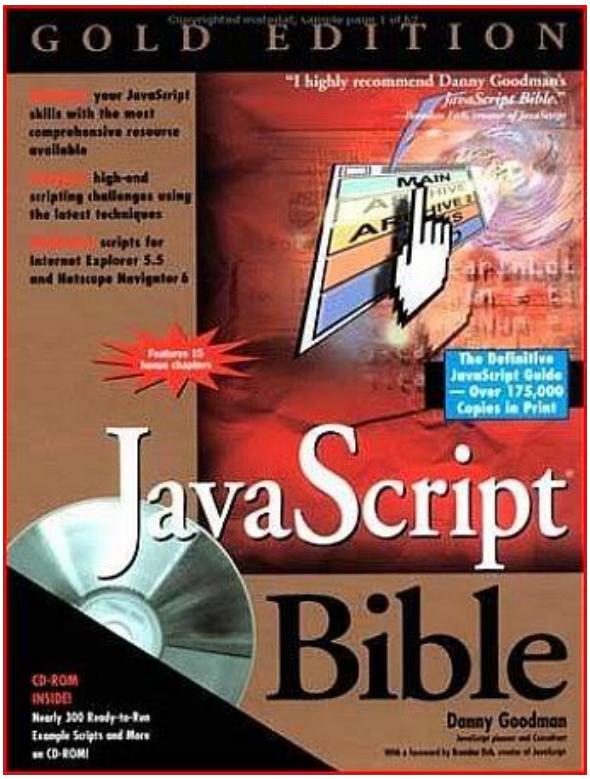
web tham khảo : http://www.w3schools.com

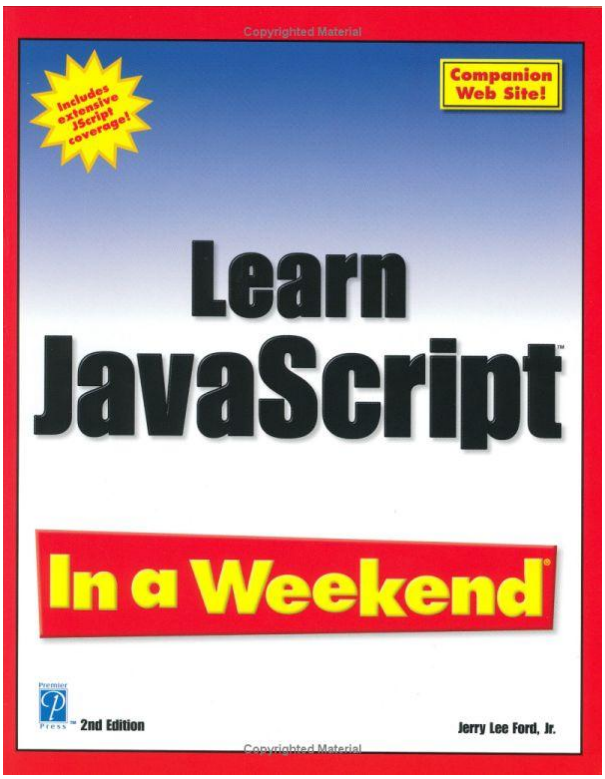
các sách tham khảo :

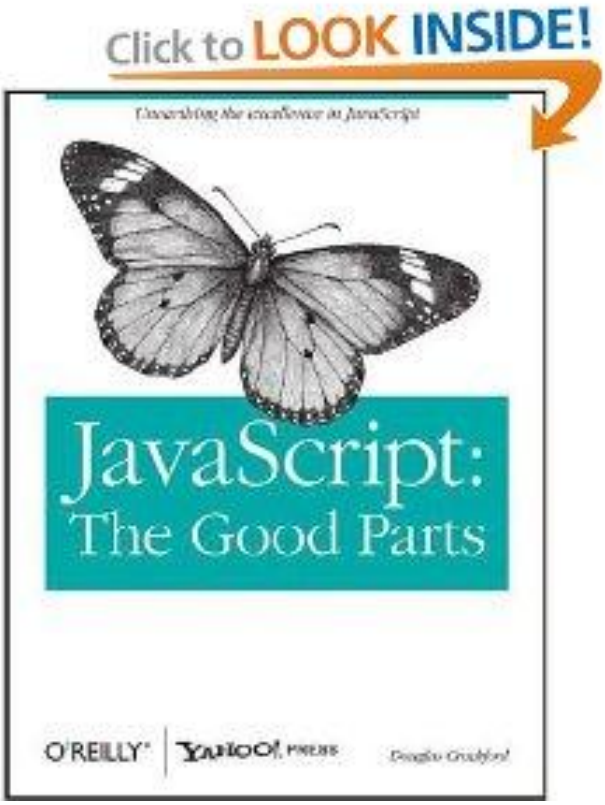

sitepoint
SIMPLY JAVASCRIPT
BY KEVIN YANK & CAMERON ADAMS
EVERYTHING YOU NEED TO LEARN JAVASCRIPT FROM SCRATCH
www.pnasoft.com


Wrox Programmer to Programmer™
Beginning
JavaScript
3rd Edition
Paul Wilton, Jeremy McPeak
WROX


javascript
DeMYSTiFieD
A SELF-TEACHING GUIDE
No formal training in JavaScript needed!
Lots of examples illustrating application of concepts
Complete with chapter-ending quizzes and final exam
Concise and thorough with minimal jargon
Click me
Osborne
Jim Keogh


GOLD EDITION
"I highly recommend Danny Goodman's JavaScript Bible."
your JavaScript skills with the most comprehensive resource available
high-end scripting challenges using the latest techniques
scripts for Internet Explorer 5.5 and Netscape Navigator 6
The Definitive JavaScript Guide — Over 175,000 Copies in Print!
JavaScript Bible
CD-ROM INSIDE!
Nearly 300 Ready-to-Run Example Scripts and More on CD-ROM!
Danny Goodman


Copyrighted Material
Includes extensive JScript coverage!
Companion Web Site!
Learn JavaScript
In a Weekend
2nd Edition
Jerry Lee Ford, Jr.
Copyrighted Material



Đặt code JavaScript ở đâu ?

Cách 1: **Đặt trong một trang HTML**

Ví dụ:

```
<html>
<body>
<script type="text/javascript">
document.write("Hello World!");
</script>
</body>
</html>
```

Thậm chí ở code dưới đây nó còn hiện cả tag HTML

```
<html>
<body>
<script type="text/javascript">
document.write("<h1>Hello World!</h1>");
</script>
```

```
</body>
</html>
```

Để chèn đoạn code JavaScrip vào trang HTML chúng ta sử dụng tag <script>…..</script>

Bên cạnh đó chúng ta phải định nghĩa ngôn ngữ được dùng trong tag này là gì ?

JavaScript ? hay Vbscript ?

Bởi vậy đoạn code sau là bắt buộc :

<script type="text/javascript"> … </script>

Ví dụ:

```
<html>
<body>
<script type="text/javascript">
...
</script>
</body>
</html>
```

Cách 2: **đặt trong thẻ Head của trang HTML** (thực ra cách này cũng là cách 1)

Ví dụ như sau:

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
……
}
</script>
</head>
<body>
<input type="button" onclick="show_confirm()" value="Show a confirm box" />
</body>
</html>
```

Cách 3: **chúng ta có thể đặt ở phần <head> và <body> của trang HTML cũng chả sao**

```
<html>
<head>
<script type="text/javascript">
function message()
{
```

```
alert("This alert box was called with the onload event");
}
</script>
</head>

<body onload="message()">
<script type="text/javascript">
document.write("This message is written by JavaScript");
</script>
</body>

</html>
```

Cách 4: **viết code JavaScrip ở một file riêng,rồi kết nối với file HTML**

Chú ý rằng File này có phần mở rộng là **.js** ,cách kết nối như sau:

```
<html>
<head>
<script type="text/javascript" src="xxx.js"></script>
</head>
<body>
</body>
</html>
```

Và ở bên ngoài chúng ta đã có một file xxx.js rồi.

Câu lệnh JavaScript ?

JavaScript là một chuỗi các lệnh được thực thi bởi trình duyệt.

Mỗi lệnh JS sẽ báo cho trình duyệt biết phải làm gì ?

Ví dụ câu lệnh sau sẽ báo cho trình duyệt biết phải in ra màn hình dóng chữ `Hello Dolly`

```
document.write ("Hello Dolly");
```

cuối mỗi câu lệnh phải có dấu ; (chấm phẩy) để trình duyệt phân biệt với các câu lệnh khác cùng dòng.

Các lệnh JS sẽ được thực hiện theo trình tự như chúng đã viết,ví dụ sau sẽ cho chúng ta thấy điều đó

```
<html>
<body>
<script type="text/javascript">
 document.write("<h1>This is a heading</h1>");
 document.write("<p>This is a paragraph.</p>");
```

```
 document.write("<p>This is another paragraph.</p>");
</script>
 </body>
</html>
```

Kết quả hiện thị sẽ là

This is a **heading**

This is a paragraph.

This is another paragraph

Sử dụng Comment trong JS ?

Thực ra nếu như các bạn đã học lập trình C thì thấy rằng JS có khá nhiều nét giống với C,chẳng hạn như chúng ta có thể viết các câu bình luận theo cách sau:

```
<script type="text/javascript">
/*
The code below will write
one heading and two paragraphs
*/
document.write("<h1>This is a heading</h1>");
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

Sử dụng dấu /* … */

Hoặc nếu như comment chỉ có 1 dòng thì chúng ta có thể dùng dấu //

```
<script type="text/javascript">
// Write a heading
document.write("<h1>This is a heading</h1>");
// Write two paragraphs:
document.write("<p>This is a paragraph.</p>");
document.write("<p>This is another paragraph.</p>");
</script>
```

công dụng của comment là gì thì các bạn đã rõ,nó chỉ như là lời giải thích cho đoạn code của các bạn mà thôi,chứ nó không phai là câu lệnh để thực thi,chính vì vậy bạn thích viết gì cũng được,chả ảnh hưởng gì hết

Biến trong JavaScript.

Biến là đối tượng mang thông tin.

Bạn có nhớ môn đại số bạn đã học không ?

X=5 ,y=6,z=x+y

Bạn có nhớ rằng một chữ cái (ví dụ như là chữ x) có thể được dùng để chưa một giá trị (ví dụ là 5)?

Và bạn có thể sử dụng thông tin trên để tính giá trị của z là 11 ?

Những chữ cái như vậy gọi là biến,và biến có thể được dùng để mang giá trị hoặc biểu thức (x=y+z).

Biến trong JS ?

Cũng tương như môn đại số mà bạn đã học mà thôi,biến trong JS dùng để chứa giá trị hoặc một biểu thức nào đó.

Tên của biến có thể ngắn thôi (ví dụ x),nhưng nếu có dài thì cũng chả sao (ví dụ carname)

Những điều sau đây các bạn phải lưu ý :

_ tên biến có phân biệt chữ hoa chữ thường (y và Y là khác nhau đó nhé)

_ tên phải bắt đầu bằng một chữ cái hoặc kí tự gạch chân.

Kiểu của biến ?

cũng giống như C thôi,chúng ta có các kiểu sau:

kiểu ***Boolean*** có hai giá trị là *true* và *false*

kiểu ***Số*** là loại giá trị dùng đễ tính toán

kiểu ***chuỗi*** chuỗi là các kí tự bảng chữ cái + kí tự đặc biệt + số, nói chung là bất kì cái gì mà ta viết ra trên bàng phím được, thì là chuỗi!

Phân biệt kiểu chuỗi và số ta gán biến a và b với các giá trị như sau: a=1; b="1"; a sẽ mang gái trị số, còn b mang giá trị chuỗi ==> đặt bên trong dấu nháy là chuỗi!

Có một điểm khác của JS so với C là như thế này: trong C khi khai báo biến chúng ta phải định kiểu trước cho nó,nhưng trong JS thì không phải như vậy:

Nếu như ta khia báo : var x ; // thì x chưa có kiểu gì cả.

Phải đến khi ta gán giá trị cụ thể cho biến thì nó mới có kiểu: ví dụ như là :

Var x = “ anh yêu em “ // lúc này kiểu của x là string (chuỗi)

Nhưng một lúc sau hứng lên chúng ta lại gán x = 5 // lúc đó x không còn là kiểu string nữa mà bây giờ kiểu của nó là number.

Biến toàn cục và biến cục bộ ?

Để khai báo biến cục bộ ta dùng cú pháp sau : var name;

Để khai báo biến toàn cục ta chỉ cần bỏ chữ var đi là được.

Chúng ta có thể gán ngay giá trị cho biến khi khai báo:

```
var x=5;
var carname="Volvo";
```

tất nhiên chúng ta cũng có thể khai báo

```
x=5;
carname="Volvo";
```

chú ý : khác với ngôn ngữ C thì JavaScript không có kiểu hằng số CONST để biểu diễn một giá trị không đổi nào đấy.

Tốt nhất trong javascript nếu như bạn chưa gán một giá trị cụ thể nào đó cho biến thì nên khởi tạo một giá trị nào đó cho nó,kinh nghiệm một số người họ hay gán giá trị cho biến là " " ví dụ :

Var x =" "; nếu như nó chứa giá trị là một chuỗi, và gán giá trị bằng 0 ví dụ : var y = 0; nếu như nó chứa giá trị là một số.

Các phép toán trong JavaScript

Cũng giống như trong ngôn ngữ C,với JS chúng ta có các phép toán như sau

Cho rằng y = 5, bảng dưới đây giải thích các nhà khai thác số học:

phép toán	Mô tả	Ví dụ	Kết quả
+	phép cộng	x = y 2	x = 7
-	Phép trừ	x = y-2	x = 3
*	Phép nhân	x = y * 2	x = 10
/	phép chia	x = y / 2	x = 2,5
%	lấy dư	x = y 2%	x = 1
+ +	tăng 1 đơn vị	x = + + y	x = 6
-	giảm 1 dơn vị	x =-- y	x = 4

Chúng ta có thêm bảng sau đây:

Cho x = 10 và y = 5, ta có tiếp bảng sau đây.

phép toán	Ví dụ	Tương tự như	Kết quả
=	x = y		x = 5

+ =	x + = y	x = x + y	x = 15
-=	x-= y	x = x-y	x = 5
*=	x *= y	x = x * y	x = 50
/ =	x / = y	x = x / y	x = 2
% =	x% = y	x = x% y	x = 0

Phép toán + được sử dụng trên strings

Phép toán cộng còn được dùng để thêm vào chuỗi các biến hoặc các giá trị văn bản với nhau.

Để ghép nhiều chuỗi lại với nhau thành một chuỗi mới chúng ta có thể sử dụng phép toán +.

```
txt1 = "What a very";
 txt2 = "nice day";
 txt3 = txt1 + txt2;
```

sau khi thực hiện các đoạn lệnh trên,biến txt3 sẽ chứa chuỗi: “What a verynice day”.

Để thêm khoảng trống cách giữa hai chuỗi trên ,chúng ta có thể làm như sau:

```
txt1 = "What a very";
 txt2 = "nice day";
 txt3 = txt1 + "" + txt2;
```

bay giờ thì biến txt3 chứa chuỗi như sau :”What a very nice day”.

Thêm số vào chuỗi kết quả sẽ thế nào ?

Trả lời: khi bạn thêm số vào chuỗi thì kết quả sẽ là một chuỗi.

Ví dụ :

```
<html>
<body>

<script type="text/javascript">
 x=5+5;
 document.write(x);
 document.write("<br />");
x="5"+"5";

 document.write(x);
 document.write("<br />");
 x=5+"5";
 document.write(x);
 document.write("<br />");
 x="5"+5;
```

```
 document.write(x);
 document.write("<br />");
</script>
</body>
</html>
```

Kết quả của đoạn code này sẽ là

10

55

55

55.

Các phép toán so sánh và Logic trong JavaScript

Các phép toán so sánh sẽ trả về kết quả là true hoặc false.

Ví dụ cho x=5,ta có bảng dưới đây:

Phép so sánh	Giải thích	Ví dụ
==	Bằng	x==8 is false
===	Bằng và cùng kiểu	x===5 is true x=="5" is false
!=	Không bằng	x!=8 is true
>	Lớn hơn	x>8 is false
<	Bé hơn	x<8 is true
>=	Lớn hơn hoặc bằng	x>=8 is false
<=	Bé hơn hoặc bằng	x<=8 is true

Các phép so sánh này sẽ được dùng như thế nào ?

Nó sẽ được dùng trong các câu lệnh điều kiện,các vòng lặp mà chúng ta sẽ được học trong những chương sau,sau đây là một ví dụ

```
if (age<18) document.write("Too young");
```

các phép toán Logic ?

ví dụ cho x = 6 và y=3,ta có bảng sau đây:

Toán tử	Giải thích	Ví Dụ
&&	and	(x < 10 && y > 1) is true
||	or	(x==5 || y==5) is false
!	not	!(x==y) is true

JS cũng chứa một phép toán mà nó sẽ chỉ định kết quả cho một biến dựa trên một số điều kiện cho trước,cú pháp của nó như sau:

```
variablename=(condition)?value1:value2
```

khi biểu thức condition đúng thì giá trị của variablename sẽ bằng value1,còn không thì nó sẽ bằng value2.

Cấu trúc IF… ELSE

Cấu trúc có điều kiện được sử dụng để thực hiện các hành động khác nhau dựa trên các điều kiện khác nhau.

Một ví dụ vui là như thế này:

“ nếu em yêu anh thì hai ta cùng cưới,nếu không thì thôi “

```
If (em yêu anh)
 {
    Thì hai ta cùng cưới
 }
Else
{
   Thì thôi
}
```

Lệnh có điều kiện

Thường khi bạn viết code,bạn muốn hành động khác nhau về các quyết định khác nhau.Bạn có thể sử dụng điều kiện kê trong mã của bạn để làm điều này.

Trong JS có các câu lệnh điều kiện sau:

- Câu lệnh **if** : sử dụng câu lệnh này để thực hiện một thao tác nào đó khi điều kiện là đúng
- Câu lệnh **if…else** : sử dụng câu lệnh này để thực hiện một thao tác nào đó khi điều kiện là đúng,và thao tác khác khi điều kiện là sai.

- Câu lệnh **if…else if…else**:sử dụng câu lệnh này để chọn một trong nhiều khối mã được thực thi.
- Câu lệnh **Switch** : sử dụng câu lệnh này để chọn một trong những khối lệnh để thực thi.

Câu lệnh if

Cú pháp:

```
if (điều kiện)
 (
 code được thực thi nếu điều kiện là đúng
 )
```

chú ý rằng chữ if … else phải viết bằng chữ thường,chữ hoa sẽ bị sai.

ví dụ:

```
<html>
<body>

<script type="text/javascript">
 var t=prompt(" nhap gia tri vao","nhap o day");
 if(t>200)
    {
      alert(" ban da trung thuong");
    }

</script>
</body>
</html>
```

Câu lệnh if…else

Cú pháp:

```
if (điều kiện)
 (
 code được thực thi nếu điều kiện là đúng
 )
 khác
 (
 code được thực thi nếu điều kiện là không đúng
 )
```

ví dụ:

```
<html>
<body>

<script type="text/javascript">
 var t=prompt(" nhap gia tri vao","nhap o day");
 if(t>5)

    {
      alert(" gia tri vua nhap lon hon 5");
    }
 else
    {
      alert(" gia tri vua nhap khong lon hon 5");
    }
</script>
</body>
</html>
```

Câu lệnh if…else if…else

Cú pháp:

```
if (condition1)
 (
 code được thực thi nếu condition1 là đúng
 )
 else if (condition2)
 (
 code được thực thi nếu condition2 là đúng
 )
 else
 (
 code được thực thi nếu condition1 và condition2 là không đúng
 )
```

ví dụ:

```
<html>
<body>
 <script type="text/javascript">
var d = new Date();
var time = d.getHours();
if (time<10)
{
document.write("<b>Good morning</b>");
}
```

```
else if (time>=10 && time<16)
{
 document.write("<b>Good day</b>");
 }
 else
 {
 document.write("<b>Hello World!</b>");
 }
 </script>
 <p>
 This example demonstrates the if..else if...else statement.
 </p>
 </body>
 </html>
```

Cấu trúc lựa chọn Switch

Cấu trúc này được sử dụng để thực hiện những hành động khác nhau dựa trên những điều kiện khác nhau.

Sử dụng cấu trúc switch để lựa chọn 1 trong số những mã lệnh để thực thi

Cú pháp:

```
switch(n)
{
case 1:
  execute code block 1
  break;
case 2:
  execute code block 2
  break;
default:
  code to be executed if n is different from case 1 and 2
}
```

cấu trúc trên hoạt động như thế nào ? đầu tiên chúng ta có một biểu thức duy nhất n (thường là một biến).Sau đó giá trị của biến được so sánh với giá trị cho mỗi một trường hợp (các case) trong cấu trúc.nếu có một có một trường hợp nào đó của case bằng với biến n thì khối lệnh tương ứng với case đó sẽ được thực thi.câu lệnh **break** ngăn không cho chạy tiếp các khối lệnh của các case còn lại.

Khi n không tương ứng với bất kì một **case** nào thì khối lệnh của **default** sẽ được thực thi.

Ví dụ : thử cho biết kết quả khi chạy đoạn mã sau

```
 <html>
<body>
<script type="text/javascript">
var d = new Date();
theDay=d.getDay();
switch (theDay)
{
case 5:
  document.write("<b>Finally Friday</b>");
  break;
case 6:
  document.write("<b>Super Saturday</b>");
  break;
case 0:
  document.write("<b>Sleepy Sunday</b>");
  break;
default:
  document.write("<b>I'm really looking forward to this weekend!</b>");
}
</script>
 <p>This JavaScript will generate a different greeting based on what day it is.
 Note that Sunday=0, Monday=1, Tuesday=2, etc.</p>
 </body>
</html>
```

Các hàm vào ra thông dụng nhất

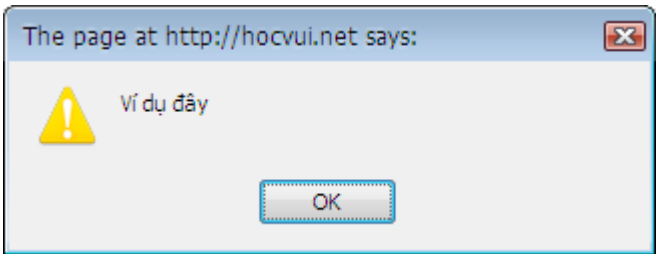
Trước khi đi vào những vấn đề cụ thể hơn chúng ta nghiên cứu những phương thức vào ra cơ bản cái đã.

Hàm **alert();**

alert(): Mở một hộp thông báo với nội dung đặt bên trong dấu ngoặc kép

VD: alert("Ví dụ đây");

Kết quả:

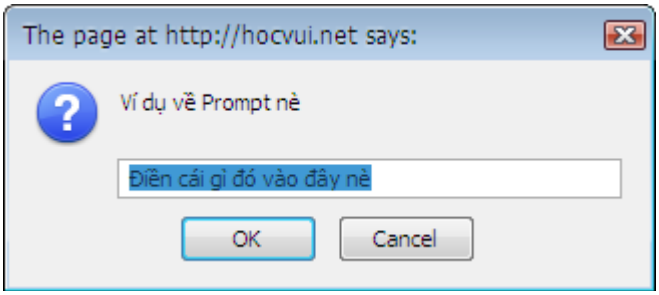


Hàm **prompt();**

prompt(): Mở một hộp thoại cho phép người sử dụng thông tin vào

VD: var t = prompt("Ví dụ về prompt nè","Điền cái gì đó vào đây nè");

Kết quả:



Hàm **document.write();**

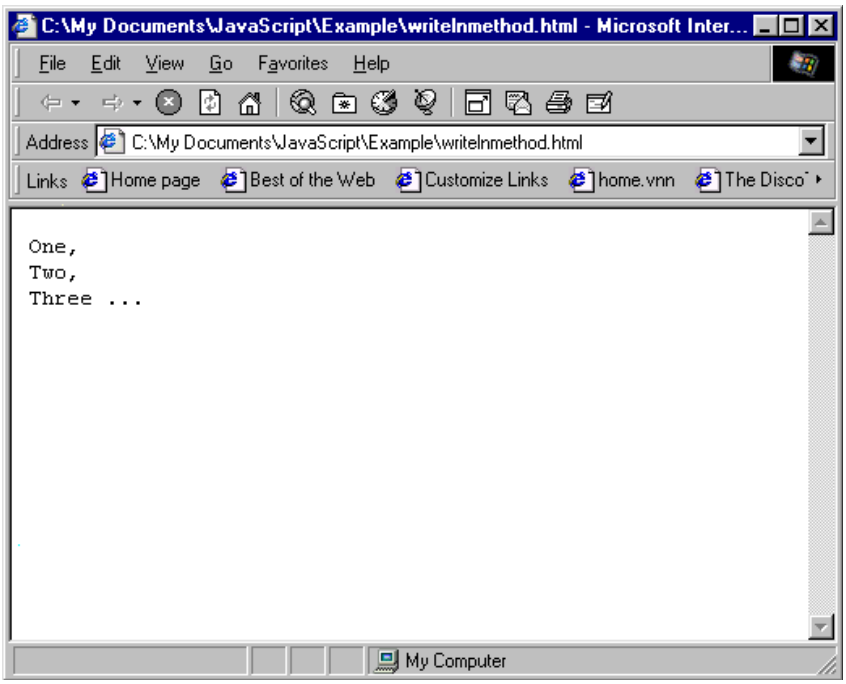
document.write(): in ra màn hình với nội dung đặt bên trong dấu ngoặc kép.

Hàm **document.writeln();**

Hàm này thì cũng có chức năng giống hàm kia,nhưng chỉ có điều là nó sẽ tự xuống dòng .Ta có ví dụ sau:

```
<SCRIPT LANGUAGE="JavaScript">
<!-- HIDE FROM OTHER BROWSERS
document.writeln("One,");
document.writeln("Two,");
document.write("Three ");
document.write("...");
// STOP HIDING FROM OTHER BROWSERS -->
</SCRIPT>
```

Kết quả hiện thị là như sau:



Hàm **confirm();**

Hàm này sẽ hiện ra một hộp thoại với một lời nhắn và hai nút OK và Cancel

Hàm này trả về giá trị là true hoặc false.

Nó sẽ trả về true nếu nhấn nút OK,và trả về false nếu nhấn nút Cancel.

Ví dụ :

```
<html>
<head>
<script type="text/javascript">
function show_confirm()
{
var r=confirm("Press a button!");
if (r==true)
  {
  alert("You pressed OK!");
  }
else
  {
  alert("You pressed Cancel!");
  }
}
</script>
</head>
<body>
 <input type="button" onclick="show_confirm()" value="Show a confirm box" />
 </body>
```

```
</html>
```

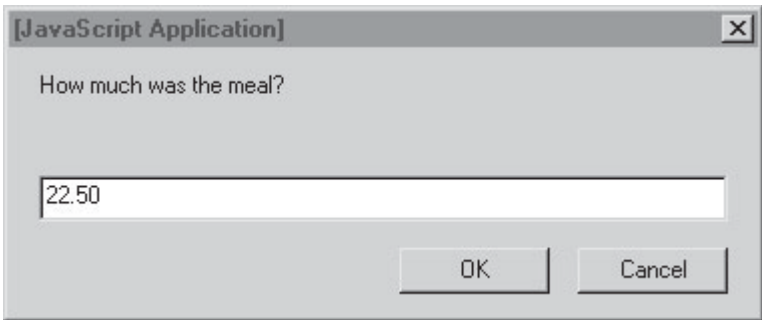
Hãy chạy đoạn code trên để trải nghiệm hàm **confirm();**

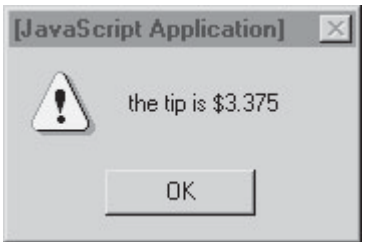
Sử dụng hàm eval() trong JavaScript

Xin quay lại với ví dụ sử dụng prompt.

```
<html>
<head>
<title>BadAdd</title>
</head>
<body>
<h1>The BadAdd</h1>
<script>
//BadAdd
//Demonstrates a potential pitfall
//Andy Harris, 09/00
var meal;
var tip = meal * .15;
var total = meal + tip;
//get the cost of the meal from the user
meal = prompt("How much was the meal?");
var tip = meal * .15;
var total = meal + tip;
alert ("the meal is $" + meal);
alert ("the tip is $" + tip);
alert ("Total bill: $" + total);
</script>
</body>
</html>
```

Với đoạn code này thì các hộp thoại hiện ra sẽ như sau :





Nhưng các bạn hãy để ý mà xem,rõ ràng kết quả không chính xác.Tại sao vậy ? câu trả lời đó là bởi vì khi bạn sử dụng prompt thì giá trị trả về là kiểu string chứ không phải là kiểu number,các bước sau các bạn lại thực hiện các phép toán trên kiểu string ! cho nên sai là không có gì phải bàn cãi.

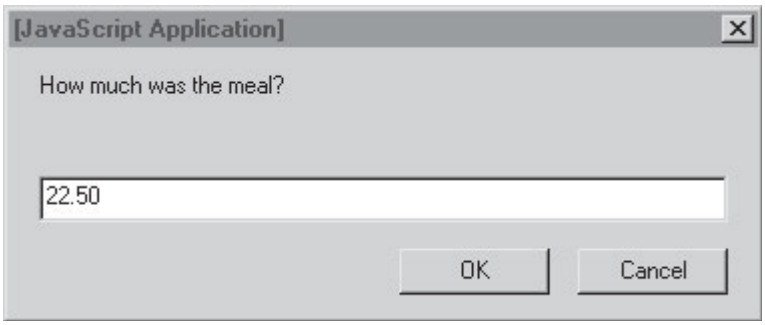
Để giải quyết vấn đề này JavaScript cung cấp cho chúng ta hàm eval(),giá trị trả về của eval luôn là một kiểu number.

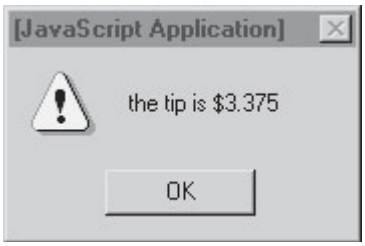
Chúng ta sẽ đính chính lại đoạn code như sau :

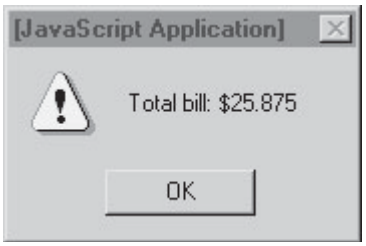
```
<html>
<head>
<title>GoodAdd</title>
</head>
<body>
<h1>The GoodAdd</h1>
<script>
//GoodAdd
```

```
//Demonstrates eval function
//Andy Harris, 09/00
var meal;
//get the cost of the meal from the user
meal = prompt("How much was the meal?");
//convert the value to a number
meal = eval(meal);
var tip = meal * .15;
var total = meal + tip;
alert ("the meal is $" + meal);
alert ("the tip is $" + tip);
alert ("Total bill: $" + total);
</script>
</body>
</html>
```

Rõ ràng kết quả bây giờ đã chính xác hơn :







Hàm(function) trong JavaScript

Một hàm sẽ được gọi từ một sự kiện (event) hoặc bằng một lời gọi đến nó.

Hàm JavaScript

Để giữ một <script> khi trình duyệt tải trang bạn có đặt nó vào một hàm.

Hàm sẽ được kích hoạt bởi từ một sự kiện(event) hoặc từ một lời gọi đến hàm.

Bạn có thể gọi hàm ở bất cứ đâu trong trang (hoặc từ một trang khác nếu hàm đó được nhúng trong một file có phần mở rộng **.js** đặt ở bên ngoài).

Hàm có thể được định nghĩa trong thẻ <head> hoặc trong thẻ <body> của trang.Tuy nhiên đặt trong thẻ <head> có vẻ khả thi hơn là bởi vì khi đó trình duyệt sẽ đọc và load hàm trước khi hàm được gọi.

Làm thế nào để xây dựng một hàm ?

Cú pháp:

```
function functionname(var1,var2,...,varX)
{
some code
}
```

var1,var2,…varX là các đối số của hàm.Các kí tự { và } đánh dấu bắt đầu và kết thúc một hàm.

Chú ý rằng một hàm mà không có đối số thì vẫn phải có dấu () phí sau tên hàm.

Chú ý : hàm phải được **viết bằng chữ thường**,khi gọi hàm thì phải gọi đúng tên của nó

Ví dụ về hàm:

Chạy thử đoạn code sau:

```
<html>
<head>
<script type="text/javascript">
function displaymessage()
{
alert("Hello World!");
}
</script>
</head>
<body>
```

```
<form>
<input type="button" value="Click me!" onclick="displaymessage()" />
</form>
 <p>By pressing the button above, a function will be called. The function will
 alert a message.</p>

</body>
</html>
```

Trong đoạn code trên nếu dòng alert("Hello World!"); không được đặt trong hàm thì dòng chữ sẽ hiện lên ngay lập tức ngay sau khi trình duyệt load đến.Thế nhưng khi nó đặt vào trong hàm thì phải chờ đến khi người dùng nhấn vào button ,hàm được kích hoạt thì dòng chữ mới bắt đầu hiện lên.

Lệnh return của hàm ?

Lệnh **return** của hàm dùng để xác định giá trị trả về từ hàm.

Vì vậy hàm mà bạn muốn trả về một giá trị nào đó thì phải có lệnh return.

Ví dụ dưới đây trả về kết quả của hai số (a,b):

```
 <html>
<head>
<script type="text/javascript">
function product(a,b)
{
return a*b;
}
</script>
</head>
<body>
<script type="text/javascript">
document.write(product(4,3));
</script>
 <p>The script in the body section calls a function with two parameters (4 and
 3).</p>
<p>The function will return the product of these two parameters.</p>
</body>
</html>
```

Nhắc lại khái niệm về biến toàn cục và biến cục bộ,một biến nằm trong hàm thì nó chỉ được sử dụng trong hàm đó mà thôi,còn một biến được khai báo ngoài tất cả các hàm (biến toàn cục) thì dùng ở đâu cũng được.

câu hỏi đề nghị:

bạn còn nhớ môn lập trình C chứ ? trong C hàm truyền theo tham trị . Vậy trong JS thì truyền theo gì ? **tham trị** hay **tham biến** ?

để kiểm chứng điều này chúng ta thử bằng đoạn code sau:

```
<html>
<head>
<script language="javascript">
b=10;
d=10;
function ham(a)
{
  a= 9;
}
function ham2(a,b)
{
  if(a==b)
    alert(" ham truyen theo tham tri");
  else
    alert(" ham truyen theo tham bien ");
}
</script>
</head>
<body>
<script language="javascript">
ham(b);
ham2(b,d);
</script>
</body>
</html>
```

Nhìn vào đoạn code trên chúng ta có thể thấy rằng ham() có chức năng gán giá trị 9 cho đối số.Như vậy thì sau khi ham(b) thực hiện xong giá trị của b phải bằng 9 ? và như vậy thì sau đó giá trị của b và d sẽ không bằng nhau ? Nhưng sự thật thì không phải như vậy.b và d vẫn bằng nhau và bằng 10.JavaScript cũng giống như C **truyền theo tham trị.**

Câu hỏi đề nghị:

Vậy làm cách nào để xây dựng được hàm thay đổi giá trị như trên ? cái khó của chúng ta là vấp phải nguyên tắc truyền theo tham trị của JS.Liệu có làm được không ?!

Thứ nữa các bạn hãy xây dựng một hàm có chức năng hoán đổi hai giá trị của hai biến cho nhau ?

Vòng lặp trong JavaScript

Vòng lặp thực thi một khối lệnh theo một số lần nhất định nào đó,hoặc thực thi khi điều kiện nào đó vẫn còn đúng.

Vòng lặp JavaScript ?

Thông thường khi bạn viết code,đôi khi bạn muốn một khối lệnh nào đó chạy đi chạy lại nhiều lần.Thay vì viết thành nhiều lần (rất tốn thời gian và công sức) bạn có thể sử dụng vòng lặp để thực hiện một công việc như thế.

Trong JS có 2 loại vòng lặp khác nhau:

- **For** – một khối lệnh trong vòng lặp sẽ được thực thi theo số lần nhất định nào đó.
- **While**- một khối lệnh trong vòng lặp sẽ được thi khi một điều kiện nào đó vẫn còn đúng.

Vòng lặp for ?

Vòng lặp này được dùng khi bạn biết trước được số lần có thể chạy của vòng lặp.

Cú pháp:

```
for (var=startvalue;var<=endvalue;var=var+increment)
{
code to be executed
}
```

thực ra nếu như các bạn đã biết về C rồi thì vấn đề này không có gì là khó cả.

ví dụ:

ví dụ dưới đây định nghĩa một vòng lặp bắt đầu với i=0,vòng lặp sẽ tiếp tục chạy chừng nào mà i còn bé hơn hoặc bằng 5.Khi I lớn hơn 5 thì vòng lặp sẽ dừng lại.

```
<html>
<body>
<script type="text/javascript">
for (i = 0; i <= 5; i++)
{
document.write("The number is " + i);
document.write("<br />");
}
</script>
</body>
</html>
```

Và dưới đây là kết quả :

The number is 0

The number is 1

The number is 2

The number is 3

The number is 4

The number is 5

Vòng lặp for…in ?

Vòng lặp này dựa trên các yếu tố của mảng(array) hoặc các thuộc tính của đối tượng.

Cú pháp :

```
for (variable in object)
  {
  code to be executed
  }
```

chú ý rằng khối lệnh nằm trong vòng lặp này được thực thi một lần cho mỗi phần tử của đối tượng.

đối số của vòng lặp có thể là tên một biến,một phần tử của mảng,hoặc thuộc tính của đối tượng.

ví dụ:

sử dụng vòng lặp for…in để tương tác với mảng(array)

```
<html>
<body>
<script type="text/javascript">
var x;
var mycars = new Array();
mycars[0] = "Saab";
mycars[1] = "Volvo";
mycars[2] = "BMW";
for (x in mycars)
{
document.write(mycars[x] + "<br />");
}
</script>
</body>
</html>
```

Kết quả như sau:

Saab

Volvo

BMW

Vòng lặp while ?

Khi bạn không thể biết trước được chính xác số lần có thể lặp của vòng lặp thì nên sử dụng vòng lặp này.

Vòng lặp sẽ thực thi khối lệnh chừng nào điều kiện còn đúng.

Cú pháp:

```
while (var<=endvalue)
  {
  code to be executed
  }
```

ví dụ:

ví dụ này xác định một vòng lặp bắt đầu với i=0,vòng lặp sẽ tiếp tục thực thi khối lệnh nằm trong nó chừng nào điều kiện (i<=5) vẫn còn đúng

```
<html>
<body>
<script type="text/javascript">
i=0;
while (i<=5)
{
document.write("The number is " + i);
document.write("<br />");
i++;
}
</script>
```

Vòng lặp do…while ?

Thực ra thì vòng lặp này cũng giống như vòng lặp while mà thôi,chỉ có điều là nó sẽ thực khối lệnh trong vòng lặp tối thiểu là một lần.(trong khi ở vòng lặp while có khi khối lệnh sẽ không được thực hiện một lần nào).

Cú pháp:

```
do
  {
  code to be executed
  }
while (var<=endvalue);
```

ví dụ:

```
<html>
<body>
<script type="text/javascript">
i = 0;
do
{
document.write("The number is " + i);
document.write("<br />");
```

```
i++;
}
while (i <= 5)
</script>
```

Kết quả sẽ như sau:

The number is 0

The number is 1

The number is 2

The number is 3

The number is 4

The number is 5

Vai trò lệnh Break và Continue trong vòng lặp ?

Hai lệnh break và continue trong vòng lặp là vô cùng quan trọng bởi vậy bạn không thể bỏ qua nó được.

Lệnh break ?

Lệnh này sẽ thực hiện việc thoát vòng lặp và thực hiện tiếp những câu lệnh còn lại của tài liệu.

Ví dụ:

```
<html>
<body>
<script type="text/javascript">
var i=0;
for (i=0;i<=10;i++)
{
if (i==3)
  {
  break;
  }
document.write("The number is " + i);
document.write("<br />");
}
</script>
<p>Explanation: The loop will break when i=3.</p>
</body>
</html>
```

Kết quả thu được sẽ như thế này:

The number is 0

The number is 1

The number is 2

Explanation: The loop will break when i=3.

Lệnh continue ?

Lệnh này sẽ thực hiện việc thoát vòng lặp và bắt đầu một vòng lặp mới.

Ví dụ:

```
<html>
<body>
<script type="text/javascript">
 var i=0;
 for (i=0;i<=10;i++)
 {
 if (i==3)
  {
  continue;
  }
 document.write("The number is " + i);
 document.write("<br />");
 }
 </script>
 <p>Explanation: The loop will break the current loop and continue with the next
 value when i=3.</p>
 </body>
 </html>
```

Kết quả sẽ là :

The number is 0

The number is 1

The number is 2

The number is 4

The number is 5

The number is 6

The number is 7

The number is 8

The number is 9

The number is 10

Explanation: The loop will break the current loop and continue with the next value when i=3.

Các lệnh thao tác trên đối tượng

JS là một ngôn ngữ hướng đối tượng,vì vậy cho nên nó có một số câu lệnh thao tác với đối tượng.

new

Biến new được thực hiện để tạo ra một thể hiện mới của một đối tượng

Cú pháp

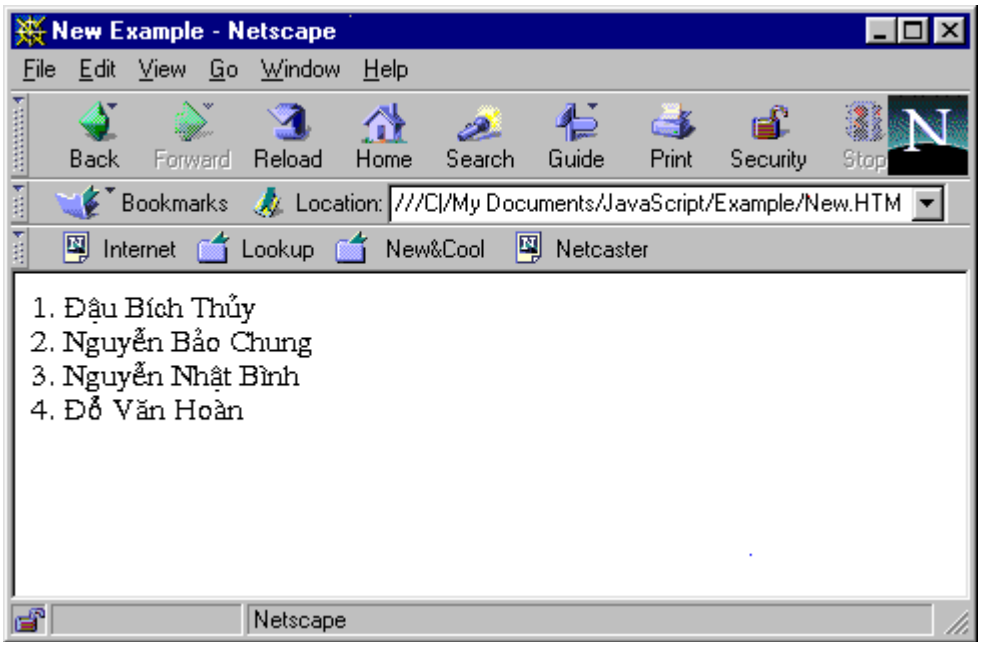
objectvar = new object_type (param1 [,param2]... [,paramN])

Ví dụ sau tạo đối tượng person có các thuộc tính firstname, lastname, age, sex.

Chú ý rằng từ khoá this được sử dụng để chỉ đối tượng trong hàm person. Sau đó ba thể hiện của đối tượng person được tạo ra bằng lệnh new

```
<HTML>
<HEAD>
<TITLE>New Example </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
function person(first_name, last_name, age, sex){
this.first_name=first_name;
this.last_name=last_name;
this.age=age;
this.sex=sex;
}
person1= new person("Thuy", "Dau Bich", "23", "Female");
person2= new person("Chung", "Nguyen Bao", "24", "Male");
person3= new person("Binh", "Nguyen Nhat", "24", "Male");
person4= new person("Hoàn", "Đỗ Văn", "24", "Male");
document.write ("1. "+person1.last_name+" " + person1.first_name + "<BR>" );
document.write("2. "+person2.last_name +" "+ person2.first_name + "<BR>");
document.write("3. "+ person3.last_name +" "+ person3.first_name + "<BR>");
document.write("4. "+ person4.last_name +" "+ person4.first_name+"<BR>");
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Kết quả hiện thị như sau:



Qua đây chúng ta có thể tìm ra được cú pháp tạo đối tượng như thế nào.

this

Từ khoá this được sử dụng để chỉ đối tượng hiện thời. Đối tượng được gọi thường là đối tượng hiện thời trong phương thức hoặc trong hàm.

Cú pháp

this [.property]

Có thể xem ví dụ của lệnh new.

with

Lệnh này được sử dụng để thiết lập đối tượng ngầm định cho một nhóm các lệnh, bạn có thể sử dụng các thuộc tính mà không đề cập đến đối tượng.

Cú pháp

with (object)

{

// statement

}

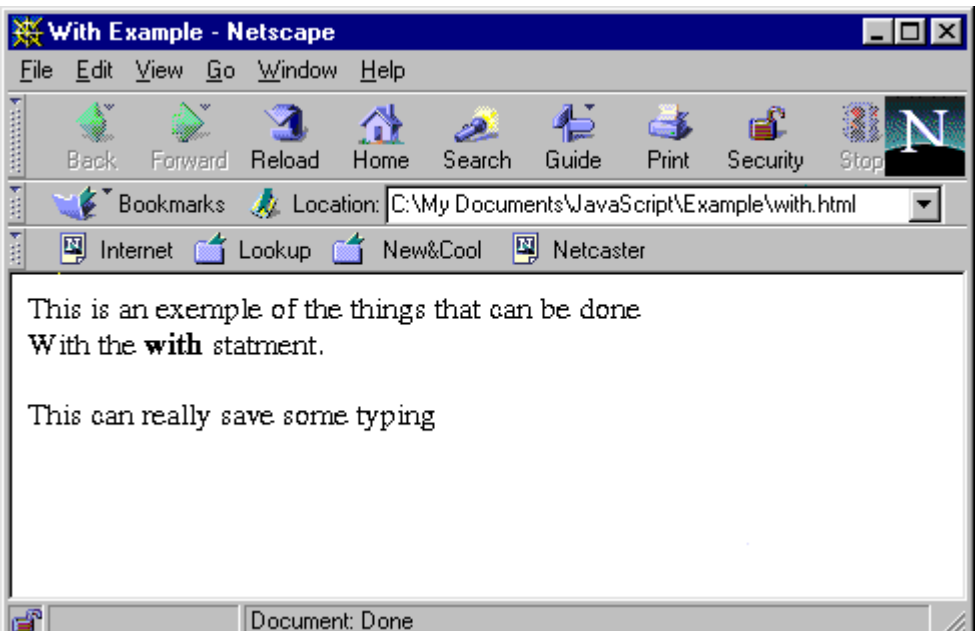
Ví dụ:

Ví dụ sau chỉ ra cách sử dụng lệnh with để thiết lập đối tượng ngầm định là document và có thể sử dụng phương thức write mà không cần đề cập đến đối tượng document

```
<HTML>
<HEAD>
<TITLE>With Example </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
```

```
with (document){
write("This is an exemple of the things that can be done <BR>");
write("With the <B>with<B> statment. <P>");
write("This can really save some typing");
}
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Kết quả hiện thị như sau:

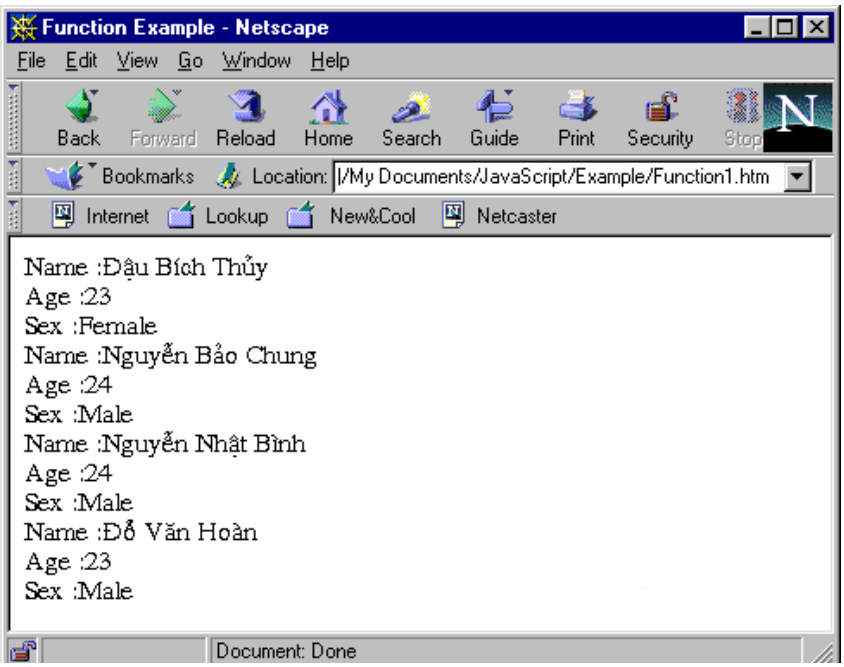


Hãy xem xét ví dụ sau :

```
<SCRIPT LANGUAGE= "JavaScript">
function person(first_name, last_name, age, sex) // tạo đối tượng person
{
this.first_name=first_name;
this.last_name=last_name;
this.age=age;
this.sex=sex;
this.printStats=printStats;
}
function printStats() // tạo phương thức printStats cho đối tượng person
{
with (document) {
```

```
write (" Name :" + this.last_name + " " + this.first_name + "<BR>" );
write("Age :"+this.age+"<BR>");
write("Sex :"+this.sex+"<BR>");
}
}
person1= new person("Thuy", "Dau Bich", "23", "Female");
person2= new person("Chung", "Nguyen Bao", "24", "Male");
person3= new person("Binh", "Nguyen Nhat", "24", "Male");
person4= new person("Hoan", "Do Van", "23", "Male");
person1.printStats();  // sử dụng phương thức trên đối tượng
person2.printStats();
person3.printStats();
person4.printStats();
</SCRIPT>
```

Kết quả hiện thị như sau :



Bình luận : bây giờ hì chúng ta đã biết cách tạo đối tượng và tạo thuộc tính cho đối tượng rồi nhé.

Tạo đối tượng trong JavaScript

1 . Định nghĩa thuộc tính đối tượng

```
function student(name,age,grade) {
```

```
this.name = name;
this.age = age;
this.grade = grade;
}
```

Để tạo một Object ta sử dụng một phát biểu new . Ví dụ để tạo đối tượng student1

```
student1 = new student("Bod",10,75);
```

3 thuộc tính của đối tượng student1 là :

```
student1.name
student1.age
student1.grade
```

Ví dụ để tạo đối tượng student2

```
student2 = new student("Jane",9,82);
```

Để thêm thuộc tính cho student1 bạn có thể làm như sau

```
student1.mother = "Susan";
```
hoặc bạn có thể định nghĩa lại hàm student

```
function student(name,age,grade,mother) {
this.name = name;
this.age = age;
this.grade = grade;
this.mother = mother;
}
```

Đối tượng là thuộc tính của đối tượng khác

Ví dụ :

```
function grade ( math , english , science ) {
this.math = math;
this.english = english;
this.science = science;
}
bobGrade = new grade(75,80,77);
```

```
janeGrade = new grade(82,88,75);
student1 = new student("Bob",10,"bodGrade");
student2 = new student("Jane",9,"janeGrade");
```

student1.grade.math : dùng để lấy điểm toán của student1

student2.grade.science : dùng lấy điểm môn khoa học của student2

2 . Thêm phương pháp cho đối tượng

```
function displayProfile() {
document.write("Name : " + this.name + "<BR");
document.write("Age : " + this.age + "<BR");
document.write("Mother's Name : " + this.mother + "<BR>");
document.write("Math Grade : " + this.grade.math + "<BR>");
document.write("English Grade : " + this.grade.english + "<BR>");
document.write("Science Grade : " + this.grade.science + "<BR>");
}
function student(name,age,grade) {
this.name = name;
this.age = age;
this.grade = grade;
this.mother = mother;
this.displayProfile = displayProfile;
}
student1.displayProfile();
```

Ví dụ :

```
<SCRIPT LANGUAGE ="JavaScript">
<!– Begin
var day="";
var month="";
var ampm="";
var ampmhour="";
```

```
var myweekday="";
var year="";
mydate = new Date();
myday = mydate.getDay();
mymonth = mydate.getMonth();
myweekday = mydate.getDate();
weekday = myweekday ;
myyear= mydate.getYear();
year = myyear;
myhours = mydate.getHours();
ampmhour = (myhours > 12 ) ? myhours-12 : myhours;
ampm = (myhours > 12 ) ? 'Buổi chiều' : 'Buổi sáng';
mytime = mydate.getMinutes();
myminutes = ((mytime < 10 ) ? ':0' : ':' ) + mytime;
if(myday == 0)
day = " Chủ nhật , ";
else if(myday == 1 )
day = "Thứ hai, ";
else if(myday == 2 )
day = "Thứ ba, ";
else if(myday == 3 )
day = "Thứ tư, ";
else if(myday == 4)
day = "Thứ năm, ";
else if(myday == 5 )
day = "Thứ sáu, ";
else if(myday == 6 )
day = "Thứ bảy, ";
if(mymonth == 0 ) {
month = "tháng một ";}
```

```
else if(mymonth ==1 )
month = "tháng hai ";
else if(mymonth ==2 )
month = "tháng ba ";
else if(mymonth ==3 )
month = "tháng tư ";
else if(mymonth ==4 )
month = "tháng năm ";
else if(mymonth ==5)
month = "tháng sáu ";
else if(mymonth ==6 )
month = "tháng bảy ";
else if(mymonth ==7 )
month = "tháng tám ";
else if(mymonth ==8 )
month = "tháng chín ";
else if(mymonth ==9 )
month = "tháng mười ";
else if(mymonth ==10 )
month = "tháng mười một ";
else if(mymonth ==11 )
month = "tháng mười hai ";
// End ->
</SCRIPT>
```

Trong phần body bạn có thể xuất ra dạng như sau :

```
<BODY>
<SCRIPT>
document.write("<b><font color=#0000ff face='Arial'>" + ampmhour + "" + myminutes + ampm )
document.write(" - " + day + " ngày " + myweekday + " ");
document.write( month + " ,năm " + year + "</font>");
```

</SCRIPT>

</BODY>

Lỗi chưa khởi tạo đối tượng trong javascript

```
1. // object constructor
2. function ObjectConstructor(a, b, c) {
3.  this.A = a;
4.  this.B = b;
5.  this.C = c;
6.  this.Total = a + b + c;
7. }
8. var obj = ObjectConstructor(1, 2, 3);
9. alert(obj.Total);
```

kết quả trả về không phải bằng 6 mà là chẳng có gì cả. Bởi đối tượng obj chưa được khởi tạo.
có 2 cách sửa lỗi này:
cách 1:
var obj = new ObjectConstructor(1, 2, 3);
cách 2:quả trả

```
1. // object constructor
2. function ObjectConstructor(a, b, c) {
3. if (!(this instanceof arguments.callee)) {
4.  return new ObjectConstructor(a, b, c);
5. }
6.  this.A = a;
7.  this.B = b;
8.  this.C = c;
9.  this.Total = a + b + c;
10.}
11.alert(obj.Total);
```

Sự kiện(event) trong JavaScript

Sự kiện là những hành động mà có thể nhận diện bằng JavaScript.

Event ?

Bằng cách sử dụng JS chúng ta có thể tạo ra các trang web động.Và event là các hành động mà ta có thể nhận diện bằng JavaScript.

Chẳng hạn chúng ta dùng sự kiện onClick của một button để cho biết rằng một chức năng sẽ chạy khi người dùng nhấp vào button đó.Chúng ta xác định các event trong các thẻ HTML.

Ví dụ về các sự kiện:

- Một click chuột
- Một trang web hoặc một hình ảnh đang tải
- Di chuyển trên một điểm đặc biệt của trang web
- Lựa chọn một trường đầu vào trong thẻ <form> của HTML
- Submit một form
- Một tổ hợp phím

Chú ý rằng : sự kiện thường đi kèm với hàm,hàm sẽ không được kích hoạt khi sự kiện (event) chưa xảy ra.

onLoad và onUnload ?

Hai sự kiện này sẽ được kích hoạt khi người dùng nhập vào hoặc rời khỏi trang.

Các sự kiện onload thường được sử dụng để kiểm tra loại trình duyệt của người truy cập và phiên bản trình duyệt, và tải phiên bản thích hợp của trang web dựa trên thông tin.

Cả hai sự kiện onload và onUnload cũng thường được sử dụng để tương tác với các tập tin cookie mà nên được thiết lập khi người dùng nhập vào một trang. Ví dụ, bạn có thể có một popup yêu cầu điền tên người dùng khi lần đầu tiên đến trang của bạn. Tên này sau đó được lưu giữ trong một cookie. Tiếp theo thời gian khách truy cập đến tại trang web của bạn, bạn có thể có một popup nói cái gì đó như: "Chào mừng John Doe!".

onFocus, và onChange onBlur ?

Các onFocus, onBlur và onChange thường được sử dụng kết hợp với các trường mẫu.

Bên dưới là một ví dụ về cách sử dụng các sự kiện onChange. Hàm checkEmail () sẽ được gọi bất cứ khi nào người dùng thay đổi nội dung của trường:

```
<input type="text" size="30" id="email" onchange="checkEmail()">
```

onSubmit ?

đây là event hay được đặt trong các <form>.Trong các form thường có nút Submit để thực hiện việc chấp nhận dữ liệu đầu vào để gửi lên sever.Thế nhưng đôi khi chúng ta phải kiểm tra các dữ liệu đó trước khi nó được gửi lên sever,sự kiện onSubmit sẽ cho phép ta làm điều đó.

Dưới đây là một ví dụ về cách sử dụng các sự kiện onSubmit. Hàm checkForm () sẽ được gọi khi người dùng nhấn nút gửi biểu mẫu (nhấn Submit). Hàm checkForm trả về hoặc là đúng hoặc sai. Nếu nó trả về đúng thì sẽ được gửi, nếu không sẽ bị hủy bỏ:

```
action="xxx.htm" method="post" <form onsubmit="return checkForm()">
```

onMouseOver và onMouseout ?

onMouseOver và onmouseout là hai event liên quan đến chuột.

Dưới đây là một ví dụ về một sự kiện onMouseOver. Một hộp cảnh báo xuất hiện khi một sự kiện onMouseOver được kích hoạt:

```
href="http://www.w3schools.com" <a onmouseover="alert('An onMouseOver event');return
false"> <img src="w3s.gif" alt="W3Schools" /> </ a >
```

Ta có tham khảo như sau:

Sự kiện	Ý nghĩa
onMouseUp	nútchuột được nhả
onMouseDown	ấnchuột
onMouseOver	chuộtdi chuyển qua vùng
onMouseOut	chuộtdi chuyển khỏi vùng
onMouseMove	chuộtđược di chuyển
onclick	clickchuột
ondbclick	click đúpchuột

Try – Catch

Đây là câu lệnh để kiểm lỗi của khối lệnh.

JavaScript bắt lỗi ?

Khi duyệt trang web trên internet chúng ta đôi khi bắt gặp một hộp thoại nói rằng có một lỗi runtime khi đang duyệt trang và có hỏi rằng "Do you wish to debug?".Những hộp thoại cảnh báo như thế này rất có ích cho nhà phát triển web,nhưng nó lại làm cho người sử dụng thoát khỏi trang web của bạn ngay lập tức,website của bạn có thể bị mất khách vì lí do này.

Bài học này sẽ giúp các bạn nắm bắt và xử lý các thông báo lỗi của JavaScript,nó sẽ có tác dụng to lớn trong việc tránh mất khách truy cập bởi những lỗi chẳng đâu vào đâu.

Try – Catch ?

Try – Catch giúp chúng ta kiểm tra lỗi của khối lệnh,trong nó chứa phần khối lệnh chạy và phần khối lệnh thực thi trong trường hợp có lỗi xảy ra.

Cú pháp:

```
try
  {
  //Run some code here
  }
catch(err)
  {
  //Handle errors here
  }
```

chú ý rằng : try – catch phải được viết bằng chữ thường,viết bằng chữ hoa sẽ gặp lỗi cú pháp.

Ví dụ sau đây sẽ thực hiện công việc hiện thị thông báo “chào mừng quý khách” khi button được click.Nhưng có điều là trong khối lệnh này có cú pháp alert(“ chào mừng quý khách”); bị đánh sai thành là adddlert(“ chào mừng quý khách”).Một lỗi JS đã xảy ra,và try…catch sẽ bắt lỗi nó và thực thi ma tùy chỉnh để xử lý.Mã này sẽ hiện thị một thông báo lỗi cho người sử dụng biết chuyện gì đang xảy ra.

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Error description: " + err.description + "\n\n";
  txt+="Click OK to continue.\n\n";
  alert(txt);
  }
}
</script>
</head>
<body>
<input type="button" value="View message" onclick="message()" />
```

```
</body>
</html>
```

Kết quả sau khi bạn nhấn vào button sẽ là thông báo như sau:

Trong ví dụ tiếp sau đây chúng ta sẽ sử dung hộp thoại confirm hiện thị một lời nhắn để cho người dùng tùy chọn như sau:

Họ có thể click vào nút OK để tiếp tục hiện thị trang web hoặc click Cancer để tới trang chủ (không đả động vào trang web lỗi ấy nữa).Nếu confirm trả về giá trị false ,người dùng nhấn Cancer,và mã chuyển hướng người dùng.Nếu confirm trả về giá trị true thì trang web hiện lên không có gì.

```
<html>
<head>
<script type="text/javascript">
var txt="";
function message()
{
try
  {
  adddlert("Welcome guest!");
  }
catch(err)
  {
  txt="There was an error on this page.\n\n";
  txt+="Click OK to continue viewing this page,\n";
  txt+="or Cancel to return to the home page.\n\n";
  if(!confirm(txt))
    {
    document.location.href="http://www.w3schools.com/";
    }
  }
}
</script>
```

```
</head>
<body>
<input type="button" value="View message" onclick="message()" />
</body>
</html>
```

Lệnh throw

Chúng ta có thể hình dung lệnh này có tác dụng như là " bắt lỗi".

Lệnh này cho phép bạn tạo một ngoại lệ.nếu bạn sử dụng lệnh này với lệnh try…catch bạn có thể kiểm soát được chương trình và đưa ra những thông báo lỗi chính xác.

Cú pháp:

```
throw(exception)
```

exception có thể là một chuỗi,một giá trị boolean,số nguyên hoặc một đối tượng.

chú ý rằng : throw phải được viết bằng chữ thường,viết chữ hoa là sai về cú pháp.

Ví dụ:

Ví dụ dưới đây xác định giá trị một biến x,nếu giá trị của x lớn hơn 10 ,bé hơn 0 hoặc không phải là số chúng ta sẽ throw một lỗi.Lỗi này sau đó bị bắt bởi các đối số bắt và đúng thông báo lỗi được hiện thị.

```
<html>
<body>
<script type="text/javascript">
var x=prompt("Enter a number between 0 and 10:","");
try
{
if(x>10)
  {
  throw "Err1";
  }
else if(x<0)
  {
  throw "Err2";
  }
else if(isNaN(x))
  {
  throw "Err3";
  }
}
catch(er)
{
```

```
if(er=="Err1")
  {
  alert("Error! The value is too high");
  }
if(er=="Err2")
  {
  alert("Error! The value is too low");
  }
if(er=="Err3")
  {
  alert("Error! The value is not a number");
  }
}
</script>
</body>
</html>
```

Trong phần try lệnh throw sẽ tiến hành việc bắt lỗi,khi lỗi xảy ra nó sẽ chuyển qua phần catch để xem tương ứng với lỗ này thì hiện thông báo gì.

Các ký tự đặc biệt trong JavaScript

Trong JS các bạn có thể thêm các ký tự đặc biệt vào văn bản bằng cách sử dụng dấu gạch chéo ngược \

Chèn ký tự đặc biệt ?

Dấu gạch chéo ngược \ được dùng để chèn dấu nháy, new lines,quote và những ký tự đặc biệt khác vào chuỗi.

Hãy nhìn vào đoạn code dưới đây

```
var txt="We are the so-called "Vikings" from the north.";
document.write(txt);
```

trong JS mỗi chuỗi đều bắt đầu và kết thúc bởi dấu nháy đơn hoặc dấu nháy kép,điều đó có nghĩa là chuỗi trên sẽ được chia nhỏ thành : We are the so-called

để giải quyết vấn đề này các bạn phải thêm một dấu \ vào phía trước mỗi dấu nháy ở từ "Vikings",ta sửa lại đoạn code như sau:

```
var txt="We are the so-called \"Vikings\" from the north.";
document.write(txt);
```

bây giờ thì kết quả hiện lên mà hình sẽ là:

We are the so-called "Vikings" from the north.

Dưới đây là một ví dụ khác:

```
document.write ("You \& I are singing!");
```

ví dụ trên sẽ cho ra kết quả như sau:

```
You & I are singing!
```

Bảng cho dưới đây bao gồm những kí tự đặc biệt khác,bạn có thể dùng để thêm vào chuỗi của mình.

Code	Outputs
\'	single quote
\"	double quote
\&	ampersand
\\	backslash
\n	new line
\r	carriage return
\t	tab
\b	backspace
\f	form feed

Một số chú ý khi sử dụng JavaScript

Hãy cẩn thận khi làm việc với JS ?

Khi bạn gọi tên hàm,goi tên biến trong JS thì hãy nhớ rằng chữ thường và chữ hoa sẽ khác nhau,nghĩa là một hàm có tên là " myfunction" sẽ không giống như "myFunction" ,và một biến có tên là "myvar" sẽ không giống như "myVar".

Khoảng trắng trong JS ?

JS bỏ qua khoảng trắng khi nó nhìn thấy,vì vậy bạn có thể thêm vào khoảng trắng để cho mã nguồn của các bạn trở nên dễ đọc hơn.Tất nhiên là cũng không nên lam dụng khoảng trắng quá nhiều sẽ làm cho dung lượng mã nguồn tăng lên,gây cản trở khi load trang.Các dòng sau là tương đương:

```
name="Hege";
name = "Hege";
```

break up một dòng mã ?

Bạn có thể dùng dấu gạch chéo ngược để làm điều đó,ví dụ dưới đây sẽ được hiện thị đúng:

```
document.write("Hello \
World!");
```

nhưng như dưới đây thì lại không đúng:

```
document.write \
("Hello World!");
```

Chương 2

JS Objects

Objects trong JavaScript

Đối tượng trong JS là một khái niệm quan trọng.

JS là ngôn ngữ hướng đối tượng.

Ngôn ngữ hướng đối tượng cho phép bạn định nghĩa đối tượng cho riêng bạn.

Lập trình hướng đối tượng ?

JS là ngôn ngữ lập trình hướng đối tượng,bạn có thể định nghĩa đối tượng cho riêng bạn.

Nhưng bây giờ chưa phải là lúc để nói cách tạo đối tượng như thế nào,nó sẽ được đề cập ở các bài tiếp theo.Chúng ta sẽ bắt đầu bằng cách nhìn xem đối tượng được xây dựng thế nào? Và cách sử dụng ra sao ?

Lưu ý rằng đối tượng chỉ là một dạng đặc biệt của dữ liệu.Một đối tượng bao gồm thuộc tính (properties) và phương thức (method).Đây là hai vấn đề rất quan trọng của đối tượng.

Thuộc tính

Thuộc tính (properties) là giá trị liên kết với một thuộc tính.

Trong ví dụ dưới đây chúng ta sử dụng thuộc tính về độ dài (length) của đối tượng string để trả lại số kí tự của một chuỗi.

```
<script type="text/javascript">
 var txt = "Hello World!";
 document.write (txt.length);
 </ Script>
```

Kết quả của đoạn code trên là

12

Phương thức

Phương thức (method) là những hành động có thể thực hiện trên đối tượng.

Trong ví dụ dưới đây chúng ta sử dụng phương thức toUpperCase ( ) của đối tượng String để hiện thị văn bản dưới dạng chữ hoa.

```
<script type="text/javascript">
 var str = "Xin chào thế giới!";
 document.write (str.toUpperCase ());
 </ Script>
```

Kết quả là :

```
HELLO WORLD!
```

Cú pháp tạo đối tượng mới như sau:

var x = new Objects;

String Objects

String là đối tượng dùng để thao tác với các văn bản.

Ví dụ về string:

“ anh yêu em nhiều lắm “ -> đó là một string

“ nhưng em không yêu anh “ -> đó cũng là một string

Để xác định một biến là kiểu string thì chúng chỉ cần khai báo :

var x = “ đoạn văn bản”;

ví dụ : var x = “ tôi yêu việt nam”;

Vấn đề bây giờ là chúng ta phải xét đến các thuộc tính và các phương thức tác động lên đối tượng String như thế nào ?

Các bạn có thể tham khảo bảng các thuộc tính và phương thức của string như sau:

String Object Properties

Property	Description
constructor	Trả về tên chức năng đã tạo ra mẫu string
length	Trả về độ dài của chuỗi
prototype	Cho phép bạn thêm các thuộc tính hoặc phương thức vào đối tượng.

String Object Methods

Method	Description
charAt()	Trả về kí tự của chuỗi ở ví trí xác định
charCodeAt()	Trả về Unicode của kí tự ở vị trí xác định.
concat()	Nối các chuỗi lại với nhau để tạo thành 1 chuỗi mới

fromCharCode()	Chuyển từ mã Unicode thành kí tự
indexOf()	Trả về vị trí xuất hiện đầu tiên của ký tự
lastIndexOf()	Trả về vị trí xuất hiện cuối cùng của kí tự trong chuỗi
match()	Tìm kiếm xâu trong một chuỗi và trả về các xâu đó.
replace()	Thay thế một xâu kí tự trong chuỗi bằng một xâu mới.
search()	Tìm kiếm một xâu kí tự nào đó và trả về vị trí đầu tiên tìm được
slice()	Tách một phần của chuỗi và trả về chuỗi mới
split()	Tách một chuỗi thành một mảng các chuỗi con
substr()	Tách các kí tự từ một chuỗi,bắt đầu ở một vị trí xác định và số ký tự phải tách ra.
substring()	Tách các kí tự từ một chuỗi ,giữa hai chỉ số quy định
toLowerCase()	Trả về một chuỗi toàn chữ thường
toUpperCase()	Trả về một chuỗi toàn chữ Hoa
valueOf()	Trả về giá trị ban đầu của một chuỗi

HTML Wrapper Methods

The HTML wrapper methods trả lại chuỗi bên trong các tag HTML thích hợp.

Method	Description
anchor()	Tạo một anchor
big()	Hiện thị chuỗi ở font lớn nhất
blink()	Hiện thị một chuỗi nhấp nháy
bold()	Hiện thị chuỗi in đậm
fixed()	Hiện thị một chuỗi sử dụng fixed-pitch font
fontcolor()	Hiện thị chuỗi với màu sắc quy định
fontsize()	Hiện thị chuỗi với một kích cỡ xác định
italics()	Hiện thị chuỗi in nghiêng

link()	Hiện thị chuỗi dưới dạng Hypelink
small()	Hiện thị chuỗi với font chữ nhỏ
strike()	Hiện thị một chuỗi có gạch ở giữa
sub()	Ghi chỉ số dưới
sup()	Ghi chỉ số trên

Dưới đâu chúng ta chỉ tìm hiểu một số thuộc tính và method quan trọng thôi,còn lại các bạn có thể tra tại http://www.w3schools.com

Prototype ?

Cho phép bạn có thể thêm thuộc tính hoặc phương thức vào dối tượng

Cú pháp :

```
object.prototype.name=value
```

prototype hỗ trợ trên tất cả các trình duyệt.

ví dụ :

sử dụng prototype để thêm một thuộc tính cho đối tượng.

```
<html>
<body>
<script type="text/javascript">
function employee(name,jobtitle,born)
{
this.name=name;
this.jobtitle=jobtitle;
this.born=born;
}
var fred=new employee("Fred Flintstone","Caveman",1970);
employee.prototype.salary=null;
fred.salary=20000;
document.write(fred.salary);
</script>
</body>
</html>
```

Kết quả hiện thị lên mà hình là

20000

Concat() ?

Phương thức này sẽ cho phép bạn nối các chuỗi lại với nhau để tạo một chuỗi mới.

Cú pháp:

string.concat(string2, string3, ..., stringX)

concat() hỗ trợ ở tất cả các trình duyệt.

```
<html>
<body>
<script type="text/javascript">
var txt1 = "Hello ";
var txt2 = "world!";
document.write(txt1.concat(txt2));
</script>
</body>
</html>
```

Kết quả hiện thị là :

Hello world!

Ví dụ dưới đâu sẽ nối 3 chuỗi với nhau:

```
<html>
<body>
<script type="text/javascript">
var str1="Hello ";
var str2="world!";
var str3=" Have a nice day!";
document.write(str1.concat(str2,str3));
</script>
</body>
</html>
```

Kết quả hiện thị là :

Hello world! Have a nice day!

Match() ?

Phương thức này cho phép bạn tìm xâu trong một chuỗi,kết quả nó sẽ trả về một mảng các xâu tìm được,hoặc nếu như không tìm được thì nó trả về null.

Match() hỗ trợ trong tất cả các trình duyệt.

Ví dụ

```
<html>
<body>
<script type="text/javascript">
var str="The rain in SPAIN stays mainly in the plain";
var patt1=/ain/gi;
document.write(str.match(patt1));
</script>
</body>
```

```
</html>
```

Kết quả hiện thị như sau :

ain,AIN,ain,ain

replace() ?

phương thức này có tác dụng thay thế một xâu nào đó trong chuỗi bằng một xâu khác.

Giá trị trả về của nó là một xâu mới.

Tham số truyền vào là 2 xâu: xâu đầu tiên là xâu của chuỗi,xâu thứ hai là xâu thay thế vào.

Hỗ trợ trên tất cả các trình duyệt.

Ví dụ:

```
<html>
<body>
<script type="text/javascript">
var str="Visit Microsoft!";
document.write(str.replace("Microsoft","W3Schools"));
</script>
</body>
</html>
```

Kết quả hiện thị là :

```
Visit W3Schools!
```

Ví dụ dưới đây sẽ thực hiện việc tìm kiếm nhưng không phân biệt chữ hoa hay thường (chúng ta dùng kí tự / để làm điều này)

```
<html>
<body>
<script type="text/javascript">
var str="Visit Microsoft!";
document.write(str.replace(/microsoft/i,"W3Schools"));
</script>
</body>
</html>
```

Kết quả hiện thị là

```
Visit W3Schools!
```

Split( ) ?

Phương thức này sẽ tách chuỗi thành mảng các chuỗi con.

Hỗ trợ trong tất cả các trình duyệt.

Ví dụ :

```
<html>
<body>
<script type="text/javascript">
```

```
var str="How are you doing today?";
document.write(str.split() + "<br />");
document.write(str.split(" ") + "<br />");
document.write(str.split("") + "<br />");
document.write(str.split(" ",3));
</script>
</body>
</html>
```

Kết quả hiện thị :

How are you doing today?

How,are,you,doing,today?

H,o,w, ,a,r,e, ,y,o,u, ,d,o,i,n,g, ,t,o,d,a,y,?

How,are,you

Substring() ?

Phương thức này sẽ có tác dụng tách một xâu nào đó trong chuỗi.

Giá trị trả về của nó là một chuỗi.

Cú pháp của nó như sau :

```
string.substring(from, to)
```

nó có 2 tham số truyền vào trong đó thì:

from : tham số này bắt buộc phải có,nó xác định xem điểm bắt đầu cho việc lấy kí tự là ở chỗ nào

to : tham số này thì không bắt buộc,nó xác định vị trí kết thúc của quá trình lấy kí tự của xâu.

Hỗ trợ trong tất cả các trình duyệt.

```
<html>
<body>
<script type="text/javascript">
var str="Hello world!";
document.write(str.substring(3)+"<br />");
document.write(str.substring(3,7));
</script>
</body>
</html>
```

Kết quả hiện thị như sau :

lo world!

lo w

getElementById

getElementById đơn giản là một cách thay thế cách tham chiếu tới đối tượng một cách bình thường:

VD: Để tham chiếu tới giá trị của trường text có tên name, bên trong form the_form:

Code:

```
<form name="the_form"> <input type="text" name="name" /> </form>
```

ta dùng code:

Code:

```
document.the_form.name.value
```

Nhưng nếu ta thêm thuộc tính id vào text name:

Code:

```
<form name="the_form"> <input type="text" name="name" id="name" /> </form>
```

ta có thể tham chiếu tới giá trị nó như sau:

Code:

```
document.getElementById("name").value
```

Với **getElementById** ta có thể dễ dàng tham chiếu tới mọi đối tượng, các thẻ div, các tag <p>.....chỉ cần thêm thuộc tính id vào để xác định các thẻ ta muốn làm vệc.

Lời bình:

Chúng ta cũng chú ý rằng trong JavaScrip tên " name " có thể trùng nhau,nhưng "id" thì phải khác nhau cho mọi đối tượng,chính vì vậy dùng "id" cũng là một cách truy cập dữ liệu rất hiệu quả.

Làm việc với thuộc tính innerHTML

innerHTML là phương thức lấy ra cái gì được bọc bởi cặp thẻ nào đó. Đây là một phương thức rất thú vị để làm việc với HTML hoàn toàn dùng javascript(bạn chỉ dùng javascript sinh ra mả html) Ví dụ ta có :

Code:

```
<div id="1"><b id="2"><i>Xem dòng text mẫu</i></b></div>
<script> document.write("Lần thứ hai vẫn thế vì chỉ bỏ cặp thẻ div
"+document.getElementById("1").innerHTML+"<br />");
document.write("nhưng lần sau thì chỉ còn in nghiêng
"+document.getElementById("2").innerHTML+"<br />"); </script>
```

Kết quả của đoạn mã trên:

Xem dòng text mẫu

Lần thứ hai vẫn thế vì chỉ bỏ cặp thẻ div ***Xem dòng text mẫu*** nhưng lần sau thì chỉ còn in nghiêng *Xem dòng text mẫu* Thay đổi định dạng dùng innerHTML. Đơn giản là ta lấy nó ra, đặt lại nó, rồi lại chèn ngược vào. Đó là tất cả những gì trong function changeText()

Code:

```
<script type="text/javascript">
function changeText()
{
var oldHTML = document.getElementById('para').innerHTML;
var newHTML = "<font color=\"#FF0000\">" + oldHTML + "</font>";
document.getElementById('para').innerHTML = newHTML;
}
</script>
<p id='para'>Chào mừng các bạn đến với <b id='boldStuff3'>hocvui.net</b> </p> <input
type='button' onclick='changeText()' value='Change Text'/>
```

Chào mừng các bạn đến với **hocvui.net**

Một ví dụ khác như sau:

ở ví dụ này khi các bạn nhấn vào button " Change link" thì chữ Microsoft sẽ biến thành chữ W3Schools

Qua ví dụ này các bạn sẽ hiểu rõ hơn về chức năng của thuộc tính **innerHTML**

```
<html>
<head>
<script type="text/javascript">
function changeLink()
{
document.getElementById('myAnchor').innerHTML="W3Schools";
document.getElementById('myAnchor').href="http://www.w3schools.com";
document.getElementById('myAnchor').target="_blank";
}
</script>
</head>
<body>
<a id="myAnchor" href="http://www.microsoft.com">Microsoft</a>
<input type="button" onclick="changeLink()" value="Change link">
</body>
</html>
```

Bạn hãy thử chạy code trên sẽ rõ.

Bài thực hành:

Hãy tạo một trang web có các thành phần như sau:

- một button

khi nhấn button đó thì lần thứ nhất sẽ hiện ra dòng chữ " phạm ngọc hiếu",lần hai hiện ra " phạm thị ngọc thúy",lần 3 hiện ra "phạm lê hương" và lần 4 thì hiện ra dòng thông báo "đã in xong 3 cái tên ".Và chú ý rằng các công việc trên thực hiện trên cùng 1 trang web.

Bài làm :

```
<html>
<head>
<title>
pham ngoc hieu
</title>
<script language="javascript">
var i=0;
var x=new Array();
x[0]=" phạm ngọc hiếu";
x[1]=" phạm thị ngọc thúy";
x[2]=" phạm lê hương ";
function ham1()
{
 document.getElementById("khung").innerHTML=x[i];
 i++;
 if(i>3) document.write(" đã in xong 3 cái tên ");
}
</script>
</style></head>
<body>
<div id="khung"> </div>
<input type="button" value="hieu" onclick="ham1();" />
</body>
</html>
```

Làm việc với ngày giờ (Date)

Đối tượng Date() trong JavaScrip được sử dụng để làm việc với ngày tháng và thời gian.

Chúng ta thử một vài ví dụ sau:

Lấy thời điểm của ngày hôm nay(new Date())

```
<html>
```

```
<body>
<script type="text/javascript">
var d=new Date();
document.write(d);
</script>
</body>
</html>
```

⇨ Thu Jan 15 21:52:13 PST 2004

Xác định thứ ngày trong tuần(hàm getDay();)

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
var weekday=new Array(7);
weekday[0]="Sunday";
weekday[1]="Monday";
weekday[2]="Tuesday";
weekday[3]="Wednesday";
weekday[4]="Thursday";
weekday[5]="Friday";
weekday[6]="Saturday";
document.write("Today is " + weekday[d.getDay()]);
</script>
</body>
</html>
```

⇨ Today is Thursday

Cụ thể như sau để lấy thông tin ngày tháng trên hệ thống,trước hết các bạn khai báo một đối tượng kiểu thời gian.

Cú pháp:

Var day = new Date();

Và dùng nó với một số **Method** theo kiểu **day.Method**

Sau đây là một số Method của đối tượng Date

Phương pháp	Mô tả
getDate ()	Trả về ngày của tháng (1-31)
getDay ()	Trả về ngày trong tuần (0-6)
getFullYear ()	Trả về năm (bốn chữ số)
getHours ()	Trả về giờ (0-23)
getMilliseconds ()	Trả về phần nghìn giây (0-999)
getMinutes ()	Trả về phút (0-59)
getMonth ()	Trả về tháng (0-11)
getSeconds ()	Trả về giây (0-59)
getTime ()	Trả về số mili giây kể từ nửa đêm ngày 1 tháng 1, 1970
getTimezoneOffset ()	Trả về sự khác biệt thời gian giữa GMT và giờ địa phương, trong vài phút
getUTCDate ()	Trả về ngày tháng, theo thời gian phổ quát (1-31)
getUTCDay ()	Trả về ngày trong tuần, theo thời gian phổ quát (0-6)
getUTCFullYear ()	Trả về năm, theo thời gian vũ trụ (bốn chữ số)
getUTCHours ()	Trả về giờ, theo thời gian phổ quát (0-23)
getUTCMilliseconds ()	Trả về phần nghìn giây, theo thời gian phổ quát (0-999)
getUTCMinutes ()	Trả về phút, theo thời gian phổ quát (0-59)
getUTCMonth ()	Trả về tháng, theo thời gian phổ quát (0-11)
getUTCSeconds ()	Trả về giây, theo thời gian phổ quát (0-59)
setDate ()	Thiết lập ngày của tháng (1-31)
setFullYear ()	Thiết lập năm (bốn chữ số)
setHours ()	Thiết lập giờ (0-23)
setMilliseconds ()	Thiết lập mili giây (0-999)
setMinutes ()	Cài đặt phút (0-59)
setMonth ()	Thiết lập tháng (0-11)
setSeconds ()	Thiết lập giây (0-59)
Settime ()	Lập một ngày và thời gian bằng cách thêm hoặc trừ đi một số quy định của mili giây đến / từ nửa đêm 01 Tháng Một 1970
setUTCDate ()	Thiết lập ngày tháng, theo thời gian phổ quát (1-31)

setUTCFullYear ()	Bộ năm, theo thời gian vũ trụ (bốn chữ số)
setUTCHours ()	Thiết lập giờ, theo thời gian phổ quát (0-23)
setUTCMilliseconds ()	Thiết lập phần nghìn giây, theo thời gian phổ quát (0-999)
setUTCMinutes ()	Thiết lập biên bản, theo thời gian phổ quát (0-59)
setUTCMonth ()	Thiết lập tháng, theo thời gian phổ quát (0-11)
setUTCSeconds ()	Đặt giây, theo thời gian phổ quát (0-59)
toDateString ()	Chuyển đổi phần ngày của một đối tượng Ngày thành một chuỗi có thể đọc được
toLocaleDateString ()	Trả về phần ngày của một đối tượng Date như một chuỗi, bằng cách sử dụng các công ước miền địa phương
toLocaleTimeString ()	Trả về phần thời gian của một đối tượng Date như một chuỗi, bằng cách sử dụng các công ước miền địa phương
toLocaleString ()	Chuyển đổi một đối tượng Date đến một chuỗi, bằng cách sử dụng miền địa phương ước
toString ()	Chuyển đổi một đối tượng Date đến một chuỗi
toTimeString ()	Chuyển đổi phần thời gian của một đối tượng Date đến một chuỗi
toUTCString ()	Chuyển đổi một đối tượng Date đến một chuỗi, theo thời gian vũ trụ
UTC ()	Trả về số mili giây trong một chuỗi ngày kể từ nửa đêm của ngày 1 tháng 1 năm 1970, theo thời gian vũ trụ
valueOf ()	Trả về giá trị nguyên thủy của một đối tượng Date

Cách cài đặt ngày:

Chúng ta có thể dễ dàng thao tác ngày bằng cách sử dụng các phương pháp có sẵn cho các đối tượng Date.

In the example below we set a Date object to a specific date (14th January 2010): Trong ví dụ dưới đây chúng tôi thiết lập một đối tượng Date đến một ngày cụ thể (14 Tháng 1 năm 2010):

```
var MyDate = new Date ();
myDate.setFullYear (2010,0,14);
```

Và trong ví dụ sau đây chúng tôi thiết lập một đối tượng Date được 5 ngày trong tương lai:

```
var MyDate = new Date ();
myDate.setDate (myDate.getDate ()+ 5);
```

So sánh hai ngày:

Ngày đối tượng cũng được dùng để so sánh hai ngày.

Ví dụ sau ngày hôm nay so với ngày 14 tháng 1 năm 2010:

```
var MyDate = new Date ();
myDate.setFullYear (2010,0,14);
var ngày hôm nay = new Date ();

if (MyDate> ngày hôm nay)
(
alert ("Hôm nay là trước ngày 14 tháng 1 năm 2010");
)
khác
(
alert ("Hôm nay là sau 14 tháng 1 năm 2010");
)
```

Thực ra thì chúng ta chỉ cần chú ý đến một vài hàm thôi.

Hàm **getTime();**

Hàm này trả về thời gian tính bằng mili giây và thời điểm tính mốc của nó để tính là vào ngày 1/1/1970.

Đó chính là điều mà các bạn phải lưu ý.

```
<html>
<body>
<script type="text/javascript">
var d=new Date();
document.write(d.getTime() + " milliseconds since 1970/01/01");
</script>
</body>
</html>
```

Kết quả của đoạn code trên sẽ là

1074236320171 milliseconds since 1970/01/01

Hàm **setFullYear();**

Hàm này dùng để đặt ngày tháng.

Cú pháp của nó là :

```
Date.setFullYear(year,month,day)
```

chú ý rằng hàm này luôn đi kèm với một đối tượng Date

trong đó tham số year là bắt buộc,còn hai tham số còn lại thì không bắt buộc phải có.

```
<html>
<body>
<script type="text/javascript">
var d = new Date();
d.setFullYear(2020);
document.write(d);
</script>
</body>
</html>
```

Kết quả của đoạn code trên sẽ là :

Wed Jan 15 23:12:08 PST 2020

Còn đây là ví dụ thứ 2:

```
<html>
<body>
<script type="text/javascript">
var d = new Date();
d.setFullYear(1992,10,3);
document.write(d);
</script>
</body>
</html>
```

Và đây là kết quả

Tue Nov 3 23:13:57 PST 1992

Hàm **getHours();**

Hàm này trả về giờ hiện tại (từ 0- 23 giờ).

Nó luôn đi với một đối tượng Date.

Cú pháp: Date.getHours();

```
<html>
<body>
```

```
<script type="text/javascript">
var d = new Date();
document.write(d.getHours());
</script>
</body>
</html>
```

Kết quả là

23

Một ví dụ khác

```
<html>
<body>
<script type="text/javascript">
var d = new Date("July 21, 1983 01:15:00");
document.write(d.getHours());
</script>
</body>
</html>
```

Kết quả là

1

Hàm getMinutes();

Hàm này trả về phút hiện tại(từ 0- 59)

Nóa luôn đi với một đối tượng Date

Cú pháp: Date.getMinutes();

Ví dụ:

```
<html>
<body>
<script type="text/javascript">
var d = new Date("July 21, 1983 01:15:00");
document.write(d.getMinutes());
</script>
</body>
</html>
```

Kết quả chắc các bạn cũng biết rồi

15

Hàm getSeconds();

Tương tự như hai hàm trên các bạn tự nghiên cứu.

Ví dụ sau tạo một đồng hồ chạy (như đồng hồ điện tử)

Code:

```
<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date();
var h=today.getHours();
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',500);
}
function checkTime(i)
{
if (i<10)
  {
  i="0" + i;
  }
return i;
}
</script>
</head>
<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

Các bạn hãy thử chạy code trên sẽ thấy chiếc đồng hồ điện tử đang chạy.

Tạo đồng hồ bấm giờ bằng JavaScrip

Bây giờ chúng ta sẽ dùng hai hàm setTimeOut và hàm clearTimeOut để tự tạo cho mình một cái đồng hồ bấm giờ xem sao.

Giao diện của chiếc đồng hồ sẽ như thế này đây:

Đây Đồng Hồ bấm giờ

bắt đầu | dừng lại | tiếp tục

Khi nhấn " bắt đầu " thì đồng hồ sẽ chạy bắt đầu từ thời điểm 0.

Khi nhấn " dừng lại" thì đồng hồ sẽ dừng lại tại thời điểm đó luôn.

Khi nhấn tiếp tục thì đồnghồ sẽ chạy tiếp.

Code nguồn như sau:

```
<html>
<head>
<script language="javascript">
var k=-1;
var t;
var l;
function ham1()
{
  k=k+1;
  document.fom.dongho.value=k;
  t=setTimeout("ham1();",1000);
}
function ham2()
{
  clearTimeout(t);
  k=(-1);
  ham1();
}
function ham3()
{
  clearTimeout(t);
}
</script>
</head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"></head>
<body>
<b><h3> Đây đồng hồ bấm giờ </h3></b><br>
<form name="fom" method="post">
<input type="text" name="dongho"><br><br>
<input type="button" value="bắt đầu" id="bd" onClick="ham2();">
```

```
<input type="button" value="dừng lại" id="dl" onClick="ham3();">
<input type="button" value="tiếp tục" id="tt" onClick="ham1();">
</form>
</body>
</html>
```

Ok vậy là xong cái đồng hồ bấm giờ,thế nhưng cái đồng hồ đó đã thực sự hoàn hảo chưa ?

Chắc là chưa.Để làm rõ nhược điểm của đoạn code này chúng ta hãy thử làm như thế này nhé:

Bạn hãy nhấn " bắt đầu " cho đồng hồ chạy,đợi 1 lát rồi hãy nhấn nút " tiếp tục" tiếp.Đó bây giờ hãy nhìn đồng hồ chạy mà xem nó chạy bị " giật cục ! ",và lúc đó phải nhấn nút " dừng lại" 2 lần thì đồng hồ mới dừng.Tại sao vậy nhỉ ? gợi ý nhé : đó là do chức năng setTimeout đã bi kích hoạt hai lần , để khắc phục điều này chúng ta phải tắt chức năng setTimeout đầu đã rồi mới cho setTimeout thứ hai chạy được (dùng clearTimeout).

Khắc phục cụ thể như sau:

Thêm hàm này nữa:

```
function ham4()
 {
  clearTimeout(t);
  ham1();
 }
```

Chỉnh đoạn code thành

```
<input type="button" value="tiếp tục" id="tt" onClick="ham4();">
```

Đó bây giờ thì nhấn thoải mái chả sợ bị 'giật cục' nữa.

Lời bình:

Qua bài này chúng ta hiểu rõ hơn về cách thức hoạt động của hai hàm ***setTimeout*** *và* ***clearTimeout.***

Tạo chiếc đồng hồ điện tử cho chính bạn

Bây giờ chúng ta sẽ thực hành tạo ra một chiếc đồng hồ điện tử,các bạn phải tạo ra một trình chạy luôn cập nhật giá trị thời gian của hệ thống:

Giao diện thô sơ của chiếc đồng hồ đó như sau :

bây giờ là 1:49:41

Code nguồn :

```
<html>
<head >
<script language="javascript">
 function ham()
 {
```

```
  var x=new Date();
  document.getElementById("gio").innerHTML=x.getHours();
  document.getElementById("phut").innerHTML=x.getMinutes();
  document.getElementById("giay").innerHTML=x.getSeconds();
  setTimeout("ham();",1000);
 }
</script>
</head>
</html>
<body onLoad="ham();">
bây giờ là <b id="gio"> </b>:<b id="phut"> </b>:<b id="giay"> </b>
</body>
</html>
```

Qua bài tập này chúng ta hãy thử làm mộ bài tập nâng cao hơn một chút đó là tạo ra một chiếc đồng hồ điện tử có các chức năng hiện thị giờ hiện tại, hẹn giờ (có âm phát ra đó nhé),có chế độ bấm giờ.

Mảng (Array)

Mảng là một biến đặc biệt,tại cùng một thời điểm nó có thể giữ nhiều hơn một giá trị.

Tác dụng của mảng thì vô cùng to lớn,chẳng hạn là như thế này:

Chẳng hạn bạn có một list danh sách thẻ card và bạn có thể lưu nó bằng cách như sau:

```
cars1="Saab";
cars2="Volvo";
cars3="BMW";
```

đơn giản là chúng ta có thể làm như vậy,thế nhưng nếu số thẻ card là 3000 cái thì bạn định làm thế nào ? không lẽ lại viết lần lượt như trên cho đến khi hết 3000 cái thì thôi à ?! Bó tay !!.

Giải pháp tốt nhất trong trường hợp này là sử dụng mảng (Array).

Mảng có thể giữ tất cả các biến của bạn chỉ với một cái tên duy nhất,và để truy cập đến từng biến ta sẽ thông qua các chỉ số của mảng.

Mỗi phần tử của mảng có một ID riêng của mình để nó có thể dễ dàng truy nhập.

Chú ý rằng chỉ số của mảng bắt đầu từ 0.

Tạo một mảng (Array) ?

Chúng ta có 3 cách để khai báo một mảng.

Trong các trường hợp dưới đây chúng ta khai báo một mảng tên là myCars:

```
var myCars  = new Array ();
myCars [0] = "Saab";
myCars[1]="Volvo";
myCars [2] = "BMW";
```

cách 2:

```
var myCars=new Array("Saab","Volvo","BMW");
```

cách 3:

```
var myCars=["Saab","Volvo","BMW"];
```

truy cập một mảng ?

thông qua chỉ số chúng ta sẽ truy cập vào từng phần tử của mảng.

ví dụ chúng ta sẽ truy cập vào mảng myCars trên:

```
document.write(myCars[0]);
```

kết quả hiện thị sẽ là :

```
Saab
```

Gán một giá trị phần tử của mảng ?

Để gán thì ta cũng chỉ cần truy cập vào phần tử của mảng rồi tiến hành việc gán giá trị mà thôi.

Ví dụ :

```
myCars[0]="Opel";
```

mảng là đối tượng vậy cho nên nó sẽ có thuộc tính và method:

Property	Mô tả
constructor	Trả về chức năng đã tạo ra mẫu thử nghiệm của đối tượng Array
length	trả về số phần tử trong mảng
prototype	Cho phép bạn thêm các thuộc tính và phương pháp vào mảng

Method	Mô tả
concat ()	nối hai mảng với nhau thành một mảng mới
join()	nối các phần tử của mảng thành 1 chuỗi
pop ()	Loại bỏ các yếu tố cuối cùng của một mảng, và trả về yếu tố đó
push ()	Thêm yếu tố mới vào cuối của một mảng, và trả về chiều dài mới
reverse ()	Đảo ngược thứ tự trong một mảng

shift ()	Loại bỏ các yếu tố đầu tiên của một mảng, và trả về yếu tố đó
slice ()	Chọn một phần của một mảng, và trả về mảng mới
sort ()	Phân loại các yếu tố của một mảng
splice ()	Thêm / Loại bỏ các yếu tố từ một mảng
toString ()	Chuyển đổi một mảng cho một chuỗi, và trả về kết quả
unshift ()	Thêm yếu tố mới cho sự khởi đầu của một mảng, và trả về chiều dài mới
valueOf ()	Trả về giá trị nguyên thủy của một mảng

Concat – nối hai mảng với nhau

```
<html>
<body>
<script type="text/javascript">
var parents = ["Jani", "Tove"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(children);
document.write(family);
</script>
</body>
</html>
```

Kết quả : Jani,Tove,Cecilie,Lone

Tương tự vậy nối 3 mạng cũng dễ thôi

```
<html>
<body>
<script type="text/javascript">
var parents = ["Jani", "Tove"];
var brothers = ["Stale", "Kai Jim", "Borge"];
var children = ["Cecilie", "Lone"];
var family = parents.concat(brothers, children);
document.write(family);
</script>
</body>
</html>
```

Kết quả : Jani,Tove,Stale,Kai Jim,Borge,Cecilie,Lone

Join – nối các phần tử của mảng

```
<html>
```

```
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.join() + "<br />");
document.write(fruits.join("+") + "<br />");
document.write(fruits.join(" and "));
</script>
</body>
</html>
```

Kết quả :

Banana,Orange,Apple,Mango

Banana+Orange+Apple+Mango

Banana and Orange and Apple and Mango

Pop – xóa phần tử cuối cùn của mảng

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.pop() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.pop() + "<br />");
document.write(fruits);
</script>
</body>
</html>
```

Kết quả như sau :

Mango

Banana,Orange,Apple

Apple

Banana,Orange

Push – thêm phần tử mới vào cuối mảng

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.push("Kiwi") + "<br />");
document.write(fruits.push("Lemon","Pineapple") + "<br />");
document.write(fruits);
</script>
```

```
</body>
</html>
```

Kết quả là :

Banana,Orange,Apple,Mango,Kiwi,Lemon,Pineapple

Reverse – đảo ngược thứ tự phần tử trong mảng

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.reverse());
</script>
</body>
</html>
```

Kết quả như sau :

Mango,Apple,Orange,Banana

Shift – loại bỏ phần tử đầu tiên của mảng

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.shift() + "<br />");
document.write(fruits + "<br />");
document.write(fruits.shift() + "<br />");
document.write(fruits);
</script>
</body>
</html>
```

Kết quả như sau :

Banana

Orange,Apple,Mango

Orange

Apple,Mango

Slice

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.slice(0,1) + "<br />");
document.write(fruits.slice(1) + "<br />");
```

```
document.write(fruits.slice(-2) + "<br />");
document.write(fruits);
</script>
</body>
</html>
```

Sort – sắp xếp các phần tử của mảng

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.sort());
</script>
</body>
</html>
```

Kết quả hiện thị là :

Apple,Banana,Mango,Orange

Sort numbers

```
<html>
<body>
<script type="text/javascript">
function sortNumber(a, b)
{
return a - b;
}
var n = ["10", "5", "40", "25", "100", "1"];
document.write(n.sort(sortNumber));
</script>
</body>
</html>
```

Kết quả là :

1,5,10,25,40,100

Hoặc kết quả sắp xếp sẽ ngược lại:

```
<html>
<body>
<script type="text/javascript">
function sortNumber(a, b)
{
return b - a;
}
var n = ["10", "5", "40", "25", "100", "1"];
```

```
document.write(n.sort(sortNumber));
</script>
</body>
</html>
```

Kết quả là :

100,40,25,10,5,1

Splice

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write("Removed: " + fruits.splice(2,0,"Lemon") + "<br />");
document.write(fruits);
</script>
</body>
</html>
```

Kết quả là :

Removed:

Banana,Orange,Lemon,Apple,Mango

toString

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.toString());
</script>
</body>
</html>
```

Kết quả hiện thị là :

Banana,Orange,Apple,Mango

unShift

```
<html>
<body>
<script type="text/javascript">
var fruits = ["Banana", "Orange", "Apple", "Mango"];
document.write(fruits.unshift("Kiwi") + "<br />");
document.write(fruits.unshift("Lemon","Pineapple") + "<br />");
document.write(fruits);
</script>
```

```
<p><b>Note:</b> The unshift() method does not work properly in Internet Explorer,
 it only returns undefined!</p>
</body>
</html>
```

Kết quả hiện thị là :

undefined

undefined

Lemon,Pineapple,Kiwi,Banana,Orange,Apple,Mango

Note: The unshift() method does not work properly in Internet Explorer, it only returns undefined!

Boolean Objects

Đối tượng boolean là đối tượng có 2 giá trị hoặc đúng (true) hoặc sai (false).

Đối tượng boolean được sử dụng để chuyển đổi một giá trị không mang kiểu boolean thành kiểu boolean (đúng hoặc sai).

Ví dụ dưới đây tạo một đối tượng Boolean có tên là myBoolean:

```
var myBoolean=new Boolean();
```

chú ý rằng nếu đối tượng Boolean không được thiết lập giá trị ban đầu hoặc nếu nó là 0,null,” ”,false,undefined, hoặc NaN thì đối tượng được thiết lập giá trị là false.Còn lại nó được thiết lập là true.

Tất cả các dòng mã dưới đây đều thiết lập giá trị cho đối tượng là false:

```
var myBoolean = new Boolean ();
var myBoolean = new Boolean (0);
var myBoolean = new Boolean (null);
var myBoolean = new Boolean ("");
var myBoolean = new Boolean (false);
var myBoolean = new Boolean (Nan);
```

và tất cả dòng mã dưới đây đều thiết lập giá trị cho đối tượng là đúng:

```
var myBoolean = new Boolean (true);
var myBoolean = new Boolean ("true");
var myBoolean = new Boolean ("false");
var myBoolean = new Boolean ("Richard");
```

các phương thức và thuộc tính của đối tượng Boolean:

Boolean Object Properties

Bất động sản	Mô tả

constructor	Trả về chức năng đã tạo ra mẫu của đối tượng Boolean
prototype	Cho phép bạn thêm các thuộc tính và phương pháp vào đối tượng

Boolean Object Method

Phương pháp	Mô tả
toString ()	Chuyển đổi một giá trị Boolean thành một chuỗi, và trả về kết quả
valueOf ()	Trả về giá trị ban đầu của một đối tượng Boolean

Constructor

Cú pháp:

```
boolean.constructor
```

ví dụ sau đây trả lại chức năng đã tạo ra mẫu của đối tượng Booblean

```
<html>
<body>
<script type="text/javascript">
var bool=new Boolean();
document.write(bool.constructor);
</script>
</body>
</html>
```

Kết quả hiện thị là :

function Boolean() { [native code] }

toString ()

cú pháp:

```
boolean.toString()
```

ví dụ dưới đây thực hiện việc chuyển đổi giá trị trị Boolean thành String

```
<html>
<body>
<script type="text/javascript">
var bool = new Boolean(1);
document.write(bool.toString());
</script>
</body>
</html>
```

Kết quả :

True

Math Objects

Math Objects là đối tượng thực hiện các thao tác toán học.

Math không phải là một constructor.Tất cả các thuộc tính hay phương thức của nó có thể được gọi bằng cách sử dụng Math như một đối tượng mà không cần tạo nó.

Cú pháp cho việc sử dụng thuộc tính và phương thức của Math:

```
var pi_value=Math.PI;
var sqrt_value=Math.sqrt(16);
```

hằng số toán học

JS cung cấp tám hắng số toán học mà có thể được truy cập từ đối tượng Math.Đó là : E , PI, square root of 2, square root of 1/2, natural log of 2, natural log of 10, base-2 log of E, và base-10 log of E. Bạn có thể tham khảo ở bảng sau:

```
Math.E
Math.PI
Math.SQRT2
Math.SQRT1_2
Math.LN2
Math.LN10
Math.LOG2E
Math.LOG10E
```

Những method có sẵn

Round() – làm tròn

```
<html>
<body>
<script type="text/javascript">
document.write(Math.round(0.60) + "<br />");
document.write(Math.round(0.50) + "<br />");
document.write(Math.round(0.49) + "<br />");
document.write(Math.round(-4.40) + "<br />");
document.write(Math.round(-4.60));
</script>
</body>
</html>
```

Kết quả:

1

1

0

-4

-5

Random() – tạo số ngẫu nhiên

```
<html>
<body>
<script type="text/javascript">
//return a random number between 0 and 1
document.write(Math.random() + "<br />");
//return a random integer between 0 and 10
document.write(Math.floor(Math.random()*11));
</script>
</body>
</html>
```

Kết quả :

0.950635502524875

3

Max() – tìm ra số lớn nhất

```
<html>
<body>
<script type="text/javascript">
document.write(Math.max(5,10) + "<br />");
document.write(Math.max(0,150,30,20,38) + "<br />");
document.write(Math.max(-5,10) + "<br />");
document.write(Math.max(-5,-10) + "<br />");
document.write(Math.max(1.5,2.5));
</script>
</body>
</html>
```

Kết quả :

10

150

10

-5

2.5

Min() – tìm ra số bé nhất

```
<html>
<body>
<script type="text/javascript">
```

```
document.write(Math.min(5,10) + "<br />");
document.write(Math.min(0,150,30,20,38) + "<br />");
document.write(Math.min(-5,10) + "<br />");
document.write(Math.min(-5,-10) + "<br />");
document.write(Math.min(1.5,2.5));
</script>
</body>
</html>
```

Kết quả :

5

0

-5

-10

1.5

Sau đây là bảng các method của Math các bạn có thể tham khảo

Phương pháp	Mô tả
abs (x)	Trả về giá trị tuyệt đối của x
acos (x)	Trả về arccosine của x, trong radian
asin (x)	Trả về arcsine của x, trong radian
atan (x)	Trả về arctangent của x là một giá trị số giữa-PI / 2 và PI / 2 radian
atan2 (y, x)	Trả về arctangent của thương của đối số của nó
ceil(x)	Trả về x, được làm tròn lên đến số nguyên gần nhất
cos (x)	Trả về cosin của x (x là radian)
exp (x)	Trả về giá trị của E^x
floor (x)	Trả về x, được làm tròn xuống đến số nguyên gần nhất
log (x)	Trả về logarit tự nhiên (cơ số E) của x
max (x, y, z ,..., n)	Trả về giá trị lớn nhất
min (x, y, z ,..., n)	Trả về giá trị bé nhất
pow (x, y)	Trả về giá trị của x mũ y

random ()	Trả về một số ngẫu nhiên giữa 0 và 1
round (x)	làm tròn x đến số nguyên gần nhất
sin (x)	Trả về sin của x (x là radian)
sqrt (x)	Trả về căn bậc hai của x
tan (x)	Trả về tang của một góc

Tạo số ngẫu nhiên trong JavaScript

Việc tạo ra các giá trị ngẫu nhiên là hết sức quan trọng,đó là lí do tại sao trong tất cả các ngôn ngữ lập trình luôn luôn hộ trợ chức năng này.JavaScript cũng vậy,nó cung cấp cho chúng ta chức năng tạo giá trị ngẫu nhiên bằng cách sử dụng hàm **math.random()** ,thế nhưng chỉ có điều là chỉ duy nhất hàm này trong JavaScript là thực hiện điều này thôi,và nó chỉ tạo ra giá trị ngẫu nhiên trong khoảng từ 0 đến 1.

Ví dụ :

X= math.random( );

Thì x sẽ nhận một giá trị nào đó trong khoảng từ 0 đến 1, ví dụ x= 0,2645.

Đến đây chắc bạn sẽ có thắc mắc vậy muốn tạo số ngẫu nhiên trong phạm vi lớn hơn thì làm thế nào ?

Câu trả lời là : thực ra chỉ như vậy thôi các bạn cũng có thể tạo ra số ngẫu nhiên trong khoảng bất kì rồi,cách làm là nhân với một giá trị nào đó, ví dụ :

X= Math.random( ) * 100;

Như thế rõ ràng giá trị ngẫu nhiên của x sẽ đi từ 0 tới 100.

Nhưng có thể một số người lại muốn lấy giá trị ngẫu nhiên trong khoảng từ 0 đến 100 nhưng lại muốn lấy số nguyên thôi vậy thì ta làm như sau :

X=Math.floor(Math.random( ) *101);

Các bạn chú ý rằng Math.floor( ) là phương thức làm tròn xuống đến số nguyên gần nhất,đó là lý do tại sao chúng ta lại phải nhân với 101 ,chứ không phải là nhân với 100;

So mẫu (RegExp) trong JavaScript

Như thé nào là so mẫu (RegExp)?

Nó là một đối tượng mô tả mô hình của kí tự.

Khi bạn tìm kiếm trong văn bản,bạn có thể sử dụng một mô hình để mô tả những gì bạn đang tìm kiếm.

Một mô hình đơn giản có thể là một kí tự duy nhất.

Một mô hình phức tạp hơn có thể có nhiều kí tự hơn,và có thể được dùng để phân tích,định dạng kiểm tra,thay thế hoặc nhiều việc khác nữa.

Regexp thường được sử dụng để thực hiện chức năng tìm kiếm và thay thế trên văn bản.

Cú pháp:

```
var txt=new RegExp(pattern,modifiers);
```

hoặc đơn giản hơn

```
var txt=/pattern/modifiers;
```

như chúng ta đã thấy trong cú pháp trên có hai trường là pattern và modifiers,chúng ta sẽ lần lượt tìm hiểu chúng.

RegExp modifiers ?

Cờ modifiers được sử dụng để báo cho trình duyệt biết rằng là tìm kiếm không phân biệt chữ hoa hay chữ thường ? hay tìm kiếm trên toàn bộ văn bản? hay tìm kiếm trên toàn bộ văn bản và không phân biệt chữ hoa hay thường ?

Modifiers i được sử dụng để : tìm kiếm không phân biệt chữ hoa hay chữ thường.

Modifiers g được sử dụng để : tìm kiếm trên toàn bộ văn bản.

Modifiers ig được sử dụng để : tìm kiếm trên toàn bộ văn bản và không phân biệt chữ hoa và chữ thường.

Ví dụ sau sẽ thực hiện việc tìm kiếm từ “w3schools” trong một chuỗi :

```
<html>
<body>
<script type="text/javascript">
var str = "Visit W3Schools";
var patt1 = /w3schools/i;
document.write(str.match(patt1));
</script>
</body>
</html>
```

Kết quả :

W3Schools

Ví dụ sau sẽ thực hiện việc tìm kiếm từ “is” trên toàn bộ văn bản (theo mặc định thì trình duyệt sẽ tìm thấy từ đầu tiên trong văn bản sẽ dừng lại):

```
<html>
<body>
<script type="text/javascript">
var str="Is this all there is?";
```

```
var patt1=/is/g;
document.write(str.match(patt1));
</script>
</body>
</html>
```

Kết quả hiện thị là :

is,is

ví dụ dưới đây sẽ thực hiện việc tìm kiếm trên toàn bộ văn bản và không phân biệt chữ hoa chữ thường:

```
<html>
<body>
<script type="text/javascript">
var str="Is this all there is?";
var patt1=/is/gi;
document.write(str.match(patt1));
</script>
</body>
</html>
```

Kết quả hiện thị là :

Is,is,is

Hai phương thức (method) quan trọng của đối tượng RegExp

Test() ?

Phương thức này sẽ cho biết xem một chuỗi có nằm trong một chuỗi khác hay không ?

Nếu có thì trả về true.

Nếu sai thì trả về false.

Ví dụ dưới đây kiểm tra xem kí tự e có nằm trong chuỗi hay không ?

```
<html>
<body>
<script type="text/javascript">
var patt1=new RegExp("e");
document.write(patt1.test("The best things in life are free"));
</script>
</body>
</html>
```

Kết quả hiện thị :

True

Exec() ?

Phương thức này cũng giống phương thức trên chỉ có điều nó sẽ trả về giá trị nội dung được tìm thấy.

Ví dụ sau tìm kiếm chuỗi kí tự "e"

```
<html>
<body>
<script type="text/javascript">
var patt1=new RegExp("e");
document.write(patt1.exec("The best things in life are free"));
</script>
</body>
</html>
```

Kết quả hiện thị sẽ là :

e

chúng ta có thể tham khảo bảng sau (đầy đủ hết,tuy nhiên một số cái ít khi dùng lắm)

Modifiers

Modifiers are used to perform case-insensitive and global searches:

Modifier	Description
i	Perform case-insensitive matching
g	Perform a global match (find all matches rather than stopping after the first match)
m	Perform multiline matching

Brackets

Brackets are used to find a range of characters:

Expression	Description
[abc]	Find any character between the brackets
[^abc]	Find any character not between the brackets
[0-9]	Find any digit from 0 to 9
[A-Z]	Find any character from uppercase A to uppercase Z
[a-z]	Find any character from lowercase a to lowercase z
[A-z]	Find any character from uppercase A to lowercase z

[adgk]	Find any character in the given set
[^adgk]	Find any character outside the given set
(red|blue|green)	Find any of the alternatives specified

Metacharacters

Metacharacters are characters with a special meaning:

Metacharacter	Description
.	Find a single character, except newline or line terminator
\w	Find a word character
\W	Find a non-word character
\d	Find a digit
\D	Find a non-digit character
\s	Find a whitespace character
\S	Find a non-whitespace character
\b	Find a match at the beginning/end of a word
\B	Find a match not at the beginning/end of a word
\0	Find a NUL character
\n	Find a new line character
\f	Find a form feed character
\r	Find a carriage return character
\t	Find a tab character
\v	Find a vertical tab character
\xxx	Find the character specified by an octal number xxx
\xdd	Find the character specified by a hexadecimal number dd
\uxxxx	Find the Unicode character specified by a hexadecimal number xxxx

Quantifiers

Quantifier	Description

n+	Matches any string that contains at least one n
n*	Matches any string that contains zero or more occurrences of n
n?	Matches any string that contains zero or one occurrences of n
n{X}	Matches any string that contains a sequence of X n's
n{X,Y}	Matches any string that contains a sequence of X or Y n's
n{X,}	Matches any string that contains a sequence of at least X n's
n$	Matches any string with n at the end of it
^n	Matches any string with n at the beginning of it
?=n	Matches any string that is followed by a specific string n
?!n	Matches any string that is not followed by a specific string n

RegExp Object Properties

Property	Description
global	Specifies if the "g" modifier is set
ignoreCase	Specifies if the "i" modifier is set
lastIndex	The index at which to start the next match
multiline	Specifies if the "m" modifier is set
source	The text of the RegExp pattern

RegExp Object Methods

Method	Description
compile()	Compiles a regular expression
exec()	Tests for a match in a string. Returns the first match
test()	Tests for a match in a string. Returns true or false

Bài thực hành 1:

1. Viết một đoạn lệnh JavaScript sử dụng cách thức confirm() và câu lệnh if...then

để thực hiện:

Hỏi người sử dụng có muốn nhận được một lời chào không

Nếu có thì hiện ảnh wellcome.jpg và một lời chào.

Nếu không thì viết ra một lời thông báo

2. Viết một đoạn lệnh JavaScript để thực hiện:

· Hỏi người sử dụng: "10+10 bằng bao nhiêu?"

· Nếu đúng thì hỏi tiếp: Có muốn trả lời câu thứ hai không?"

· Nếu muốn thì hỏi: "10*10 bằng bao nhiêu?"

· Đánh giá kết quả bằng biểu thức logic sau đó viết ra màn hình:

Đúng: "CORRECT"; Sai: "INCORRECT"

3. Câu lệnh nào trong các câu sau đây sử dụng sai thẻ sự kiện

a. <BODY onClick="doSomething();">

b. <INPUT TYPE=text onFocus="doSomething();">

c. <INPUT TYPE=textarea onLoad="doSomething();">

d. <BODY onUnload="doSomething();">

e. <FORM onLoad="doSomething();">

f. <FORM onSubmit="doSomething();">

4. Điều gì xảy ra khi thực hiện script sau:

```
<HTML>
<HEAD>
<TITLE>Exercise 5.4</TITLE>
<SCRIPT LANGUAGE="JavaScript">
<!-- HIDE FROM OTHER BROWSERS
var name = "";
function welcome() {
name = prompt("Welcome to my page! What's Your Name?","name");
}
function farewell() {
alert("Goodbye " + name + ". Thanks for visiting my page.");
}
// STOP HIDING FROM OTHER BROWSERS -->
</SCRIPT>
</HEAD>
<BODY onLoad="welcome();" onUnload="farewell();";>
```

This is my page!

</BODY>

</HTML>

5. Sử dụng vòng lặp while để mô phỏng các vòng lặp for sau:

a.

for (j = 4; j > 0; j --) {

document.writeln(j + "<BR>");

}

b.

for (k = 1; k <= 99; k = k*2) {

k = k/1.5;

}

c.

for (num = 0; num <= 10; num ++) {

if (num == 8)

break;

}

6. hãy xây dựng một trang web có chức năng như sau:

" thực hiện bài kiểm tra"

Xuất hiện 2 button một cái là " vào bài kiểm tra" một cái là " không làm "

Khi nhấn nút "vào bài kiểm tra" thì bài kiểm tra phép tính bắt đầu với các số và phép toán là ngẫu nhiên.Nếu trả lời sai thì thoát và hiện thông báo " bạn về học lại đi " ,nếu đúng thì hiện câu hỏi tiếp theo cho đến câu 10 thì thoát và hiện thông báo " quá giỏi".

<u>Bài làm</u>

1)

```
<html>
<head>
<title>
thuc hanh JS
</title>
<script language="javascript">
function loi_chao()
  {
    document.write("<h1> chào bạn Phạm Ngọc Hiếu </h1>");
 document.write("<img src='ap_20090712075315776.jpg'>");
```

```
  }
function loi()
  {
    document.write(" <h1> không chào thì thôi bye bye </h1> ");
  }
function thong_bao()
  {
    var t=confirm("bạn có muốn hiện lời chào không ");
 if(t==true)  loi_chao();
 else      loi();
  }
</script>
<head>
<body>
<script language="javascript">
thong_bao();
</script>
</body>
</html>
```

2)

```
<html>
<head>
<title>
thực hành JS
</title>
<script language="javascript">
function ham()
 {
   if(document.fom.ket_qua.value==eval("10+10"))
     {
       var l= confirm(" bạn có muốn tiếp tục câu thứ hai không ");
    if(l==true)
    ham2();
    else ham3();
  }
   else
     {
       document.write("<h1>bạn trả lời sai rồi bye bye </h1>");
    setTimeout("ham3()",3000);
  }
 }
```

```
 function ham3()
 {
   window.close();
 }
function ham2()
 {
   var x=prompt("10*10 bằng bao nhiêu");
   if(x==eval(10*10))
     {
    document.write("<h1> bạn trả lời đúng rồi </h1>");
    ham3();
  }
   else
     {
    document.write("<h1> bạn trả lời sai rồi </h1>");
    ham3();
  }
 }
</script>
</head>
<body>
<form method="post" name="fom" >
10+10=
<input type="text" name="ket_qua">
<input type="button" value="xác nhận" onClick="ham();">
</form>
</body>
</html>
```

3.

a,c,e là sai còn lại là đúng.

4.

Khi mở trang web thì lời chào hiện lên cùng hộp thoại yêu cầu nhập tên vào.

Khi thoát trang web thì lời chòa cùng với tên hiện lên.

5.

a)

```
  j=4;
  while(j>0)
  {
   Document.writeln(j +"</br>");
```

```
    j--;
   }
 b)
  k=1;
while(k<=99)
{
  k=k/1.5;
  k=k*2;
}
c)
 num=0;
while(num <=10)
{
   If(num==8) break;
   Num++;
}
```

Bài thực hành 2:

hãy thử tạo một trang web có mã như sau,nhìn cách thức hoạt động của nó rồi hãy tạo một trang web có chức năng tương tự (tất nhiên là phải do bạn tự viết).

(code hơi dài nhưng đừng thế mà choáng nhé,thực ra cũng chẳng có gì cả).

```
<html>
<head>
<title>
thực hành JavaScript
</title>
<style>
b h1{color:#FF0000}
</style>
<script language="javascript">
var i=-1;
var j=-1;
var dung_sai=new Array();
    dung_sai[0]="<b><h1>đúng</h1></b>";
```

```
 dung_sai[1]="<b><h1>sai</h1></b>";
var cau_hoi=new Array();
    cau_hoi[0]=" nhà bác học việt nam nào sau đây vừa đạt được một danh hiệu toán
 học cao cấp của                  thế giới ?";
 cau_hoi[1]=" câu thủ nào sau đây vừa được ManCity mua về ? ";
var tra_loi2=new Array();
    tra_loi2[0]=" a) <a href='#' onclick='div_sai();'> hoàng tụy </a> ";
 tra_loi2[1]=" b) <a href='#'  onclick='div_dung();'> ngô bảo châu </a> ";
 tra_loi2[2]=" c) <a href='#'  onclick='div_sai();'> văn như cương </a> ";
 tra_loi2[3]=" d) <a href='#'  onclick='div_sai();'> đào hữu vinh </a> ";
 tra_loi2[4]=" a) <a href='#'  onclick='div_sai();'> xavi </a> ";
 tra_loi2[5]=" b) <a href='#'  onclick='div_sai();'> elano </a> ";
 tra_loi2[6]=" c) <a href='#'  onclick='div_sai();'> kaka </a> ";
 tra_loi2[7]=" d) <a href='#'  onclick='div_dung();'> boateng </a> ";
function dung()
   {
     i=i+1;
  if(j==7) thoat();
  document.getElementById("chut").innerHTML="";
     document.getElementById("the1").innerHTML=cau_hoi[i];
  document.getElementById("traloi0").innerHTML=tra_loi2[++j];
  document.getElementById("traloi1").innerHTML=tra_loi2[++j];
  document.getElementById("traloi2").innerHTML=tra_loi2[++j];
  document.getElementById("traloi3").innerHTML=tra_loi2[++j];
   }
function div_dung()
{
  document.getElementById("chut").innerHTML=dung_sai[0];
}
function div_sai()
{
  document.getElementById("chut").innerHTML=dung_sai[1];
}
function thoat1()
{
 window.close();
}
function thoat()
{
 document.write(" bài kiểm tra đã kết thúc ,tạm biệt bạn chúc bạn học giỏi ");
 t=setTimeout("thoat1();",2000);
```

```
}
</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8">
</head>
<body>
<b><i><h1> bài kiểm tra IQ</h1></i></b>
<div id="the" style="top:129px; left:12px; width:400; height:300;
 position:absolute">
 <div id="the1" style="top:1px; left:1px; width:400; height:100;
 position:absolute"> game online nào sau đây được bình chọn là game hay nhất trong
 năm 2009 ?</div>
 <div id="traloi0" style="top:101px; left:1px; width:400; height:50;
 position:absolute"> a) <a href="#" onClick="div_dung();"> thiên long bát bộ </a>
 </div>
<div id="traloi1" style="top:152px; left:0px; width:400; height:50;
 position:absolute"> b) <a href="#" onClick="div_sai();"> Alantica </a> </div>
 <div id="traloi2" style="top:203px; left:1px; width:400; height:50;
 position:absolute">  c) <a href="#" onClick="div_sai();"> kiếm thế </a> </div>
 <div id="traloi3" style="top:255px; left:1px; width:400; height:50;
 position:absolute"> d) <a href="#" onClick="div_sai();"> võ lâm truyền kì </a>
 </div>
</div>
<div id="thom" style="width:60; height:30; position:absolute; left: 245px; top:
 377px;">
  <input name="button" type="button" value="tiếp" onClick="dung();" >
</div>
<div id="chut" style="position:absolute; width:160; height:50; left: 197px; top:
 171px;">  </div>
</body>
</html>
```

Lời bình:

Thực ra thì bài tập này mình cũng làm khá vất vả mới hoàn thành (do trình độ có hạn),sau khi làm mình thấy phải chú ý những điều sau:

- phải hết sức cận thận khi trong một chuỗi mà có nhiều hơn 2 dấu " (dấu nháy kép),đơn giản là bởi vì ví như các bạn có chuỗi sau :

  "anh yêu em "nhưng em không yêu" anh "

  Thế thì máy nó sẽ không hiểu đó là 1 chuỗi,các bạn phải viết lại như sau:

  "anh yêu em'nhưng em không yêu'anh"

  Sử dụng nháy đơn sẽ giả quyết được trường hợp này.

- trong quá trình điều khiển vị trí cho khung thì chú ý rằng để cài đặt được vị trí thì chúng ta phải đặt thông số : " position:absolute" cho nó.không có thông số này thì coi như khung "chết cứng".Còn để điều chỉnh vị trí cho đẹp thì chuyển qua cửa sổ design rồi kéo thả đến vị trí thích hợp,các thông số trong cửa sổ code sẽ tự động điều chỉnh.
- Qua đây chúng ta có thể thấy được công dụng rất hiệu quả của chức năng getElementById(),và innerHTML .
- " ++j" và "j++" là cùng ý nghĩa nhưng khác nhau,một cái là cộng trước sau đó mới dùng,còn một cái là dùng trước sau đó mới cộng.không tin các bạn thử thay "++j" ở đọn code trên bằng "j++" mà xem.
- Khi muốn cho một liên kết thực hiện một hàm nào đó (mà không nhảy tới trang khác) thì chúng ta có hai cách :

  _ hoặc chúng ta sử dụng phép gán href="#"

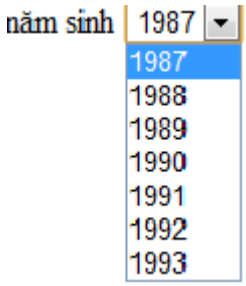
  _ hoặc là chúng ta href="javascript: ham();"

bài thực hành 3

Dùng JavaScript tạo một form lựa chọn theo kiểu select.

Bài làm

```
<script language="javascript">
for(i=1987;i<1994;i++)
{
document.write("<option>"+i+" </option>");
}
</script>
```



Với phương pháp này chúng ta khỏi cần phải sử dụng lặp lại cú pháp <option> </option>.

Bài thực hành 4

ở bài thực hành này chúng ta sẽ tạo ra một form nhập liệu để gửi lên sever lưu vào cơ sở dữ liệu như sau :

trang đăng kí thành viên

tên đăng nhập phamngochieu

mật khẩu ••••••••••

nhắc lại mật khẩu •••••••

Gửi

ban da nhap sai mat khau

Nếu như phần mật khẩu và phần nhắc lại mật khẩu khác nhau thì một dòng thông báo " bạn đã nhập sai mật khẩu " hiện lên.

Bài làm :

```
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>login</title>
<script language="javascript">
 function ham()
 {
   if(document.fom.matkhau1.value != document.fom.matkhau2.value)
   {

  document.getElementById("hay").innerHTML="ban da nhap sai mat khau";
  return false;
   }
   else
   return true;
 }
</script>
</head>
<body>
<h1> trang đăng kí thành viên </h1> <br /><br />
```

```
<form  method="post" onsubmit="return ham();" name="fom" action="dang_ki.php">
tên đăng nhập <input type="text" name="ten" style="position:absolute; left:
 149px;" />
<br /><br />
mật khẩu  <input type="password" name="matkhau1" style="position:absolute; left:
 147px;" />
<br /><br />
nhắc lại mật khẩu <input type="password" name="matkhau2" style="position:absolute;
 left: 146px;" />
<br /><br />
<input type="submit" />
</form>
<div id="hay"> </div>
</body>
</html>
```

Qua ví dụ này các bạn phải lưu ý một điều như sau :

ở dòng

```
<form  method="post" onsubmit="return ham();" name="fom" action="dang_ki.php">
```

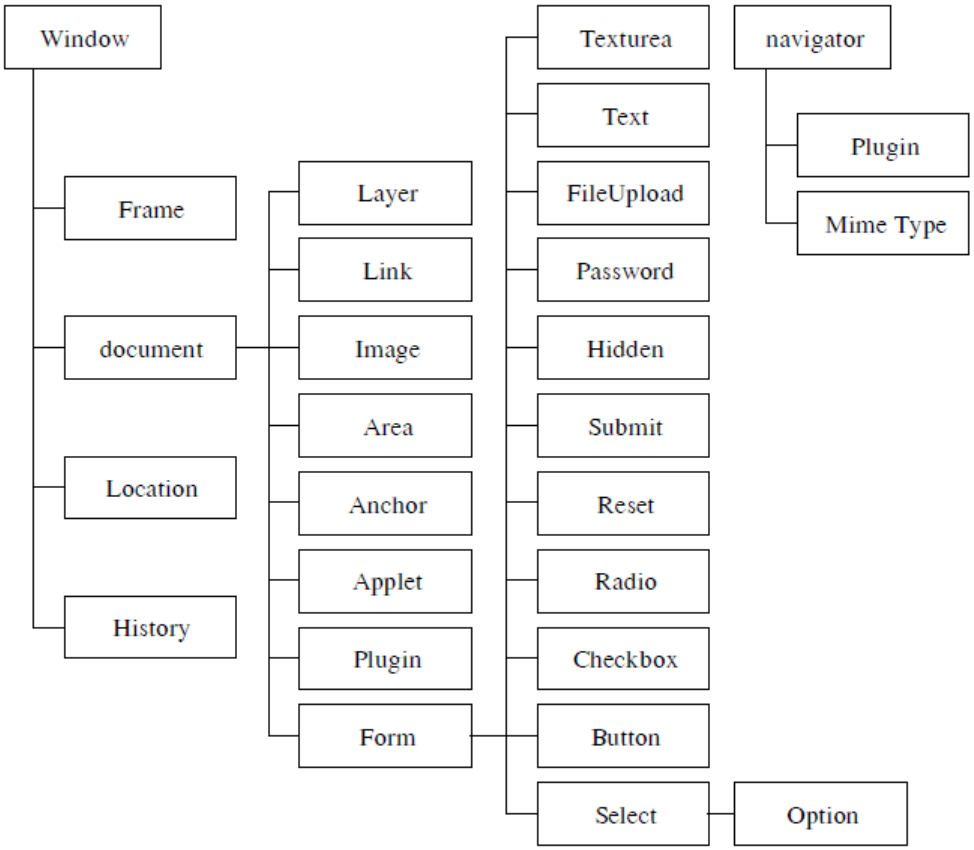
Các bạn phải thêm từ khóa return trong biến cố onsubmit.Nếu không có từ khóa này các bạn sẽ gặp rắc rối to đấy.

Chương 3

JS advanced

Sự phân cấp đối tượng trong JS

Trong JavaScrip đối tượng được phân ở các cấp khác nhau,lớn nhất,rộng nhất là đối tượng window,tiếp theo là document.,history,…Chúng ta có thể tham khảo sơ đồ sau:



JavaScript Navigator

Hầu hết các câu lệnh của JavaScript đều chạy trên mọi trình duyệt,thế nhưng điều đó cũng có nghĩa là sẽ có một số sẽ không chạy trên một loại trình duyệt nào đó,đặc biệt là trình duyệt cũ.

Để có thể khắc phục được nhược điểm này chúng ta sẽ làm cho trang web có khả năng nhận biết được loại trình duyệt mà người dùng đang sử dụng.

Các đối tượng Navigator chứa thông tin về trình duyệt và thông tin phiên bản của khách truy nhập,và nhiều thông tin khác nữa.

Navigator không có các phương thức mà nó chỉ có các thuộc tính:

appCodeName

Xác định tên mã nội tại của trình duyệt (Atlas).

AppName

Xác định tên trình duyệt.

AppVersion

Xác định thông tin về phiên bản của đối tượng navigator.

userAgent

Xác định header của user - agent.

Ví dụ:

```
<html>
<body>
<script type="text/javascript">
document.write("Browser CodeName: " + navigator.appCodeName);
document.write("<br /><br />");
document.write("Browser Name: " + navigator.appName);
document.write("<br /><br />");
document.write("Browser Version: " + navigator.appVersion);
document.write("<br /><br />");
document.write("Cookies Enabled: " + navigator.cookieEnabled);
document.write("<br /><br />");
document.write("Platform: " + navigator.platform);
document.write("<br /><br />");
document.write("User-agent header: " + navigator.userAgent);
</script>
</body>
</html>
```

Kết quả hiện thị như sau:

Browser CodeName: Mozilla

Browser Name: Microsoft Internet Explorer

Browser Version: 4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; AskTB5.4)

Cookies Enabled: true

Platform: Win32

User-agent header: Mozilla/4.0 (compatible; MSIE 8.0; Windows NT 5.1; Trident/4.0; AskTB5.4)

Bài thực hành:

Hãy nghiêm chỉnh hoàn thiện bài thực hành sau:

Viết code JS xác định tên trình duyệt,phiên bản của trình duyệt bạn đang sử dụng.

Đáp án:

```
<script language="javascript">
  var x=navigator.appName;
  var y=navigator.appCodeName;
  var z=navigator.appVersion;
  var t=navigator.userAgent;
  document.writeln("trình duyệt mà các bản đang dùng là "+x);
```

```
document.writeln("phiên bản đang dùng là "+z);
document.writeln("mã nội tại của trình duyệt là"+y);
document.writeln("header"+t);
</script>
```

javaScript window

window là đối tượng ở mức cao nhất,nó có khá nhiều thuộc tính và phương thức quan trọng. sau đây chúng ta chỉ tìm hiểu một vài thuộc tính và phương thức quan trọng nhất.

thuộc tính

closed

giá trị trả về của nó là một giá trị boolean cho biết cửa sổ đã đóng hay chưa.

Cú pháp:

Window.closed

Hỗ trợ trong tất cả các trình duyệt.

ví dụ

```
<html>
<head>
<script type="text/javascript">
function openWin()
{
myWindow=window.open('','','width=200,height=100');
myWindow.document.write("<p>This is 'myWindow'</p>");
}
function closeWin()
{
myWindow.close()
}

function checkWin()
{
if (myWindow.closed)
  {
  document.getElementById("msg").innerHTML="'myWindow' has been closed!";
  }
else
  {
  document.getElementById("msg").innerHTML="'myWindow' has not been closed!";
  }
```

```
}
</script>
</head>
<body>
<input type="button" value="Open 'myWindow'" onclick="openWin()" />
<input type="button" value="Close 'myWindow'" onclick="closeWin()" />
<br /><br />
<input type="button" value="Has 'myWindow' been closed?" onclick="checkWin()" />
<br /><br />
<div id="msg"></div>
</body>
</html>
```

Hãy thử chạy đoạn code trên và tìm hiểu nó.

defaultStatus

Giá trị trả về của nó là một chuỗi văn bản được hiện thị trên thanh trạng thái (phía dưới trình duyệt).

Thuộc tính này không hoạt động trên trạng thái mặc định của IE,Crome,fifox,muốn nó hoạt động được bạn phải điều chỉnh lại mặc định của các trình duyệt trên.

Cú pháp:

Window.defaultStatus

Ví dụ :

```
<html>
<body>
<script type="text/javascript">
window.defaultStatus="This is the default text in the status bar!!";
</script>
<p>Look at the text in the statusbar.</p>
<p><b>Note:</b> The defaultStatus property does not work in the default
 configuration of IE, Firefox, Chrome, or Safari.</p>
</body>
</html>
```

Frames

Đây là thuộc tính của window,nó trả về một mảng các frame(kể cả iframe) trong trang.

Để xác định được số phần tử của mảng này ta dùng thuộc tính length.(frames.length).

Cú pháp:

Window.frames

Hỗ trợ trong tất cả các trình duyệt.

Ví dụ sau đây sẽ thực hiện việc tìm tất cả các frame trong một trang và thay đổi thuộc tính src của các frame đó thành địa chỉ “dantri.vn”

```
<html>
<body>
<iframe src="http://microsoft.com"></iframe>
<iframe src="http://google.com"></iframe>
<iframe src="http://youtube.com"></iframe>
<script type="text/javascript">
for (var i=0; i<frames.length; i++)
  {
  frames[i].location="http://dantri.vn"
  }
</script>
</body>
</html>
```

Length

Thuộc tính này trả về số frame(kể cả iframe) trong một cửa sổ.

Cú pháp:

Window.length

Hỗ trợ trong tất cả các trình duyệt.

Name

Trả về tên của cửa sổ hiện thời.

Cú pháp:

Window.name

Hỗ trợ trong tất cả các trình duyệt.

Ví dụ sau trả về tên của cửa sổ mới:

```
<html>
<head>
<script type="text/javascript">
function openWin()
{
myWindow=window.open('','MsgWindow','width=200,height=100');
myWindow.document.write("<p>This window's name is: " + myWindow.name +
"</p>");
}
</script>
</head>
<body>
<input type="button" value="Open 'myWindow'" onclick="openWin()" />
</body>
</html>
```

Parent

Trả về cửa sổ cha mẹ của cửa sổ hiện thời.

Cú pháp:

Window.parent

Hỗ trợ trong tất cả các trình duyệt.

Ví dụ sau đây chúng ta sẽ cho xuất hiện một cửa sổ mới(cửa sổ con) và thông báo tên của cửa sổ cha mẹ của nó.

```
<html>
<head>
<script type="text/javascript">
function openWin()
{
window.open('','','width=200,height=100');
alert(window.parent.location);
}
</script>
</head>
<body>
<input type="button" value="Open window" onclick="openWin()" />
</body>
</html>
```

Self

Trả về cửa sổ hiện thời.

Cú pháp:

Window.self

Hỗ trợ trong tất cả các trình duyệt.

Ví dụ

```
<html>
<head>
<script type="text/javascript">
function check()
{
if (window.top!=window.self)
  {
  document.write("<p>This window is not the topmost window! Am I in a frame?</p>")
```

```
  }
else
  {
  document.write("<p>This window is the topmost window!</p>")
  }
}
</script>
</head>
<body>
<input type="button" onclick="check()" value="Check window" />
</body>
</html>
```

Top

Trả về cửa sổ trên cùng.

Cú pháp:

Window.top

Hỗ trợ trong tất cả các trình duyệt.

Ví dụ :

```
<html>
<head>
<script type="text/javascript">
function check()
{
if (window.top!=window.self)
  {
  document.write("<p>This window is not the topmost window! Am I in a frame?</p>")
  }
else
  {
  document.write("<p>This window is the topmost window!</p>")
  }
}
</script>
</head>
<body>
<input type="button" onclick="check()" value="Check window" />
</body>
</html>
```

Phương thức

Sau đây là các phương thức quan trọng trong window.

Open()

Mở một cửa sổ mới.

Hỗ trợ trong tất cả các trình duyệt.

Cú pháp:

window.open *(URL, tên, thông số kỹ thuật, thay thế)*

ví dụ :

```
<html>
<head>
<script type="text/javascript">
function open_win()
{
window.open("http://www.w3schools.com")
}
</script>
</head>
<body>
<input type=button value="Open Window" onclick="open_win()" />
</body>
</html>
```

Đôi khi chúng ta còn dùng cú pháp sau để mở một cửa sổ mới:

```
Var [name]=window.open(URL,tên,thông số kỹ thuật,thay thế)
```

Ví dụ dưới đây lại mở một cửa sổ blank mới:

```
<html>
<body>
<script type="text/javascript">
myWindow=window.open('','','width=200,height=100')
myWindow.document.write("<p>This is 'myWindow'</p>")
myWindow.focus()
</script>
</body>
</html>
```

Close()

Đóng cửa sổ.

Hỗ trợ trong tất cả các trình duyệt.

Cú pháp:

Window.close();

Ví dụ :

```
<html>
<head>
<script type="text/javascript">
function openWin()
{
myWindow=window.open("","","width=200,height=100");
myWindow.document.write("<p>This is 'myWindow'</p>");
}
function closeWin()
{
myWindow.close();
}
</script>
</head>
<body>
<input type="button" value="Open 'myWindow'" onclick="openWin()" />
<input type="button" value="Close 'myWindow'" onclick="closeWin()" />
</body>
</html>
```

Chú ý:

Hai sự kiện quan trọng nhất gắn liền với cửa sổ là onLoad và onUnLoad,nghĩa là sự kiện xảy ra khi cửa sổ mở hay lúc đóng.

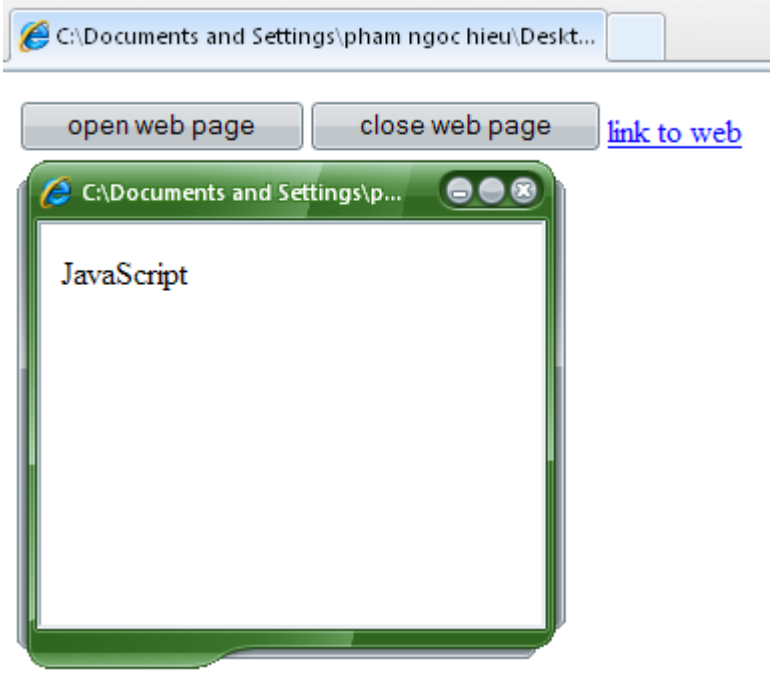
Bài thực hành:

Hãy tạo một trang web trong đó có hai button " open web page" và " close web page" khi nhấn vào hai nút này thì một cửa sổ mới kích cỡ 200 *200(phía trong có ghi chữ "JavaScrip") hiện ra hoặc đóng lại,và một đường link dẫn tới một trang web mới thay cho trang web hiện thời.

Bài làm:

```
<html>
<head>
 <script language="javascript">
 function mo()
 {
  newwin=window.open("","newweb","width=200px,height=200px");
  newwin.document.write("<p>JavaScript<p>");/* phải có tag <p> đó nhé */
 }
```

```
 function dong()
 {
  newwin.window.close();
 }
 </script>
</head>
<body>
<input type="button" value="open web page" onclick="mo();" />
<input type="button" value="close web page" onclick="dong();" />
<a href="http://dantri.vn"> link to web </a>
</body>
</html>
```



Location Object

Đối tượng Location chứa thông tin về URL hiện tại.

Location là một phần của đối tượng window và nó được truy cập thông qua thuộc tính window.location.

Chúng ta có thể tham khảo bảng thuộc tính và phương thức ở bảng tóm tắt sau.

Location Object Properties

Property	Description
hash	Returns the anchor portion of a URL
host	Returns the hostname and port of a URL

hostname	Returns the hostname of a URL
href	Returns the entire URL
pathname	Returns the path name of a URL
port	Returns the port number the server uses for a URL
protocol	Returns the protocol of a URL
search	Returns the query portion of a URL

Location Object Methods

Method	Description
assign()	Loads a new document
reload()	Reloads the current document
replace()	Replaces the current document with a new one

Chú ý rằng tất cả các thuộc tính của đối tượng location khi truy cập đều trả về giá trị là xâu kí tự.

Chúng ta sẽ tìm hiểu từng vấn đề 1:

Hash

Thuộc tính này sẽ trả về achor của URL hiện tại (bao gồm cả #).

Hỗ trợ trong tất cả các trình duyệt.

Cú pháp:

location.hash

ví dụ trang web hiện tại của chúng ta là http://www.example.com/test.htm#part2

vậy thì:

```
<script type="text/javascript">
document.write(location.hash);
</script>
```

Kết quả hiện thị sẽ là :

```
#part2
```

Host

Nó sẽ trả về hostname và cổng của URL.

Hỗ trợ trong tất cả các trình duyệt.

Cú pháp:

Location.host

Ví dụ :

Trang web hiện tại của chúng ta là www.w3schools.com

```
<script type="text/javascript">
document.write(location.host);
</script>
```

Kết quả hiện thị là :

www.w3schools.com

hostname

nó sẽ trả về host name của cửa sổ hiện tại.

hỗ trợ trong tất cả các trình duyệt.

cú pháp:

location.hostname

ví dụ cũng tương tự như ví dụ trên :

```
<script type="text/javascript">
document.write(location.host);
</script>
```

Kết quả hiện thị là :

www.w3schools.com

lời bình: thuộc tính này khá giống với thuộc tính host.

href

nó sẽ trả về URL đầy đủ của trang hiện tại.

hỗ trợ trên tất cả các trình duyệt.

cú pháp:

location.href

ví dụ:

```
<script type="text/javascript">
document.write(location.href);
</script>
```

Kết quả hiện thị là :

http://www.w3schools.com/jsref/prop_loc_href.asp

pathname

nó sẽ trả về một phần của URL hiện tại.

hỗ trợ trong tất cả các trình duyệt.

cú pháp:

location.pathname

ví dụ trang web hiện tại của chúng ta là http://www.w3schools.com/jsref/prop_loc_href.asp

```
<script type="text/javascript">
document.write(location.pathname);
</script>
```

Kết quả hiện thị sẽ là :

/jsref/tryit_view.asp

Port

Nó sẽ trả về số cổng của sever sử dụng cho URL.

Chú ý rằng nếu số cổng là 80 thì sẽ không hiện thị.

Cú pháp:

Location.port

Ví dụ :

```
<script type="text/javascript">
document.write(location.port);
</script>
```

Protocol

Trả về protocol của URL hiện tại.

Hỗ trợ trong tất cả các trình duyệt.

Cú pháp:

Location.protocol

Ví dụ:

```
<script type="text/javascript">
document.write(location.protocol);
</script>
```

Kết quả hiện thị là :

http:

search

trả về chuỗi truy vấn của URL.(bắt đầu từ dấu ?)

hỗ trợ trong tất cả các trình duyệt.

cú pháp:

location.search

ví dụ chúng ta đang ở trang web

http://www.example.com/submit.htm?email=someone@example.com

```
<script type="text/javascript">
document.write(location.search);
</script>
```

Kết quả hiện thị là :

?email=someone@example.com

Assign()

Phương thức này sẽ load một trang web mới.

Hộ trợ trong tất cả các trình duyệt.

Cú pháp:

Location.assign(URL)

ví dụ:

```
<html>
<head>
<script type="text/javascript">
function newDoc()
  {
  window.location.assign("http://www.w3schools.com")
  }
</script>
</head>
<body>
<input type="button" value="Load new document" onclick="newDoc()" />
</body>
</html>
```

Reload()

Phương thức này sẽ có tác dụng là tải lại trang.

Hỗ trợ trong tất cả các trình duyệt.

Cú pháp:

Location.reload()

Ví dụ:

```
<html>
<head>
<script type="text/javascript">
function reloadPage()
  {
  window.location.reload()
  }
</script>
</head>
<body>
```

```
<input type="button" value="Reload page" onclick="reloadPage()" />
</body>
</html>
```

Replace()

Phương thức sẽ thay thế trang hiện tại bằng một trang khác.

Hỗ trợ trong tất cả các trình duyệt.

Cú pháp:

Location.replace(newURL)

Ví dụ:

```
<html>
<head>
<script type="text/javascript">
function replaceDoc()
  {
  window.location.replace("http://www.w3schools.com")
  }
</script>
</head>
<body>
<input type="button" value="Replace document" onclick="replaceDoc()" />
</body>
</html>
```

Frame Object

Frame cũng là môt cách thiết kế web HTML rất hay,và tất nhiên JS cũng làm được nhiều điều với Frame.

Một trang web có thể có một vài frame,các frame có thể cuộn độc lập nhau,và một frame có một URL riêng,frame không có các chương trình xử lý sự kiện.

Các thuộc tính của frame

- Frames : đây là một mảng các frame của trang
- Name : thuộc tính name của thẻ <FRAME>
- Length : số lượng các frame con trong một frame.
- Parent : cửa sổ hay frame chưa nhóm frame hiện thời.
- Self : frame hiện thời.

Nhắc lại về cách tạo frame

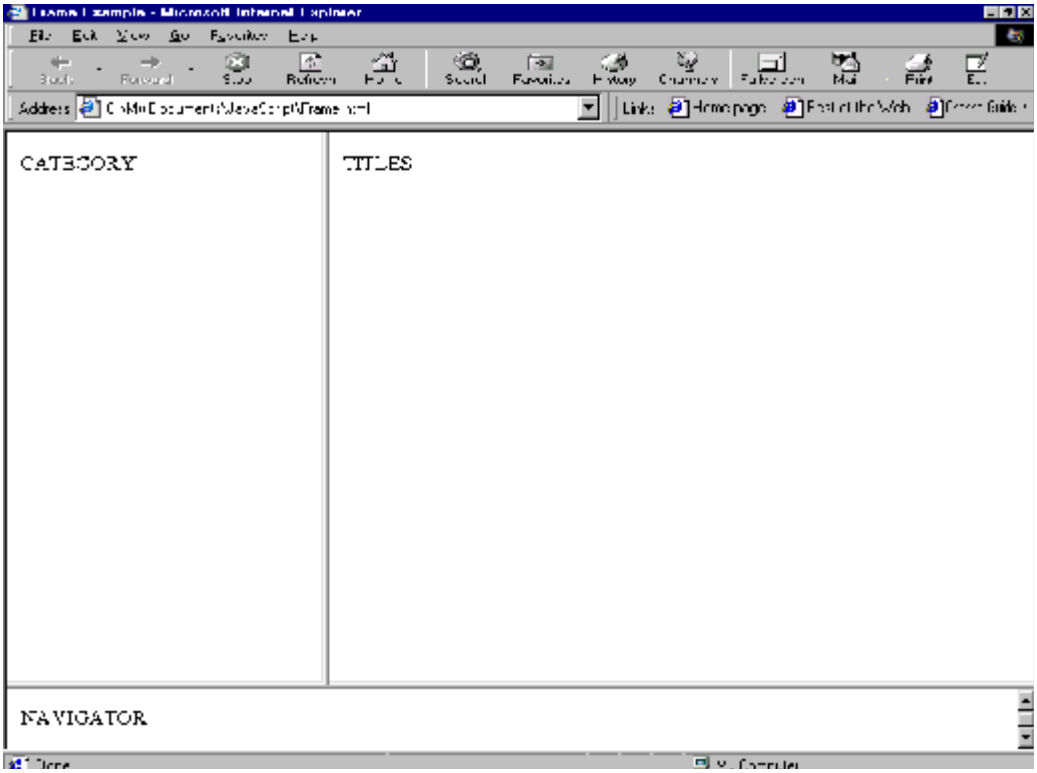
Sau đây chúng ta chỉ nhắc lại cách tạo frame trong HTML mà thôi.

Ta sử dụng thẻ <FRAMESET>

Ta có ví dụ sau:

```
<HTML>
<HEAD>
<TITLE>Frame Example </TITLE>
<FRAMESET ROWS="90%,10%">
<FRAMESET COLS="30%,70%">
<FRAME SRC=CATEGORY.HTM NAME="ListFrame">
<FRAME SRC=TITLES.HTM NAME="contentFrame">
</FRAMESET >
<FRAME SRC=NAVIGATOR.HTM NAME="navigateFrame">
</FRAMESET >
</HEAD>
<BODY> </BODY>
</HTML>
```

Sau đây là hình ảnh của frame được tạo:



Sau đây là sơ đồ hiện thị cấu trúc của các frame:cả 3 frame đều trên cùng một cửa sổ cha,mặc dù 2 trong số các frame đó nằm trong một frame khác.

```
Top
├── listFrame (category.html)
├── contentFrame (titles.html)
└── navigatorFrame (navigator.html)
```

để tương tác với frame bằng JavaScript thì chúng ta sử dụng thuộc tính frames như sau:

listFrame chính là top.frames[0];

contentFrame chính là top.frames[1];

navigatorFrame chính là top.frames[2];

Bây giờ chúng ta cũng sẽ tạo ra một trang web tương tự như trên nhưng bằng cách khác,nếu như ở trong trường hợp trên chúng ta tạo tất cả các frame trên cùng 1 trang thì lần này chúng ta sẽ tạo ra các frame ở các trang khác nhau rồi liên kết chúng lại trong 1 trang khác:

Mức frameset cao nhất được xây dựng như sau:

```
<HTML>
<HEAD>
<TITLE>Frame Example </TITLE>
<FRAMESET ROWS="90%,10%">
<FRAME SRC=muske13.HTML NAME="upperFrame">
<FRAME SRC=NAVIGATOR.HTM NAME="navigateFrame">
</FRAMESET >
</HEAD>
<BODY>
</BODY>
</HTML>
```

Trong file muskel13.HTML chúng ta lại xây dựng một frameset khác (frameset con)

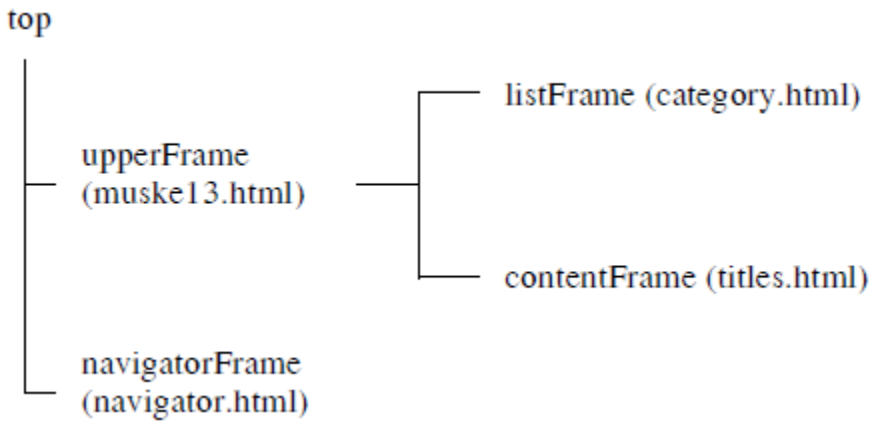
```
<HTML>
<HEAD>
<TITLE>Frame Example </TITLE>
<FRAMESET COLS="30%,70%">
<FRAME SRC=CATEGORY.HTM NAME="ListFrame">
<FRAME SRC=TITLES.HTM NAME="contentFrame">
</FRAMESET >
</HEAD>
<BODY>
</BODY>
</HTML>
```

Chúng ta sẽ được một trang có hiện thị khung như trên,nhưng không phải là chúng giống nhau hoàn toàn,cấu trúc frame trong 2 trang này là khác nhau:

_ một trang thì các frame đều trên cùng 1 trang

_ trang còn lại các frame được xây dựng trên các trang con khác nhau.

Chúng ta có sơ đồ



Chúng ta có thể gọi tới những frame trước đó bằng cách sử dụng thuộc tính frames

upperFrame chính là top.frames[0];

navigatorFrame chính là top.frames[1];

listFrame chính là top.frames[0].frames[0];

contentFrame chính là top.frames[0].frames[1];

lời bình: các kĩ thuật trên cho chúng ta xác định được vị trí của các frame trong trang web,việc xác định vị trí của frame hết sức quan trọng,qua đó chúng ta có thể tương tác được với frame.Và một trong những công việc quan trọng khi chúng ta thao tác với frame là cập nhật một địa chỉ URL vào trong Frame.

Cập nhật nội dung vào frame

Cái hay của frame là trên một trang chúng ta có thể hiện thị nội dung của nhiều trang khác nhau,muốn vậy ta phải tạo ta frame trước rồi sau đó cập nhật nội dung (bằng cách truyền địa chỉ URL) và từng frame.

Vậy làm cách nào để truyền địa chỉ URL vào cho frame ?

Rất đơn giản chúng ta dùng thuộc tính location của frame.

Ví dụ :

```
<INPUT TYPE="button" VALUE="Titles only"
onClick="top.frames[0].frames[0].location='artist.html'">
```

khi nhấn nút thì ngay lập tức nội dung của trang artist.html sẽ được hiện thị trên frame có vị trí là top.frames[0].frames[0].

Thế nhưng nếu bây giờ code như sau:

<INPUT TYPE="button" VALUE="Titles only"
onClick="top.frames[0].location='artist.html'">

khi đó nội dung sẽ được hiện thị ở frame top.frames[0] và hai frame con của nó sẽ bị đóng lại như chúng chưa từng tồn tại bao giờ.

Các thao tác trong frame

Frame cũng là một đối tượng,và đã là đối tượng thì chúng ta sẽ có những thao tác lên nó.những thao tác đó là gì ?

- từ frame này tác động lên frame kia.
- Lấy các thuộc tính của frame.
- Lấy các giá trị của phần tử thuộc frame

Bài thực hành với Frame:

Để làm được điều trên chúng ta phải biết cách tham chiếu đến các phần tử của frame.

Để hiểu rõ hơn về vấn đề này chúng ta đi tìm hiểu ví dụ sau:

Tạo một trang web có 3 khung ,mỗi khung chứa nội dung của một trang khác ở ngoài.

Các trang ngoài đó là : top.html (đặt ở khung trên cùng) ,right.html (đặt ở khung bên phải) và left.html (đặt ở khung bên trái).

Nội dung của top.html là : có các đường link mà khi nhấn vào các đường link đó thì hiện ra thông báo địa chỉ của trang chứa top.html,địa chỉ của trang mẹ chứa top.html,địa chỉ của trang chứa left.html,và hiện ra dòng văn bản hiện trong form text ở frame left.

Nội dung của left.html là : chứa đường link mà khi nhấn vào đó nội dung của hai frame top và right đổi cho nhau.

Nội dung của trang right là : chứa đường link mà khi nhấn vào đó sẽ làm đổi màu nền của frame top.

Bài làm :

Code trang chủ:

```
<html>
<head>
 <title>
 th?c hành Js
 </title>
 </head>
<frameset rows="50%,*" style="border:#99CC00">
<frame scrolling="auto" src="top.html" name="top">
<frameset cols="50%,*">
  <frame scrolling="auto" src="left.html" name="left">
  <frame scrolling="auto" src="right.html" name="right">
```

```
</frameset>
</frameset><noframes></noframes>
</html>
```

Code trang top.html

```
<html>
<head>
<title>
file dinh
</title>
<script language="javascript">
function ham1()
 {
   alert(self.location);
 }
function ham2()
{
 alert(parent.location);
}
function ham3()
{
 alert(parent.frames[1].location);
}
function ham4()
{
  alert(parent.frames[1].document.fom.tex.value);
}
</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"><style
 type="text/css">
</style></head>
<body>
 <a href="#" onclick="ham1();"> click vào đây thì sẽ hiện ra địa chỉ của trang
 top.html</a>   <br /> <br />
 <a href="#" onClick="ham2();"> click vào đây thì sẽ hiện ra địa chỉ trang mẹ chứa
 top.html </a> <br> <br>
 <a href="#" onClick="ham3();"> địa chỉ trang left.html </a> <br> <br>
 <a href="#" onClick="ham4();"> hiện thị văn bản trong hộp text trong left.html
 </a>
</body>
</html>
```

Code trang left

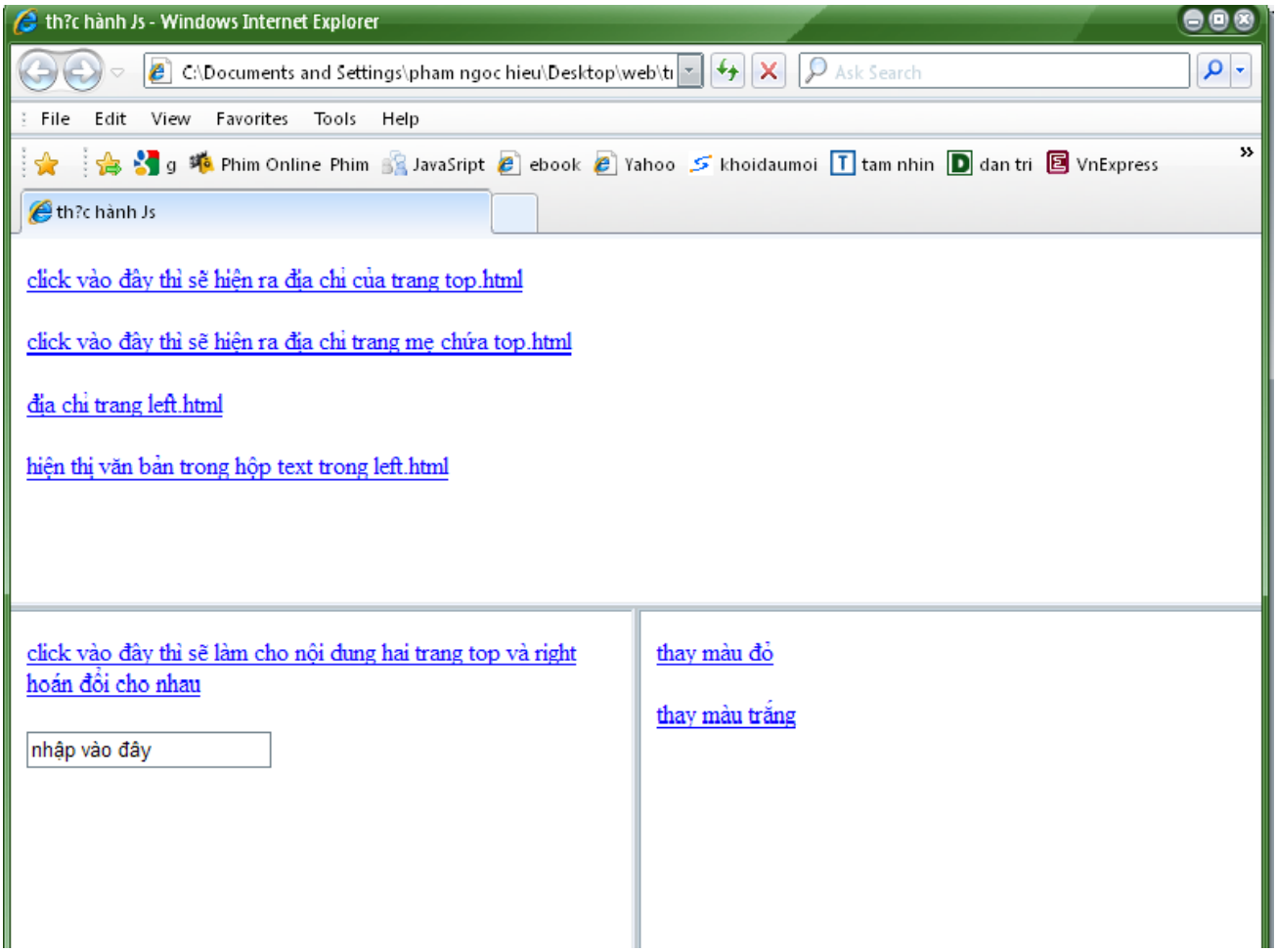
```
<html>
<head>
<title>
left
</title>
<script language="javascript">
function ham5()
{
  var x;
  x=parent.frames[0].location
  parent.frames[0].location=parent.frames[2].location;
  parent.frames[2].location=x;
}
</script>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"><style
 type="text/css">
</style></head>
<body>
<a href="#" onclick="ham5();"> click vào đây thì sẽ làm cho nội dung hai trang top
 và right hoán đổi cho nhau </a> <br> <br>
<form name="fom" method="post">
<input type="text" name="tex" value="nhập vào đây">
</form>
</body>
</html>
```

Code trang right

```
<html>
<head>
<title>
right
</title>
<script language="javascript">
function ham1()
{
  parent.frames[0].document.bgColor="red";
}
function ham2()
{
  parent.frames[0].document.bgColor="white";
}
</script>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8"><style
 type="text/css">
</style></head>
<body>
<a href="#" onclick="ham1();"> thay màu đỏ </a> <br /> <br />
<a href="#" onclick="ham2()"> thay màu trắng </a>
</body>
</html>
```

Dao diện của trang web như sau:



Document Object

Mỗi một tài liệu HTML khi tải vào trình duyệt sẽ trở thành một đối tượng document.

Đối tượng document được tạo ra bởi cặp thẻ <body> </body>.

Đối tượng document là một phần của đối tượng window và có thể được truy cập thông qua thuộc tính window.document.

Collection	Description	W3C
anchors[]	Trả về một mảng các liên kết trong trang	Yes
forms[]	Trả về một mảng các form trong trang	Yes
images[]	Trả về một mảng các ảnh trong trang	Yes
links[]	Trả về một mảng các links trong trang	Yes

Document Object Properties

Property	Description	W3C
cookie	Trả về tất cả tên của cookie trong trang	Yes
documentMode	Returns the mode used by the browser to render the document	No
domain	Returns the domain name of the server that loaded the document	Yes
lastModified	Returns the date and time the document was last modified	No
readyState	Returns the (loading) status of the document	No
referrer	Returns the URL of the document that loaded the current document	Yes
title	Đặt hoặc trả về giá trị title cua trang	Yes
URL	Trả về tên đầy đủ của URL của trang	Yes

Document Object Methods

Method	Description	W3C
close()	Closes the output stream previously opened with document.open()	Yes
getElementById()	Accesses the first element with the specified id	Yes
getElementsByName()	Accesses all elements with a specified name	Yes
getElementsByTagName()	Accesses all elements with a specified tagname	Yes
open()	Opens an output stream to collect the output from document.write() or document.writeln()	Yes
write()	Writes HTML expressions or JavaScript code to a document	Yes
writeln()	Same as write(), but adds a newline character after each statement	Yes

Chúng ta lần lượt nghiên cứu một vài phương thức và thuộc tính:

Anchors

Cú pháp:

`document.anchors[].property`

hỗ trợ trong tất cả các trình duyệt.

ví dụ:

```
<html>
<body>
<a name="html">HTML Tutorial</a><br />
<a name="css">CSS Tutorial</a><br />
<a name="xml">XML Tutorial</a><br />
<a href="/js/">JavaScript Tutorial</a>
<p>Number of anchors:
<script type="text/javascript">
document.write(document.anchors.length);
</script></p>
</body>
</html>
```

Kết quả :

HTML Tutorial

CSS Tutorial

XML Tutorial

JavaScript Tutorial

Number of anchors: 3

```
<html>
<body>
<a name="html">HTML Tutorial</a><br />
<a name="css">CSS Tutorial</a><br />
<a name="xml">XML Tutorial</a>
<p>innerHTML of the first anchor:
<script type="text/javascript">
document.write(document.anchors[0].innerHTML);
</script>
</p>

</body>
</html>
```

Kết quả hiện thị:

```
InnerHTML of first anchor: HTML Tutorial
```

Forms

Cú pháp:

`document.forms[].property`

hỗ trợ trong tất cả các trình duyệt.

ví dụ:

```
<html>
<body>
<form name="Form1"></form>
<form name="Form2"></form>
<form></form>
<p>Number of forms:
<script type="text/javascript">
document.write(document.forms.length);
</script></p>
</body>
</html>
```

Kết quả hiện thị như sau:

Number of forms: 3

Ví dụ:

```
<html>
<body>
<form name="Form1"></form>
<form name="Form2"></form>
<form></form>
<p>Name of first form:
<script type="text/javascript">
document.write(document.forms[0].name);
</script></p>
</body>
</html>
```

Kết quả :

Name of first form: Form1

Images

Đối tượng này chúng ta sẽ nghiên cứu kĩ trong phần sau.

Cú pháp: `document.images[].property`

Hỗ trợ trong tất cả các trình duyệt.

Ví dụ:

```
<html>
<body>
<img border="0" src="klematis.jpg" width="150" height="113" />
<img border="0" src="klematis2.jpg" width="152" height="128" />
<p>Number of images:
<script type="text/javascript">
document.write(document.images.length);
</script></p>
</body>
</html>
```

Kết quả:

Number of images: 2

Ví dụ:

```
<html>
<body>

<img id="klematis lilac" border="0" src="klematis.jpg" width="150" height="113" />
<img id="klematis pink" border="0" src="klematis2.jpg" width="152" height="128" />
<p>Id of first image:
<script type="text/javascript">
document.write(document.images[0].id);
</script></p>
</body>
</html>
```

Kết quả :

Id of first image: klematis lilac

Links

Cú pháp: `document.links[].property`

Hỗ trợ trong tất cả các trình duyệt.

```
<html>
<body>
<img src ="planets.gif" width="145" height="126" alt="Planets" usemap
 ="#planetmap" />
<map name="planetmap">
<area shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun" />
<area shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury" />
<area shape="circle" coords="124,58,8" href="venus.htm" alt="Venus" />
</map>
<p><a href="/js/">JavaScript Tutorial</a></p>
```

```
<p>Number of areas/links:
<script type="text/javascript">
document.write(document.links.length);
</script></p>
</body>
</html>
```

Kết quả hiện thị :

Number of areas/links: 4

Ví dụ:

```
<html>
<body>
<img src ="planets.gif" width="145" height="126" alt="Planets" usemap
 ="#planetmap" />
<map name="planetmap">
<area id="sun" shape="rect" coords="0,0,82,126" href="sun.htm" alt="Sun" />
<area id="mercury" shape="circle" coords="90,58,3" href="mercur.htm" alt="Mercury"
 />
<area id="venus" shape="circle" coords="124,58,8" href="venus.htm" alt="Venus" />
</map>
<p><a id="javascript" href="/js/">JavaScript Tutorial</a></p>
<p>Id of first area/link:
<script type="text/javascript">
document.write(document.links[0].id);
</script></p>
</body>
</html>
```

Kết quả :

Id of first area/link: sun

Image Object

Bài học này chúng ta sẽ tìm hiểu về đối tượng image (ảnh) trong JavaScript.

Khi học về HTML chúng ta đã có cách chèn ảnh vào trang web là dùng thẻ tag <img>.

Nhưng với JavaScript chúng ta cũng có thể chèn được bằng cách tạo ra đối tượng image bằng cú pháp:

var [tên_biến] = new Image();

khi khai báo như thế này nghĩa là chúng ta đã tạo ra một đối tượng là ảnh rồi,mặc dù chưa cho các thông số của bức ảnh đó như src, độ rộng,độ dài…

lời bình:

qua đây chúng ta biết thêm được rằng có 2 cách để chèn một ảnh vào trang web,cách thứ nhất là dùng tag <img> của HTML và cách thứ hai là tạo ra một đối tượng Image bằng JavaScript bằng cú pháp new Image();

khi chúng ta học về HTML chúng ta đã biết sử dụng về các thông số của thẻ <img> các thông số đó cũng chính là thuộc tính của đối tượng Image luôn.

Cụ thể như bảng sau:

Image Object Properties

Property	Description
align	Thuộc tính về căn lề cho ảnh
alt	Thuộc tính này trả về văn bản hiện lên khi chuột rê qua ảnh
border	Thuộc tính về độ dài đường viền ảnh
complete	Trả về giá trị khi trình duyệt load ảnh thành công
height	Thuộc tính chiều cao của một ảnh
hspace	Thuộc tính khoảng cách từ ảnh đến các lề bên trái hoặc phải
longDesc	Sets or returns the value of the longdesc attribute of an image
lowsrc	Sets or returns a URL to a low-resolution version of an image
name	Thuộc tính tên của một ảnh
src	Địa chỉ của ảnh hiện thị
useMap	Sets or returns the value of the usemap attribute of an image
vspace	Thuộc tính khoảng cách từ ảnh đến các lề trên hoặc dưới
width	Thuộc tính chiều rộng của ảnh

Image Object Events

Event	The event occurs when...
onabort	Load một hình ảnh bị gián đoạn
onerror	Sự kiện khi load một ảnh bị lỗi
onload	Sự kiện khi load xong một ảnh

Chúng ta có thể tìm hiểu kĩ hơn các thuộc tính và phương thức này của đối tượng image tại địa chỉ http://www.w3schools.com/jsref/dom_obj_image.asp có đầy đủ hết.

Bài thực hành với image:

Hãy tạo một button bằng javaScript và khi di chuột qua button đó thì màu của button sẽ thay đổi. Gợi ý là hãy tạo ảnh của một button với hai màu khác nhau,rồi dùng mã JS mà thay đổi hai ảnh ấy cho nhau sẽ tạo thành một button thay đổi màu.

Bài làm :

```
<html>
<head>
<title>
th?c hành JS
</title>
<script language="javascript">
function ham1()
{
 document.anh.src="button/anh2.GIF";
}
function ham2()
{
 document.anh.src="button/anh1.GIF";
}
</script>
</head>
<body>
<a href="#" onMouseOver="ham1();" onMouseOut="ham2();"> <img src="button/anh1.GIF"
 name="anh" width="200" height="40" border="0">
</a>
</body>
</html>
```

Như vậy là chúng ta đã có được 1 button với hiệu ứng chuyển màu đẹp mắt rồi.Nhưng bài tập này chỉ với 1 nút thôi bây giờ chúng thử làm 1 dãy button giống như menu của website vậy xem sao ?

Dao diện tham khảo:

Các bạn hãy nhìn vào dao diện sau khi chuột di qua các button thì màu tím của button sẽ bị thay thế bằng các màu khác. Sau đây là code nguồn:

```
<html>
<head>
<title> JS </title>
<script language="javascript">
function ham1(x,y)
{
 document[x].src=y;
}
</script>
</head>
<body>
<div style="width:323px;
position:absolute; left: 4px;
top: 1px;">
<img src="button/30160757_google-
ldilbert1-0107.jpg"
 border="0"></div>
<div style="width:600;" align="left">
<div style="width:200; position:absolute; left: 3px; top: 158px;" align="left">
<a href="http://dantri.com.vn/" onMouseOver="ham1('hieu9','button/hieu8.gif');"
 onMouseOut="ham1('hieu9','button/hieu9.gif');"><img src="button/hieu9.gif"
 name="hieu9" width="150" height="50"> </a></div>
<div style="width:200; position:absolute; left: 3px; top: 389px;" align="left">
<a href="http://www.vnexpress.net/GL/Home/"
 onMouseOver="ham1('hieu','button/hieu1.gif');"
 onMouseOut="ham1('hieu','button/hieu.gif');" ><img src="button/hieu.gif"
 name="hieu" width="150" height="50"> </a></div>
<div style="width:200; position:absolute; left: 2px; top: 332px;" align="left"><a
 href="http://vietnamnet.vn/" onMouseOver="ham1('hieu2','button/hieu3.gif');"
 onMouseOut="ham1('hieu2','button/hieu2.gif');" ><img src="button/hieu2.gif"
 name="hieu2" width="150" height="50"></a></div>
<div style="width:200; position:absolute; left: 2px; top: 274px;" align="left">
<a href="http://www.w3schools.com/"
 onMouseOver="ham1('hieu5','button/hieu4.gif');"
```

```
 onMouseOut="ham1('hieu5','button/hieu5.gif');"><img src="button/hieu5.gif"
 name="hieu5" width="150" height="50"> </a></div>
<div style="width:200; position:absolute; left: 2px; top: 215px;" align="left">
<a href="http://www.khoidaumoi.com/"
 onMouseOver="ham1('hieu6','button/hieu7.gif');"
 onMouseOut="ham1('hieu6','button/hieu6.gif');"><img src="button/hieu6.gif"
 name="hieu6" width="150" height="50"></a></div>
</div>
</body>
</html>
```

Đừng có hốt hoảng khi nhìn thấy đoạn code này nhé.

Kỹ thuật tạo button bằng JavaScript là rất quan trọng,bởi vì tạo trang web thì không thể không có các button,đôi khi button chúng ta tạo ra phải thật đẹp mắt,nhiều hiệu ứng.Phải thực hành nhiều về vấn đề này.

Bài thực hành tiếp theo như sau:

Hãy tạo một trang web có 2 bức ảnh (do bạn chọn) dùng JS để tạo hiệu ứng có hiệu ứng là khi rê chuột vào một bức ảnh nào đó thì một hộp thoại hiện ra nói về bức ảnh đó,và chuột ra khỏi ảnh thì hộp thoại đó biến mất.

Gợi ý các bước có thể làm như sau :

_ dùng CorelDraw để vẽ các hộp thông báo theo ý muốn

_ chèn các ảnh vào trang web

_ sử dụng JS để tạo hiệu ứng cho trang web của bạn.

Dao diện tham khảo:

Code nguồn như sau :

```
<html>
<head>
<title>
JS
</title>
<script language="javascript">
function ham1()
{
 document.getElementById("hieu").width=200;
 document.getElementById("hieu").height=100;
}
function ham2()
{
 document.getElementById("hieu").width=0;
 document.getElementById("hieu").height=0;
}
function ham3()
{
 document["thuy"].width=200;
 document["thuy"].height=100;
}
function ham4()
{
 document["thuy"].width=0;
 document["thuy"].height=0;
}
</script>
<style>
b h2{background-color:#009966}
</style>
</head>
<body>
```

```
<b><h2> thuc hanh JavaScript</h2></b><br>
<div style="width:100px; position:absolute; left: 26px; top: 159px;"><a href="#">
<img name="kim1" src="4a335826_69953354_b5a666a3f6dbf588cbefd00.jpg" width="100px"
 height="100px" border="3" style="border-color:#006666" onMouseOver="ham1();"
 onMouseOut="ham2();"></a></div>
<div style="width:100px; position:absolute; left: 25px; top: 276px;"><a href="#">
<img name="kim2" src="4b11291d_32c6ef2b_1ow3-1-.jpg" width="100px" height="100px"
 border="3" style="border-color:#006666" onMouseOver="ham3();"
 onMouseOut="ham4();"></a></div>
<div style="width:200px; position:absolute; left: 135px; top: 150px;">
<img name="anh1" src="Graphic2.gif" width="0 px" height="0 px" border="0"
 id="hieu"></div>
<div style="width:200px; position:absolute; left: 134px; top: 269px;">
<img name="anh2" src="anh.gif" width="0px" height="0px" border="0"
 id="thuy"></div>
</body>
</html>
```

ở đoạn code này chúng ta đã dùng tới 4 hàm khá giống nhau,để tiết kiệm hơn chúng ta có thể chỉ cần xây dựng hai hàm mà thôi

```
<script language="javascript">
function ham1(name)
{
 document.getElementById(name).width=200;
 document.getElementById(name).height=100;
}
function ham2(name)
{
 document.getElementById(name).width=0;
 document.getElementById(name).height=0;
}
</script>
<body>
<b><h2> thuc hanh JavaScript</h2></b><br>
<div style="width:100px; position:absolute; left: 26px; top: 159px;"><a href="#">
<img name="kim1" src="4a335826_69953354_b5a666a3f6dbf588cbefd00.jpg" width="100px"
 height="100px" border="3" style="border-color:#006666"
 onMouseOver="ham1('hieu');" onMouseOut="ham2('hieu');"></a></div>
<div style="width:100px; position:absolute; left: 25px; top: 276px;"><a href="#">
<img name="kim2" src="4b11291d_32c6ef2b_1ow3-1-.jpg" width="100px" height="100px"
 border="3" style="border-color:#006666" onMouseOver="ham1('thuy');"
 onMouseOut="ham2('thuy');"></a></div>
```

```
<div style="width:200px; position:absolute; left: 135px; top: 150px;">
<img name="anh1" src="Graphic2.gif" width="0 px" height="0 px" border="0"
 id="hieu"></div>
<div style="width:200px; position:absolute; left: 134px; top: 269px;">
<img name="anh2" src="anh.gif" width="0px" height="0px" border="0"
 id="thuy"></div>
</body>
```

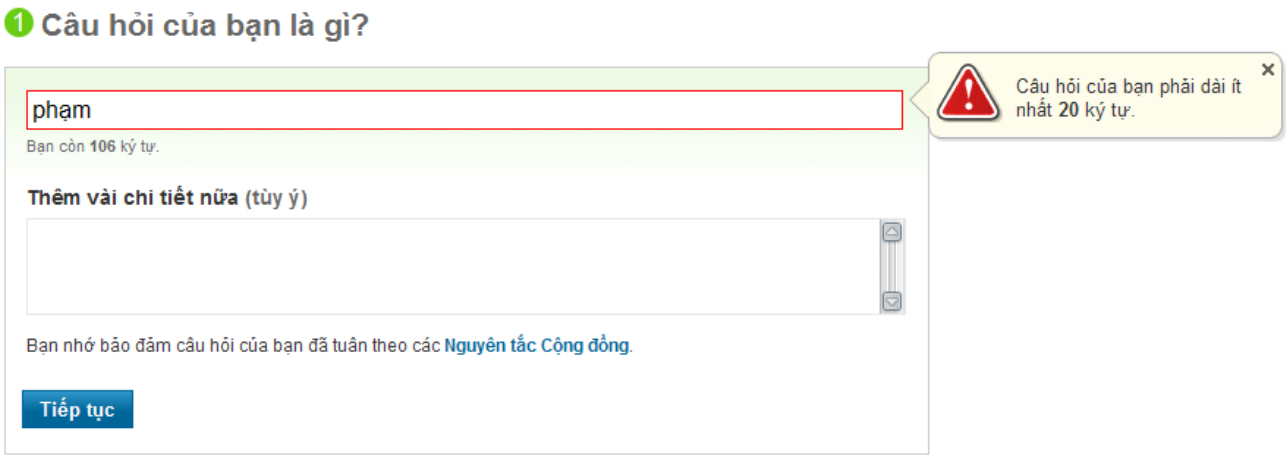
Kết quả cũng tương tự.

Hoặc chúng ta có thể dùng cách truy cập sau:

```
<script language="javascript">
function ham1(name)
{
 document[name].width=200;
 document[name].height=100;
}
function ham2(name)
{
 document[name].width=0;
 document[name].height=0;
}
</script>
```

Bài thực hành tạo dao diện Yahoo Hỏi đáp:

Chúng ta thử đăng nhập vào trang yahoo hỏi đáp và thử tạo ra một trang dao diện như sau:



Ô nhập liệu trên khi bạn nhập vào ít hơn 20 kí tự và nhấn vào nút " tiếp tục" thì ngay lập tức một cảnh báo hiện ra như trên,và nhập tối đa là 110 kí tự(bên dưới ô nhập liệu sẽ luôn hiện ra thông báo số kí tự còn lại).

Code tham khảo như sau:

```
<html ">
```

```
<head>
<title>Untitled Document</title>
<style>
 h5{color:#999999}
.style3 {font-size: 18px}

</style>
<script language="javascript">
function ham1()
{
document.getElementById("modau").value="";
}
function ham2()
{
document.getElementById("note").width=0;
document.getElementById("note").height=0;
}
function ham3()
{
document.getElementById("note").width=250;
document.getElementById("note").height=73;
}
function ham5()
{
if(document.getElementById("modau").value.length>110)
alert("câu hỏi của bạn quá dài");
else
{
if(document.getElementById("modau").value.length<20)
 ham3();
else
{
 window.open("http://dantri.com.vn");
 document.getElementById("note").width=0;
 document.getElementById("note").height=0;
}
}
}
function ham6()
{
```

```
document.getElementById("2").innerHTML=110-
 document.getElementById("modau").value.length;
}
</script>
</head>
<body>
<div style="width:383px; position:absolute; height: 40px;">
<img src="tuade.gif" width="371" height="39" /></div><br /><br />
<div style="width:700px; position:absolute; left: 9px; top: 85px;">
<img src="nen.gif" width="649" height="311" /></div>
<div style="width:625px; position:absolute; left: 18px; top: 95px; height: 38px;">
<input id="modau" type="text" style="width:608px; position:absolute; left: 7px;
 top: 7px; height: 28px;" value='bạn muốn hỏi gì ?' onfocus="ham1();"
 onkeyup="ham6();" />
</div><br />
<div id="1" style="width:190px; position:absolute; left: 26px; top: 121px; height:
 27px;">
<h5>Bạn còn <b id="2">110 </b>kí tự</h5>
</div><br /><br />
<div style="width:auto; position:absolute; left: 24px; top: 177px;">
<b><span class="style3">Thêm vài chi tiết nữa</span></b></div> <br />
<div style="width:auto; position:absolute">
<textarea style="position:absolute; left: 16px; top: 87px; width: 613px; height:
 80px;"> </textarea>
</div><br />
<div style="width:auto; position:absolute; left: 24px; top: 307px;">
Bạn nhớ đảm bảo câu hỏi của bạn đã tuân theo các <a href="#"><b><font
 color="#3366CC">Nguyên tắc cộng đồng</font></b></a></div><br /><br />
<div style="width:auto; position:absolute">
<div style="width:auto; position:absolute; height: 66px; left: 653px; top: -
 97px;">
<img src="canhbao.gif" width="0" height="0" id="note" /></div>
<a href="#" onclick="ham5();">
<img src="nut.jpg" width="107" height="35" style="position:absolute; left: 15px;
 top: 171px; width: 99px; height: 29px;" border="0" /></div>
</a >
</body>
</html>
```

Bình luận :

Qua những bài thực hành này chúng ta hiểu rõ hơn về đối tượng image,các kĩ thuật can thiệp vào đối tượng image,chú ý tới 3 cách như sau:

Document.getElementById(“ id “).property

Document[“ name”].property

Document.images[ ].property

Trên đây là các cách tương tác với đối tượng Image.

Chúng ta cũng biết rõ hơn về cách tạo ra các button hình ảnh:

Với các button bình thường chúng ta có thể tạo button ảnh bằng cách sử dụng cú pháp:

<button > <img src=”URL” > </button>

Nhưng nếu như button lại lại là nút submit của form thì phải dùng cú pháp sau:

<input type=”image” src=”URL”>

Lúc này ảnh có tác dụng như một nút submit của form.

Kiểm tra tính hợp lệ của giá trị trong form với JavaScript

Khi xây dựng một web site có thu thập các thông tin từ người sử dụng thông qua các form nhập liệu, bạn luôn cần chú ý đến việc kiểm tra các thông tin đó để xem dạng thức của chúng có phù hợp với yêu cầu của bạn không. Giả sử, bạn cung cấp một form nhập số điện thoại nhưng người sử dụng của bạn vô tình hay cố tình nhập vào đó các chữ cái. Rõ ràng là việc kiểm tra như vậy là cần thiết để sàng lọc tất cả các thông tin không hợp lệ ít nhất là về mặt hình thức trước khi đưa nó về bộ phận lưu trữ và quản lý dữ liệu. Bài viết này gợi ý cho bạn một trong những cách thức dùng Javascript để xác thực tính hợp lệ của các thông tin đó trên máy khách.

Giả sử chúng ta có một trang như sau:

```
<html>
<head>
<basefont face=”Arial”>
<script language=”JavaScript”>
function validateForm()
{
// kiểm tra tên
...
```

```
// kiểm tra chiều dài mật khẩu
...
// kiểm tra địa chỉ email
...
// kiểm tra số tuổi
...
// kiểm độ tuổi
...
return true;
}
</script>
</head>
<body>
<!-- Phần mã HTML cho form
<form action="someform.php" method="post" onSubmit="return validateForm();">
Tên họ: <br> <input type="text" name="name" id="name"><p>
Mật khẩu:<br> <input type="password" name="password" id="password"><p>
Địa chỉ email: <br> <input type="text" name="email" id="email"><p>
Tuổi: <br><input type="text" name="age" id="age"><p>
<input type="submit" value="Gửi thông tin đi"></form>
</body>
</html>
```

Bạn có thể thấy đây chỉ là một form đơn giản gồm các trường thu nhận dữ liệu từ người sử dụng: tên, mật khẩu, địa chỉ email, tuổi và cuối cùng là một nút nhấn. Trước khi form được gửi đi thì sự kiện "onSubmit" sẽ gọi hàm validateForm(), hàm này sẽ kiểm tra các thông tin sau:

• Tên trong form đã được điền chưa

• Chiều dài mật khẩu có từ 6 kí tự trở lên không

• Địa chỉ email address có định dạng đúng không

• Giá trị của tuổi có phải là con số và nằm từ 1 đến 99 không.

Nếu có một giá trị không phù hợp qua đợt kiểm tra thì hàm validateForm() sẽ in ra một thông báo lỗi và trả lại giá trị false, nghĩa là người sử dụng sẽ không gửi thông tin từ form đi được. Nếu các đợt kiểm tra không gặp lỗi thì hàm sẽ trả lại giá trị true và form được phép gửi giá trị đi. Như vậy sự kiểm tra trên máy khách sẽ giúp giảm thiểu các thông tin không đạt tiêu chuẩn nằm trong cơ sở dữ

liệu.

Đây là một tác vụ rất bình thường trên mạng Internet và dựa trên cách tư duy lập trình hướng đối tượng, chúng ta cần xây dựng nó thành một đoạn mã cho phép sử dụng lại: bao gói các tác vụ kiểm tra trong một đối tượng và mỗi khi cần dùng cho một ứng dụng web khác, chúng ta sẽ tạo ra các mẫu của đối tượng đó.

Bắt đầu

Trở lại với vấn đề, theo đoạn mã lúc đầu thì chúng ta đã định nghĩa đối tượng formValidator gồm có 2 bộ phận:

1. Một chuỗi các phương thức nhận dữ liệu cần được kiểm tra như là các đối số phương thức, để kiểm tra các dữ liệu này xem hợp lệ hay không. Cái tiêu chuẩn để đưa ra đánh giá là "hợp lệ" có thể được định nghĩa bên trong phạm vi của phương thức, và trả lại một giá trị thích hợp.
2. Một cấu trúc dữ liệu JavaScript (ở đây là một mảng) dùng để lưu trữ một danh sách các lỗi phát hiện được trong suốt quá trình kiểm tra tính hợp lệ và các phương thức để thao tác trên cấu trúc dữ liệu này. Như bạn sẽ thấy, ở đây có 2 thành phần cơ bản làm cho việc xây dựng một đối tượng Validator, một đối tượng với nhiều phương thức tổng quát, trở nên rất dễ dàng.

Ta cần phải xác định cách thức làm việc của lớp này. Đây là cách của tôi: // khởi tạo đối tượng

```
fv = new formValidator();
// tiến hành kiểm tra
// kiểm tra trường dữ liệu rỗng
if (fv.isEmpty(document.forms[0].elements[0].value))
{
fv.raiseError("Vui lòng nhập vào giá trị");
}
// kiểm tra phạm vi dữ liệu
if (!fv.isWithinRange(document.forms[0].elements[1].value, 1, 99))
{
fv.raiseError("Nhập giá trị trong khoảng từ 1-99");
}
// nếu có lỗi thì hiển thị, ngược lại thì tiếp tục
if (fv.numErrors() > 0)
{
```

fv.displayErrors();

}

Như bạn thấy, một khi đối tượng được khởi tạo, có nhiều phương thức được gọi (với các giá trị nhập tương ứng trong form như là tham số) để kiểm tra xem dữ liệu nhập có hợp lệ không. Nếu không, thông báo lỗi sẽ được tạo ra và lưu vào ngăn xếp lỗi của đối tượng; những thông báo này có thể hiển thị sau đó, khi việc kiểm tra tất cả các trường dữ liệu hoàn tất. Sau khi đã xác định rõ các chức năng cơ bản của đối tượng, chúng ta nên dành thời gian liệt kê các phương thức quan trọng cùng với chức năng của nó:

• isEmpty() – kiểm tra một biến trong form có rỗng hay không;
• isNumber() – kiểm tra một biến trong form có phải là số hay không;
• isAlphabetic() – kiểm tra biến trong form có phải chỉ toàn chữ hay không;
• isAlphaNumeric() – kiểm tra một biến trong form có phải chỉ chứa toàn chữ và số hay không;
• isWithinRange() – kiểm tra một biến trong form có nằm trong một khoảng giá trị số định sẵn hay không;
• isEmailAddress() – kiểm tra một biến trong form có chứa một địa chỉ email hợp lệ hay không;
• raiseError() – chèn một thông báo lỗi vào ngăn xếp lỗi;
• displayErrors() – hiển thị danh sách các thông báo lỗi trong hộp thoại cảnh báo (alert box);
• numErrors() – trả về số lượng các thông báo lỗi đã tạo ra.

Đây là những phương thức chủ yếu; có thể có thêm các phương thức khác mà tôi sẽ thêm vào trong quá trình phát triển. Hãy bắt đầu bằng cách thiết lập định nghĩa cho lớp:

// tạo một đối tượng

function formValidator()

{

// phần còn lại

}

Do tôi dự định sẽ xây dựng một số phương thức quản lý lỗi cơ bản cho lớp này, tôi cần phải thêm vào một biến để giữ thông tin này.

// tạo đối tượng

function formValidator()

{

```
// tạo mảng lưu các thông báo lỗi
this.errorList = new Array;
// phần còn lại
}
```

Bây giờ chúng ta lần lượt xây dựng các phương thức kiểm tra.
Đây là phương thức đầu tiên, thực hiện một phép kiểm tra rất cơ bản, kiểm tra xem một biến có chứa giá trị nào nữa hay không.

```
// kiểm tra xem dữ liệu nhập có phải là toàn khoảng trắng hoặc rỗng hay không.
function isEmpty(val)
{
if (val.match(/^s+$/) || val == “”)
{return true;}
else
{return false;}
}
```

Đây là một phương thức khá đơn giản, nhưng đáng để xem xét một cách chi tiết, bởi vì tất cả các phương thức tiếp theo đều có cấu trúc tương tự. Phương thức isEmpty() được gọi với một đối số duy nhất: giá trị sẽ được kiểm tra. Sau đó nó sẽ sử dụng phương thức match() của đối tượng String của JavaScript để kiểm tra giá trị này so với một biểu thức chính quy. Trong ví dụ cụ thể này, biểu thức chính quy được thiết kế để so khớp với một chuỗi chỉ chứa các khoảng trắng thôi; những chuỗi những vậy được xem là không chứa một chút dữ liệu thực tế nào. Nếu tìm thấy một chuỗi như vậy, thì hàm này trả lại giá trị true; nếu không (nghĩa là, nếu giá trị tìm thấy thực sự chứa dữ liệu), thì hàm này trả lại false. Như bạn thấy, kiểu cấu trúc cho phương thức đối tượng này làm cho vấn đề trở nên đơn giản khi bạn muốn đưa lời gọi phương thức vào trong một biểu thức điều kiện nằm trong mã kịch bản xử lý form của bạn.
Bây giờ chúng ta chuyển sang một hàm khác:

```
// kiểm tra xem liệu giá trị nhập vào có phải là con số hay không
function isNumber(val)
{
if (isNaN(val))
{return false;}
```

```
else
{return true;}
}
```

Ở đây, hàm isNaN() của JavaScript được sử dụng để kiểm tra liệu giá trị được cung cấp cho nó có phải là một con số hay không.

Một phương thức hữu ích khác, nhất là khi làm việc với các con số, là WithinRange(). Phương thức này cung cấp một giải pháp khá đơn giản để kiểm tra liệu giá trị nhập vào có nằm trong một dãy số nhất định nào đó hay không.

```
// kiểm tra xem liệu giá trị đó có nằm trong vùng min và max hay không
function isWithinRange(val, min, max)
{
if (val >= min && val <= max)
{return true;}
else
{return false;}
}
```

Cuối cùng, bạn có thể thêm vào một phương thức isChecked() để kiểm tra xem liệu một hộp kiểm (checkbox) nào đó có được đánh dấu hay không:

```
// kiểm tra xem liệu giá trị đó có được chọn hay không
function isChecked(obj)
{
if (obj.checked)
{return true;}
else
{return false;}
}
```

Bạn hãy chú ý đến sự khác biệt giữa phương thức này và các phương thức trước nó. Các phương thức đối tượng trước được viết ra để chấp nhận lấy giá trị trên form làm đối số; còn phương thức này lại chấp nhận một đối tượng làm đối số. Đối tượng này biểu diễn một phần tử trong hộp thoại cảnh

báo. Nó cũng có khả năng tạo ra các thủ tục (routines) kiểm tra phức tạp khác thông qua sự hỗ trợ các biểu thức chính quy có sẵn trong JavaScript. Bạn hãy xem phương thức tiếp theo, isAlphabetic(), phương thức này sử dụng một biểu thức chính quy để kiểm tra xem liệu có phải tất cả các kí tự nằm trong chuỗi nhập vào có phải là các chữ cái hay không…

```
// kiểm tra xem dữ liệu nhập vào có phải là chữ cái hay không function isAlphabetic(val)
{
if (val.match(/^[a-zA-Z]+$/))
{return true;}
else
{return false;}
}
```

… hoặc kiểm tra xem chuỗi nhập vào có chứa cả chữ cái nằm lẫn với số hay không.

```
// kiểm tra xem dữ liệu nhập vào có phải là có cả chữ nằm lẫn với số //(alphanumeric)
function isAlphaNumeric(val)
{
if (val.match(/^[a-zA-Z0-9]+$/))
{return true;}
else
{return false;}
}
```

Khả năng sử dụng các biểu thức chính quy để thực hiện việc kiểm tra dữ liệu sẽ trở nên rất có ích khi mà bạn muốn kiểm tra địa chỉ email mà người sử dụng cung cấp. Chúng ta sẽ dùng phương thức isEmailAddress() để làm việc đó:

```
// kiểm tra xem liệu thông tin nhập vào có đúng là một email hợp lệ hay không

function isEmailAddress(val)
{
if (val.match(/^([a-zA-Z0-9])+ ([.a-zA-Z0-9_-])* @([a-zA-Z0-9_-])+(.[a-zA-Z0-9_
-]+)+/))
{return true;}
else
```

{return false;}

}

Đây cũng mới chỉ là các thao tác kiểm tra tính hợp lệ ở mức độ cơ bản. Bạn có thể tự định nghĩa riêng cho mình các thủ tục mới vào để phục vụ cho nhu cầu của bạn; theo như mẫu mà tôi đã làm ở trên. Bây giờ chúng ta sẽ bổ sung thêm một số phương thức để đảm trách nhiệm vụ trả về các thông báo lỗi trong trường hợp một giá trị nào đó được coi là không hợp lệ khi kiểm tra.

Đầu tiên là ta đưa vào phương thức raiseError(), phương thức này cho phép bạn đưa thông báo lỗi mà người có thể đọc được vào trong ngăn xếp chứa lỗi. Nếu bạn nhìn vào lược đồ đầu tiên mà tôi dùng để thiết kế cách làm việc của đối tượng thì bạn sẽ thấy phương thức này thường được gọi nếu quá trình kiểm tra tính hợp lệ trả lại kết quả false. Phương thức này nhận một thông báo lỗi làm đối số. Đây chính là thông báo được đưa vào ngăn xếp chứa lỗi; ngăn xếp chứa lỗi này tự nó hoạt động như là một mảng JavaScript thông thường. Bạn hãy xem phương thức tạo đối tượng đã nói đến ở trên để hiểu được sự khởi tạo của nó.

// bổ sung một thông báo lỗi vào danh sách lỗi

function raiseError(msg)

{

this.errorList[this.errorList.length] = msg;

}

Phương thức numErrors() trả lại kích thước của mảng “errorList”; nếu kích thước của mảng lớn hơn 1, thì nó chỉ ra rằng có ít nhất một lỗi đã xuất hiện trong quá trình kiểm tra tính hợp lệ của dữ liệu. Xem mã sau:

// trả về các lỗi có trong danh sách lỗi

function numErrors()

{

return this.errorList.length;

}

Bây giờ, phương thức numErrors() sẽ chỉ cho bạn biết có bao nhiêu lỗi đã xuất hiện; chứ nó không cho bạn biết các thông báo lỗi thực sự. Vì thế, bạn cần phương thức displayErrors(), phương thức này trả lại nội dung của mảng errorList thông qua một loạt các hộp thoại của alert().

// hiển thị tất cả các lỗi

// duyệt lần lượt qua các phần tử trong mảng chứa thông báo

// lỗi và in ra nội dung tương ứng

```
function displayErrors()
{
for (x=0; x< } this.errorList[x]); + ? alert( ? Error : { x++)>= min && val <= max)
{return true;}
else
{return false;}
}
// kiểm tra xem dữ liệu nhập vào có đúng là một địa chỉ email hợp lệ hay không
function isEmailAddress(val)
{
if (val.match(/^([a-zA-Z0-9])+ ([.a-zA-Z0-9_-])*@([a-zA-Z0-9_-])+
(.[a-zA-Z0-9_-]+)+/))
{return true;}
else
{return false;}
}
// kiểm tra xem giá trị trong form có được đánh dấu chọn hay không
function isChecked(obj)
{
if (obj.checked)
{return true;}
else
{return false;}
}
// thêm một lỗi vào danh sách chứa thông báo lỗi
function raiseError(msg)
{
this.errorList[this.errorList.length] = msg;
}
// trả lại số lượng lỗi chứa trong mảng chứa thông báo lỗi
function numErrors()
{
```

```
return this.errorList.length;
}
// kết thúc đối tượng
```

Lúc này, chúng ta đã có đủ các khối mã để có thể thực hiện quá trình kiểm tra tính hợp lệ. Chú ý là đối tượng kiểm tra tính hợp lệ phần trên được lưu trong file “formValidator.js”, file này sẽ được đọc ở phần đầu trang như sau.

```
<html>
<head>
<basefont face=”Arial”>
<script language=”JavaScript” src=”formValidator.js”>
</script>
<script language=”JavaScript”>
// kiểm tra giá trị dữ liệu trên form
function checkForm()
{
// khởi tạo đối tượng
fv = new formValidator();
// thực hiện các bước kiểm tra
// kiểm tra xem trường nhập tên có dữ liệu không
if (fv.isEmpty(document.forms[0].elements[0].value))
{
fv.raiseError(“Bạn chưa nhập tên”);
}
// kiểm tra xem trường nhập tuổi có dữ liệu không
if (fv.isEmpty(document.forms[0].elements[1].value))
{
fv.raiseError(“Bạn chưa nhập tuổi”);
}
// kiểm tra xem dữ liệu về tuổi có thuộc phạm vi mong muốn không
if (!fv.isWithinRange(document.forms[0].elements[1].value,1, 99))
{
fv.raiseError(“Vui lòng nhập tuổi trong khoảng từ 1-99”);
```

```
}
// kiểm tra xem trường nhập địa chỉ email có dữ liệukhông
if (fv.isEmpty(document.forms[0].elements[5].value))
{
fv.raiseError("Bạn chưa nhập địa chỉ e-mail");
}
// kiểm tra xem định dạng của địa chỉ email có hợp lệkhông
if (!fv.isEmpty(document.forms[0].elements[5].value)
&&
!fv.isEmailAddress(document.forms[0].elements[5].value))
{
fv.raiseError("Địa chỉ e-mail không hợp lệ");
}
// kiểm tra xem hộp kiểm có được đánh dấu không
if (!fv.isChecked(document.forms[0].elements[6]))
{
fv.raiseError("Vui lòng xác nhận rằng bạn đồng ý với những điều khoản của chúng tôi");
}
// thực hiện xong các bước
// nếu thấy có lỗi thì hiển thị, còn không thì tiếp tục
if (fv.numErrors() > 0)
{fv.displayErrors();return false;}
else
{return true;}
}
</script>
</head>
<body>
<form action="action.cgi" method="POST"
onSubmit="return checkForm()">
<b>Họ và tên:</b><br>
<input type="text" name="name" size="15"><p>
<b>Tuổi:</b><br>
```

```
<input type="text" name="age" size="2" maxlength="2"><p>
<b>Giới tính:</b><br>
<input type="Radio" name="sex" value="m" checked>Nam
<input type="Radio" name="sex" value="f">Nữ
<p><b>Loại báo đọc hằng ngày:<br>
<select name="stype">
<option value="1">Báo Học Lập Trình
<option value="2">PC Magazine
<option value="3">Java Developers Journal
</select>
<p>
<b>Địa chỉ email:<br>
<input type="text" name="email" size="25"><p>
<input type="Checkbox" name="agree">
Tôi chấp nhận tuân thủ theo các nội quy và điều lệ ghi trên site này
<p>
<input type="Submit" name="submit" value="Lưu lại">
</form>
</body>
</html>
```

Ở đây, khi form được gửi đi, sự kiện "onSubmit" sẽ gọi hàm checkForm(), hàm này sẽ khởi tạo một đối tượng mới của lớp formValidator. Sau khi đối tượng được khởi tạo xong, tất cả các thủ tục kiểm tra tính hợp lệ sẽ sẵn sàng cung cấp hoạt động cho trang chứa mã kịch bản như là các phương thức đối tượng, và bạn có thể sử dụng chúng để kiểm tra nhiều giá trị nhập vào từ form.

Tổng kết

Trong bài viết này, tôi đã giới thiệu cách thức kiểm tra giá trị nhập vào từ người dùng trong một form bằng JavaScript theo phương pháp lập trình hướng đối tượng. Tất nhiên, bạn có thể không cần dùng đến đối tượng để thực hiện việc kiểm tra tính hợp lệ. Tuy nhiên, việc viết mã dựa theo phương pháp lập trình hướng đối tượng sẽ giúp giảm bớt thời gian viết mã bằng việc sử dụng lại thư viện các hàm, bao gộp chúng trong một đối tượng và gắn đối tượng đó vào form khi cần. Hy vọng bài viết này sẽ có ích cho bạn.

JavaScript Cookie

Cookie là những phần dữ liệu nhỏ có cấu trúc được chia sẻ giữa web site và browser của người dùng. cookies được lưu trữ dưới những file dữ liệu nhỏ dạng text (size dưới 4k). Chúng được các site tạo ra để lưu trữ/truy tìm/nhận biết các thông tin về người dùng đã ghé thăm site và những vùng mà họ đi qua trong site.

Những thông tin này có thể bao gồm tên/định danh người dùng, mật khẩu, sở thích, thói quen...Cookie được browser của người dùng chấp nhận lưu trên đĩa cứng của máy mình, ko phải browser nào cũng hỗ trợ cookie. Sau một lần truy cập vào site, những thông tin về người dùng được lưu trữ trong cookie. Ở những lần truy cập sau đến site đó, web site có thể dùng lại những thông tin trong cookie (như thông tin liên quan đến việc đăng nhập vào 1 forum...) mà người ko phải làm lại thao tác đăng nhập hay phải nhập lại các thông tin khác. Vấn đề đặt ra là có nhiều site quản lý việc dùng lại các thông tin lưu trong cookie ko chính xác, kiểm tra ko đầy đủ hoặc mã hoá các thông tin trong cookie còn sơ hở giúp cho hacker khai thác để vượt qua cánh cửa đăng nhập, đoạt quyền điêu khiển site .

Ví dụ về các cookie:

Tên cookie - Lần đầu tiên một khách truy cập đến trang web của bạn, họ phải điền vào tên của người đó. Tên này sau đó được lưu giữ trong một cookie. Tiếp theo thời gian khách truy cập đến tại trang web của bạn, người đó có thể nhận được một thông điệp chào mừng như "Chào mừng John Doe!" Cái tên được lấy từ các cookie lưu trữ.

Mật khẩu cookie - Lần đầu tiên một khách truy cập đến trang web của bạn, họ phải điền vào mật khẩu.mật khẩu sau đó được lưu trữ trong một cookie. Tiếp theo thời gian người truy cập đến tại trang web của bạn, mật khẩu được lấy từ các tập tin cookie.

Ngày cookie - Lần đầu tiên một khách truy cập đến trang web của bạn, ngày tháng hiện hành được lưu trữ trong một cookie. Next time the visitor arrives at your page, he or she could get a message like "Your last visit was on Tuesday August 11, 2005!" Tiếp theo thời gian khách truy cập đến tại trang web của bạn, người đó có thể nhận được một thông báo như "Chuyến thăm lần cuối vào lúc

Thứ Ba ngày 11 tháng 8 năm 2005!" The date is retrieved from the stored cookie Các ngày được lấy từ các cookie lưu trữ.

_ Cookies thường có các thành phần sau :

+ Tên: do người lập trình web site chọn

+ Domain: là tên miền từ server mà cookie được tạo và gửi đi

+ Đường dẫn: thông tin về đường dẫn ở web site mà bạn đang xem

+ Ngày hết hạn: là thời điểm mà cookie hết hiệu lực .

+ Bảo mật: Nếu giá trị này đựơc thiết lập bên trong cookie, thông tin sẽ đựơc mã hoá trong quá trình truyền giữa server và browser.

+ Các giá trị khác: là những dữ liệu đặc trưng được web server lưu trữ để nhận dạng về sau các giá trị này ko chứa các khoảng trắng, dấu chấm, phẩy và bị giới hạn trong khoảng 4k.

Cookie có làm hại máy tính không?

TTO - Cookie là gì? Cookie có hại gì không? Nó có phải là phần mềm gián điệp không? Xin cho biết cách phòng trừ cookie nếu nó có hại!

Cookie được xem là những tin nhắn đơn giản được máy chủ đang quản lý một website, chủ động gửi đến trình duyệt web đang dùng để lướt trang web đó, nhằm mục đích theo dõi các hoạt động của người đang xem website.

Tin nhắn này có định dạng mào đầu (header) HTTP chỉ là một chuỗi văn bản. Chuỗi văn bản này sẽ được đưa vào trong bộ nhớ của trình duyệt web. Trình duyệt web sẽ lần lượt lưu giữ các thông tin cookie lên đĩa cứng, vì thế khi trình duyệt tắt đi và bật trở lại thì các thông tin cookie vẫn có thể được dùng tiếp.

Tin nhắn này có định dạng là một chuỗi văn bản đơn giản, nó sẽ được đưa vào thư mục lưu trữ "Temp" của trình duyệt web và trình duyệt web sẽ tự động lưu trữ mọi thông tin cookie vào đĩa cứng. Các thông tin này sẽ phản ánh thói quen của người dùng khi vào trang web đó, chẳng hạn như chuyên mục nào thích xem nhất… và sau đó sẽ tự động gửi các thông tin thu thập được về cho chủ nhân của trang web.

Khi tắt đi trình duyệt thì cookie vẫn còn lưu trữ trong máy và vẫn sẽ tiếp tục hoạt động ở các lần ghé thăm sau. Dựa vào các thông tin mà cookie gửi về, chủ nhân trang web có thể biết được khách lướt web đang quan tâm về những vấn đề gì để sau đó tung quảng cáo phù hợp để bán sản phẩm.

Tung cookie có rất nhiều cái lợi cho chủ trang web:

- Họ có thể biết được một số thông tin về những người đang truy cập trang web của mình, biết được mức độ thường xuyên truy cập cũng như cả thời gian chi tiết truy cập.

- Họ có thể biết được sự cảm nhận của người dùng khi duyệt web đó. Các cookie có thể giúp cho việc lưu trữ các thông tin cá nhân của bạn, mà những thông tin này giúp cho bạn khi vào trang web đó lần sau sẽ cảm thấy thuận tiện hơn. Điển hình là việc vào các trang web yêu cầu bạn "check" vào những ô đại loại như "Remember me on this computer"… chủ yếu là các trang web e-mail mạng, diễn đàn v.v mà lần truy cập sau bạn sẽ thấy nickname của mình hiện sẵn trong ô username, thậm chí nó còn nhớ luôn password và bạn chỉ cần "Enter" là vào thẳng được tài khoản đã đăng ký. Các trang mua sắm trực tuyến còn "siêu" hơn, nó sẽ nhớ luôn những lần mua sắm trước của bạn để tung quảng cáo ra những sản phẩm mới hơn cái bạn đã mua. Các máy chủ sẽ theo dõi việc mua sắm của bạn, xem bạn chọn các món hàng nào và sẽ lưu các thông tin này vào trong các cookie.

- Họ sẽ dùng cookie để điều chỉnh các quảng cáo của mình. Cookie sẽ cung cấp cho chủ nhân của nó những quảng cáo nào được xem nhiều nhất từ đó đề ra các biện pháp điều chỉnh hoặc thiết kế phù hợp.

Chủ nhân của các cookie không thể dùng nó để phát tán virus, spyware, trojan… được vì nó không thể truy được vào đĩa cứng của bạn. Tuy nhiên nó cũng có thể thu thập được khá nhiều thông tin cá nhân của bạn, nhất là những thông tin mà bạn cung cấp cho trang web, chẳng hạn như thông tin thẻ tín dụng của bạn sẽ chứa trong các cookie nếu bạn cho phép nó nhớ theo kiểu "Remember me on this computer" hoặc "Nhớ tên truy cập…".

Thực chất Cookie không có gì độc hại nhưng hiểm họa tiềm năng thì cũng có thể có. Những cookie thuộc dạng "siêu hiện đại" thời nay có thể thu thập được nhiều thứ hơn bạn tưởng. Tuy không nguy hiểm cỡ như spyware, virus, trojan.. nhưng việc chủ nhân các website tung cookie vào máy tính người dùng đã bị xem là bất hợp pháp tại Mỹ.

Tốt nhất bạn nên diệt sạch cookie bằng những công cụ "quét dọn" máy tính như: SBMAV Disk Cleaner, Your Uninstaller! 2006 Pro, System Cleaner, Fast Cleaner Gold, HS Cleandisk Pro, Morpheus Clean Disk v.v

Để ngăn cấm cookie xâm nhập vào máy tính khi sử dụng Internet Explorer, bạn làm như sau: Vào menu Tool -> Internet Option -> chọn thẻ Privacy -> nhấn vào nút "Advance" -> đánh dấu vào ô "Override automatic cookie handling", sau đó đánh dấu hết vào 2 ô "Block" để hoàn toàn chặn lại cookie và nhấn OK để hoàn tất.

Thiết lập cookies

Tất cả cookie đều được lưu trong đối tượng document.cookie .Điều này có nghĩa là nó nằm trong một đối tượng mảng.Và tất cả chuyện phải làm là "móc" cái ta cần ra khỏi đống lộn xộn ấy.

Nhưng mà đầu tiên hãy đi vào thiết lập cookies cái đã!

Việc thiết lập cookie cung khá đơn giản, mọi việc đơn giản chỉ là:

document.cookie="<tên cookies>=<nội dung>";

Nói thẳng ra là ta nhét một chuỗi mà đằng trước ký tự "=" được xem là tên, và sau đó là nội dung. Còn một điều nên lưu ý nữa là một domain (tức 1 website) chỉ có thể tạo 20 cookies trên một khách và chắc chắn rằng voà thời bủi này bạn không bao giờ muốn lảng phí thứ gì! Cho nên nếu không cần thiết thì các cookies phải được xoá bỏ.

Thiết lập thời gian tồn tại của cookies

Cookies sẽ chỉ tồn tại cho tới thời điểm mà bạn quy định, vi1 dụ như bạn quy định cho nó tồn lại tới này 1 tháng 1 chẵn hạn, nhưng lưu ý một điều, thời gian phải theo định dạng **GMT** và hơi bị nản để làm theo cách này. Một giải pháp khác chính là cho cookies tồn tại trong một khoảng thời gian từ khi thiết lập.

var exdate=new Date();

exdate.setDate(exdate.getDate()+365);

Với đoạn code trên, biến exdate đã lưu thời điểm trong nó là 365 ngày sau kể từ thời điểm cookies được thiết lập. Để thiết lập thời gian tồn tại cho cookies, bạn thêm **expire=**<*ngày hết hạn*> trong chuỗi nội dung của cookies.

Hảy thử nối tất cả lại và tạo một đoạn code js tạo một cookies, trong đó lưu trữ một chuỗi mà bạn nhập vào.

Code:

```
<script language="javascript">
 var exdate=new Date();
var your_name=prompt("What's your name!?","");
exdate.setDate(exdate.getDate()+365);
document.cookie=escape("last_time_visit="+your_name+";expire="+exdate.setDate(exdate.getDate(
)+365));
</script>
```

Hàm escape ó tác dụng chuyễn đổi những khý tự đặc biệt, ví dụ ***%20*** *sẽ thay thế cho khoản trắng* bài sau chúng ta sẽ đi vào tìm hiểu cách đọc những gì mà ta đã viết...

Đọc cookies

Code:

```
function getCookie(c_name)
{
if (document.cookie.length>0)
 {
  c_start=document.cookie.indexOf(c_name + "=");
  if (c_start!=-1) { c_start=c_start + c_name.length+1;
  c_end=document.cookie.indexOf(";",c_start);
  if (c_end==-1)
          c_end=document.cookie.length; return unescape(document.cookie.substring(c_start,c_end));
  }
}
 return "";
}
```

Ở đây mình có một hàm, hàm này sẽ đọc cookies theo tên, hàm này nhận vào một tham số **c_name** chính là tên cookies ta tìm, ta bắt đầu xem xét từng dòng: **if (document.cookie.length>0)** dòng này xác định xem trong đối tượng cookies có lưu thông tin gì không, nếu trong đối tượng cookies có lưu thông tin, ta mới bắt đầu lục cookies của mình ra.

c_start=document.cookie.indexOf(c_name + "=");

Chắc hẳn các bạn còn nhớ bài nói về việc kiểm tra phần tử select của form. Việc này cũng tương tự như thế, tìm trong chuỗi cookies xem có tên cookies của ta hay không. dòng **if (c_start!=-1)** sẽ xem xét điều kiện trên.

c_start=c_start + c_name.length+1;

c_end=document.cookie.indexOf(";",c_start);

if (c_end==-1) c_end=document.cookie.length;

Ba dòng này xác định vị trí đầu tiên và cuối cùng của cookies, chắc các bạn đã hiểu cách nó tìm chứ!? **return unescape(document.cookie.substring(c_start,c_end));** Dòng này trả giá trị của hàm trở về cookies được **unescape** (thực hiện hành động ngược lại với hàm escape ở bài trước)

ta có thể in ra và sử dụng cookies với đoạn javascript sau:

Code:

```
<script type="text/javascript">
username=getCookie('username');
if (username!=null && username!="")
```

```
{
alert('Welcome again '+username+'!');
}
Else
 {
 username=prompt('Please enter your name:','');
if (username!=null && username!="")
{
setCookie('username',username,365);
}
  }
</script>
```

Sau khi đoạn code này chạy, nó sẽ khởi động hàm check cookies, nếu giá trị của cookies trả về khác rỗng thì in ra, ngược lại thì chạy hàm **setCookie** để thiết lập cookies. Hàm này mình viết hơi khác với ví dụ trong bài trước để thuận tiện hơn.

Code:

```
function setCookie(c_name,value,expiredays)
{
  var exdate=new Date();
  exdate.setDate(exdate.getDate()+expiredays);
  document.cookie=c_name+ "=" +escape(value)+ ((expiredays==null) ? "" :
";expires="+exdate.toGMTString());
}
```

Tin hẳn các bạn đã đủ mọi điều để hiễu hàm trên, chỉ có lưu ý một chút: **((expiredays==null) ? "" : ";expires="+exdate.toGMTString());** Dòng này là dạng viết tĩnh lược của câu lệnh **if** gồm có vế bên trong cặp dấu ngoặc(phía trước dấu ?) là điều kiện. Phía sau dấu ? là hai dòng le65ng cho trường hợp đúng và sai, phân cách bởi dấu (:).

Mình nhắm vào sự trong sáng của mã, và nói thật, viết mã như trên đối với mình không dễ dàng gì, mình luôn thích dùng câu lệnh if cơ bản, và dùng các dấu ngoặc móc giới hạn các dòng lệnh, viết thế tuy rườm rà, không "chuyên nghiệp" theo một số người, nhưng lại rất dễ nhìn dễ hiểu, đúng theo nghĩa cơ bản .

Sở dĩ mình dùng nó trong ví dụ này là vì mình muốn giới thiệu thêm cho các bạn, chứ như lúc mình tự học, nhìn thấy mấy dòng này trong code người ta viết thì cũng phải nhăn mặt tư duy dữ lắm!

Ví dụ :

```
<html>
<head>
<script type="text/javascript">
function getCookie(c_name)
{
if (document.cookie.length>0)
  {
  c_start=document.cookie.indexOf(c_name + "=");
  if (c_start!=-1)
    {
    c_start=c_start + c_name.length+1 ;
    c_end=document.cookie.indexOf(";",c_start);
    if (c_end==-1) c_end=document.cookie.length
    return unescape(document.cookie.substring(c_start,c_end));
    }
  }
return ""
}
 function setCookie(c_name,value,expiredays)
{
var exdate=new Date();
exdate.setDate(exdate.getDate()+expiredays);
document.cookie=c_name+ "=" +escape(value)+((expiredays==null) ? "" : ";
expires="+exdate.toUTCString());
}
 function checkCookie()
{
username=getCookie('username');
if (username!=null && username!="")
  {
  alert('Welcome again '+username+'!');
  }
else
  {
```

```
  username=prompt('Please enter your name:','');
  if (username!=null && username!="")
    {
    setCookie('username',username,365);
    }
  }
}
</script>
</head>
<body onLoad="checkCookie()">
</body>
</html>
```

Đọc thêm về Cookie

Bài viết giải thích cách hoạt động của các tập tin gây ra nhiều tranh luận này, bạn sẽ tìm thấy chúng ở đâu và chúng có thể làm gì cho bạn cũng như ảnh hưởng thế nào đến bạn?

Định nghĩa: Cookie là một tập tin dữ liệu được web server ghi trên dĩa cứng của bạn nhằm nhận dạng bạn với một website.

- Giúp một website "ghi nhớ" bạn là ai và thiết lập các sở thích phù hợp khi bạn quay trở lại.
- Lọai bỏ yêu cầu điền lại nhiều lần các mẫu đơn đặt hàng hoặc đăng ký lại trên website.
- Cho phép các website hoặc các công ty quảng cáo theo dõi cách hoặc thói quen lướt web của bạn.

Thật thú vị nếu như bạn được nhận biết. Trên web, các site có thể chào hỏi bạn như một người bạn cũ nhờ vào các Cookie. Bắt đầu với Navigator 3.0 và Internet Explorer 6.0, các trình duyệt đã làm việc với các website nhằm ghi lại những mẩu thông tin nhận dạng bé nhỏ này trên dĩa cứng của bạn để giúp các website có thể theo dõi những hoạt động của bạn và nhận dạng khi bạn quay trở lại. Các Cookie hiện đang có mặt khắp nơi trên web nhưng những người dùng đang tranh cãi về bản chất của những tập tin bé nhỏ này. Một số người cho rằng các Cookie hứa hẹn một web thân thiện người dùng hơn, số người khác lại cho rằng chúng mang một mối de dọa đến sự riêng tư.

Khi bạn ghé thăm một site có thiết lập Cookie, các câu lệnh bên trong trang web này cho phép trình duyệt của bạn liên hệ với server của site. Server này gửi thông tin ngược lại cho trình duyệt và thông tin đó được lưu vào một nơi đặc biệt trên đĩa cứng của bạn. Các trình duyệt khác nhau sẽ lưu Cookie ở những nơi khác nhau. Netscape Navigator duy trì một tập tin có tên Cookie.txt chứa tất cả những Cookie từ bất cứ site nào có tạo ra Cookie. Trên một máy tính cá nhân chạy Windows, Internet Explorer lưu các Cookie của nó trong thư mục C:windows Cookies. Khi bạn quay trở lại site, server của site hỏi trình duyệt của bạn để tìm Cookie mà trước kia nó đã tạo và trình duyệt đáp lại bằng

cách gửi thông tin của Cookie cho server.

Có hai loại Cookie: "session Cookie" biến mất sau khi bạn đóng cửa sổ trình duyệt (chấm dứt phiên kết nối) và thường được các "shopping cart" sử dụng tại các cửa hàng trực tuyến nhằm theo dõi những món hàng bạn muốn mua. "Persist-ent Cookie" được thiếp lập bởi các site tin tức, các công ty quảng cáo biểu ngữ, và những site khác muốn nhận ra bạn khi bạn quay trở lại. Những tập tin này vẫn nằm trên đĩa cứng của bạn sau khi bạn rời khỏi site.

Cả hai tập tin Cookie đều chứa địa chỉ Internet (URL) hoặc domain name của site mà bạn đã ghé thăm và một vài đoạn mã chương trình bên trong cho biết những trang nào bạn đã xem qua. "Persistent cookie" còn thêm thông tin lần cuối bạn ghé thăm site và bạn đã ghé thăm bao nhiêu lần. Chúng thường chứa một đọan mã chương trình, nó cũng chính là định danh riêng của bạn, cho phép một site biết là trước đây bạn đã từng ghé thăm. Một số Cookie có thể chứa thông tin cá nhân như tên hoặc địa chỉ email, nhưng chỉ khi nào bạn cung cấp những thông tin đó cho web site. Trái với những lời đồn đã được nhiều người biết, Cookie không thể "đánh cắp" tên hoặc địa chỉ email của bạn nếu như bạn không đưa ra.

Bị bắt quả tang khi đang ăn vụng

Nếu bạn lướt trên web, có khả năng là có rất nhiều Cookie trên đĩa cứng của bạn. Gần như tất cả những website thương mại (kể cả PC World.com) và cả những site phi thương mại có đăng quảng cáo đều thiết lập Cookie. Có lẽ chỉ những trang web cá nhân không đăng quảng cáo và được tạo trên các server của ISP địa phương thì mới không thiết lập Cookie. Kích thước của Cookie thường không đến 100 byte, vì thế chúng không ảnh hưởng đến tốc độ duyệt web của bạn. Nhưng vì mặc định các trình duyệt được thiếp lập chấp nhận các Cookie nên không thể biết được ai đã tạo ra chúng. Nếu bạn quan tâm về sự riêng tư của bạn có thể tránh các site có thiết lập Cookie.

Các Cookie thực hiện vô số chức năng cho cả người lướt trên web lẫn các website. Đối với người dùng, chúng làm cho web tiện lợi hơn. Các site đòi hỏi phải đăng ký, như site của New York Times (http://www.nytimes.com), ghi Cookie chứa tên và mật khẩu của bạn lên đĩa cứng của bạn; cookie này đăng nhận thay cho bạn ở mỗi lần bạn ghé thăm. Các trang web được cá nhân hóa, như MyYahoo (http://my.yahoo.com), sử dụng càc cookie để tùy biến trang web chỉ hiển thị các tin tức, giá cổ phiếu và các thông tin khác mà bạn cho biêt bạn muốn xem. Các cửa hàng trực tuyến sử dụng cookie để ghi nhận những món hàng bạn đã cho vào "shopping cart" điện tử trước khi bạn rời site. Những site có thể dùng các cookie để trợ giúp việc điền các mẫu đơn đặt hàng để khi bạn mua hàng lần sau các thông tin về vận chuyển và thanh tóan được tự động điền vào.

Đối với các doanh nghiệp cookie có thể đóng một vai trò tương tự như một người bán hàng. Một số site buôn bán như Amazon.com, ghi lại những gì bạn đã mua vào cookie của bạn. Nhờ một cơ sở dữ liệu lưu lại số định danh duy nhất và những món hàng đã mua của bạn mà khi cần thiết, những site

này có thể giới thiệu những quyển sách hoặc đĩa nhạc mới dựa vào sở thích của bạn. Phần lớn các website (cũng như các nhà quảng cáo) còn thiết lập cookie để theo dõi bao nhiêu lượt người dùng ghé thăm site của họ. Hiện nay, con số kết quả này được xem như là số đo cho biết mức độ bận rộn của một site.

Một số nhóm người tiêu thụ cho rằng, các cookie có mặt trái của chúng là làm trái đi sự riêng tư của những người dùng. Các cookie theo dõi bạn đã ghé thăm những nơi nào và đã xem những gì trên web. Chỉ ghé vào thăm một site cũng đủ để "có" một vài cookie và không phải tất cả những cookie này thông báo lại trực tiếp với chính site mà bạn đang ghé thăm, một số site gửi thông tin này đến nhà quảng cáo trên site của họ. Cookie Central ngân hàng chứa các thông tin về cookie, cho biết một số công ty quảng cáo trên web, kể cả Focalink và DoubleClick, đã bí mật thiết lập các cookie thông báo lại trực tiếp với họ và theo dõi những cookie của bạn bằng một cơ sở dữ liệu. Bằng cách kiểm tra chéo nhiều cookie có về bạn, họ có thể mô tả sơ lược những sở thích, thói quen tiêu xài, và cách sống của bạn nhằmg tiếp thị đúng các sản phẩm đến bạn.

Thật ra bạn cũng có tiếng nói riêng của mình đối với vấn đề này: Các trình duyệt có các chức năng để chặn đứng các cookie. Cả Communicator và IE cho phép bạn vô hiệu hóa hòan tòan các cookie hoặc chúng sẽ nhắc bạn khi thiết lập một cookie. Commu-nicator có một chức năng cho phép chỉ chấp nhận những cookie gửi ngược lại server gốc của site. Nhưng vì có một số site chứa quá nhiều cookie nên thường không đáng để thiết lập cho các site thông báo đến bạn: Bạn có thể mất rất nhiều thời gian để chấp thuận hay phán đối các yêu cầu thiêt lập cookie.

Thay vì dùng trình duyệt của mình để ngăn chặc các cookie bạn có thể chạy các tiện ích để xóa bỏ các cookie ra khỏi địa cứng của bạn như IEClean hay NSClean (http://www.nsclean.com), cookie Crusher (http://www.thelimitsoft.com/cookie.html), và c; Cutter. Để bảo vệ kỹ hơn nữa, những site như The Rewebber (http://www.anonymizer.org) và các công cụ như Freedom của Zero Knowledge sẽ ngụy trang thông tin nhận dạng của bạn khi lướt trên web nên ngay cả nếu có một web có ghi cookie lên đĩa cứng của bạn đi nữa thì nó cũng chẳng biết bạn là ai.

JavaScript Form Validation

JavaScript có thể được dùng để kiểm tra những giá trị dữ liệu đầu vào trước khi những dữ liệu đó được gửi lên sever.

javaScript có thể được dùng để kiểm tra xem:

- Ô nhập liệu có trống hay không ?
- Người dùng có nhập 1 email hợp lệ hay không ?
- Người dùng nhập vào một ngày hợp lệ ?
- Người dùng có nhập đúng kiểu dữ liệu đầu vào hay không ?

Ví dụ dưới đây kiểm tra xem người dùng đã nhập vào form hay chưa ? nếu chưa sẽ có một thông báo hiện lên,nếu người dùng đã nhập thì form sẽ được gửi đi.

```
function validate_required(field,alerttxt)
{
with (field)
  {
  if (value==null||value==="")
    {
    alert(alerttxt);return false;
    }
  else
    {
    return true;
    }
  }
}
```

mã HTML đầy đủ như sau :

```
<html>
<head>
<script type="text/javascript">
function validate_required(field,alerttxt)
{
with (field)
  {
  if (value==null||value==="")
    {
    alert(alerttxt);return false;
    }
  else
    {
    return true;
    }
  }
}
```

```
function validate_form(thisform)
{
with (thisform)
  {
  if (validate_required(email,"Email must be filled out!")==false)
  {email.focus();return false;}
  }
}
</script>
</head>

<body>
<form action="submit.htm" onsubmit="return validate_form(this)" method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>

</html>
```

Còn đoạn code sau sẽ kiểm tra xem email của người dùng nhập vào có hợp lệ hay không ?

```
function validate_email(field,alerttxt)
{
with (field)
  {
  apos=value.indexOf("@");
  dotpos=value.lastIndexOf(".");
  if (apos<1||dotpos-apos<2)
    {alert(alerttxt);return false;}
  else {return true;}
  }
}
```

mã HTML đầy đủ như sau :

```
<html>
<head>
<script type="text/javascript">
function validate_email(field,alerttxt)
{
with (field)
  {
  apos=value.indexOf("@");
  dotpos=value.lastIndexOf(".");
  if (apos<1||dotpos-apos<2)
    {alert(alerttxt);return false;}
  else {return true;}
  }
}
function validate_form(thisform)
{
with (thisform)
  {
  if (validate_email(email,"Not a valid e-mail address!")==false)
    {email.focus();return false;}
  }
}
</script>
</head>
<body>
<form action="submit.htm" onsubmit="return validate_form(this);" method="post">
Email: <input type="text" name="email" size="30">
<input type="submit" value="Submit">
</form>
</body>
</html>
```

Một số kỹ thuật hướng đối tượng làm nền tảng cho các Javascript framework

Để đọc bài viết này, bạn cần có các kiến thức cơ bản về DOM và Javascript, đặc biệt là phần hướng đối tượng trong Javascript.

Nếu bạn là 1 web developer, bạn hẳn đã dùng Javascript trong các trang web của mình, chủ yếu để làm chất keo gắn kết các thành phần của giao diện. Tuy nhiên, Javascript còn có nhiều tính năng hướng đối tượng hơn bạn biết. Việc các trình duyệt bắt đầu hỗ trợ các bộ chức năng ngày càng chuẩn của Javascript và DOM đã mở ra khả năng viết các đoạn code phức tạp và hữu dụng hơn để chạy trên máy khách. Điều đó đã tạo nên hiện tượng AJAX.

Khi bắt đầu học cách viết các ứng dụng mang đậm tính AJAX, ta sẽ nhận thấy Javascript mà ta đã từng biết chỉ là phần nhỏ của tảng băng trôi. Javascript giờ đây được sử dụng không chỉ để phục vụ cho vài nhiệm vụ lặt vặt của giao diện như kiểm định thông tin nhập, mà đã trở nên cao cấp và phân lớp hơn rất nhiều, như một ứng dụng desktop thực sự. Ta có thể thấy các thư viện lớp, mô hình đối tượng, hệ thống phân cấp, các kiểu mẫu, và rất nhiều thứ khác mà ta chỉ thấy trong các code phía máy chủ. Chúng ta sẽ cần nâng cao kỹ năng Javascript của mình nhiều hơn nữa để viết được các ứng dụng kiểu mới. Nếu bạn muốn sử dụng các bộ thư viện Javascript như Prototype.js, Scriptaculous, moo.fx, Behaviour, YUI,..., rốt cuộc bạn sẽ vẫn phải đọc các code JS. Có thể do bạn muốn tìm hiểu họ làm như thế nào, bạn tò mò ưa khám phá, hoặc thông thường bởi đó là cách duy nhất giúp bạn biết làm thế nào để sử dụng chúng, vì tài liệu hướng dẫn dường như không được quan tâm đúng mức với hầu hết các bộ thư viện trên. Dù lý do là gì thì bạn cũng sẽ đối mặt với một số kỹ thuật khá xa lạ và... rùng rợn nếu chưa từng được biết tới những thứ gì tương tự .

Mục đích của bài viết này là giải thích chính xác các kiểu cấu trúc mà đa số chúng ta còn xa lạ.

1. JSON

JavaScript Object Notation (JSON) là một trong những từ được nhắc đến nhiều trong AJAX. JSON, nói nôm na là một cách thức khai báo đối tượng trong Javascript. VD:

Code:

```
var myPet = {
      color: 'black',
      legCount: 4,
      communicate: function(repeatCount){
            for(i=0;i<repeatCount;i++)
                  alert('Woof!');
      }
};
```

Ở đây, ta tạo tham chiếu tới một đối tượng với 2 thuộc tính (color và legCount) và 1 phương thức (communicate). Mỗi thành phần của đối tượng được thể hiện bởi tên, theo sau bởi dấu : và sau đó là định nghĩa. Thuộc tính được gán giá trị, còn phương thức được gán một hàm ẩn danh (anonymous). Sau khi đối tượng được tạo và gán cho biến myPet, ta có thể sử dụng nó như sau:

Code:

```
alert('my pet is ' + myPet.color);
alert('my pet has ' + myPet.legCount + ' legs');
//if you are a dog, bark three times:
myPet.communicate(3);
```

Bạn sẽ thấy JSON được sử dụng rất nhiều trong JS: làm đối số cho hàm, giá trị trả về, server response, v.v...

2. Hàm cũng là... đối tượng!

Nghe không được bình thường cho lắm, nhưng trong JS hàm cũng là đối tượng. Bạn có thể truyền hàm chạy lòng vòng như đối số tới hàm khác như bạn truyền một đối số thông thường (VD: chuỗi) vậy. Việc này được sử dụng rộng rãi và rất thuận tiện. VD:

Code:

```
var myDog = {
      bark: function(){
            alert('Woof!');
      }
};
```

```
var myCat = {
      meow: function(){
            alert('I am a lazy cat. I will not meow for you.');
      }
};

function annoyThePet(petFunction){
      //let's see what the pet can do
      petFunction();
}

//annoy the dog:
annoyThePet(myDog.bark);
//annoy the cat:
annoyThePet(myCat.meow);
```

Chú ý rằng, ta truyền myDog.bark và myCat.meow mà không có cặp dấu (). Nếu có (), đó sẽ không phải là truyền hàm, mà là lời gọi phương thức và truyền giá trị trả về, giá trị này là undefined trong cả 2 trường hợp chó và mèo.

Và nếu bạn muốn con mèo lười kia... sủa, bạn có thể dễ dàng làm như sau:

Code:

```
myCat.meow = myDog.bark;
myCat.meow(); //alerts 'Woof!'
```

Thế là Woof! Woof! Woof! 😄

3. Mảng, phần tử và thành phần của đối tượng

2 dòng code JS sau thực hiện cùng 1 công việc:

Code:

```
var a = new Array();
var b = [];
```

Như đã biết, ta có thể truy nhập vào từng phần tử của mảng bởi []:

Code:

```
var a = ['first', 'second', 'third'];
var v1 = a[0];
```

```
var v2 = a[1];
var v3 = a[2];
```

Ngoài chỉ số, ta còn có thể truy nhập vào từng thành phần của đối tượng bởi tên của chúng. VD dưới tạo một đối tượng rỗng, sau đó nó được đưa thêm các thành phần vào:

Code:

```
var obj = {}; //new, empty object
obj['member_1'] = 'this is the member value';
obj['flag_2'] = false;
obj['some_function'] = function(){ /* do something */};
```

VD trên thực hiện công việc giống hệt với VD dưới đây:

Code:

```
var obj = {
      member_1:'this is the member value',
      flag_2: false,
      some_function: function(){ /* do something */}
};
```

Như vậy, khái niệm đối tượng và mảng kết hợp (hash) trong JS là không phân biệt. 2 dòng dưới đây thực hiện cùng 1 công việc:

Code:

```
obj.some_function();
obj['some_function']();
```

4. Lớp

Sức mạnh to lớn của ngôn ngữ lập trình hướng đối tượng có được nhờ việc sử dụng lớp. Tôi không nghĩ rằng có thể dựa vào kinh nghiệm với các ngôn ngữ lập trình khác mà đoán ra được cách thức 1 lớp được định nghĩa trong JS (bởi nó... quá khác - ND):

Code:

```
//defining a new class called Pet
var Pet = function(petName, age){
      this.name = petName;
      this.age = age;
};

//let's create an object of the Pet class
```

```
var famousDog = new Pet('Santa\'s Little Helper', 15);
alert('This pet is called ' + famousDog.name);
```

Giờ hãy xem ta làm thế nào để thêm phương thức vào cho lớp Pet. Chúng ta sẽ sử dụng thuộc tính **prototype** mà mọi lớp trong JS đều có. Thuộc tính **prototype** là 1 đối tượng chứa tất cả các thành phần mà 1 đối tượng của lớp sẽ có. (Hay nôm na thuộc tính prototype chứa "nguyên mẫu" của 1 lớp, cho dễ hiểu - ND). Thậm chí các lớp mặc định của JS, như String, Number, Date,... đều có đối tượng prototype để ta có thể thêm vào đó các phương thức và thuộc tính giúp mọi đối tượng sinh ra từ lớp đó tự động có thêm các thành phần mới.

Code:

```
Pet.prototype.communicate = function(){
      alert('I do not know what I should say, but my name is ' + this.name);
};
```

Đó là khi một thư viện như prototype.js tỏ ra tiện lợi. Nếu sử dụng prototype.js, ta có thể làm cho đoạn code sáng sủa hơn:

Code:

```
var Pet = Class.create();
Pet.prototype = {
      //our 'constructor'
      initialize: function(petName, age){
            this.name = petName;
            this.age = age;
      },

      communicate: function(){
            alert('I do not know what I should say, but my name is ' +
this.name);
      }
};
```

5. Hàm đóng vai trò đối số

Code:

```
var myArray = ['first', 'second', 'third'];
myArray.each( function(item, index){
      alert('The item in the position #' + index + ' is:' + item);
});
```

Trong VD trên, ta sử dụng thư viện prototype.js, nó sẽ thêm hàm each vào lớp Array (cụ thể thế nào tạm thời ta không cần quan tâm, khi nào nghiên cứu về prototype hẵng hay - ND). Hàm each chấp nhận 1 đối số là 1 đối tượng hàm. Hàm này đến lượt nó sẽ được gọi một lần cho mỗi phần tử của mảng, với 2 đối số là phần tử hiện tại và chỉ số của nó. Ta có thể viết lại đoạn code trên như sau:

Code:

```
function myIterator(item, index){
      alert('The item in the position #' + index + ' is:' + item);
}

var myArray = ['first', 'second', 'third'];
myArray.each( myIterator );
```

Trong TH này, ta không cần hàm iterator ở chỗ nào khác trong code, do đó nên viết theo cách đầu tiên với hàm ẩn danh.

6. This cũng có thể là that!

Một trong những vấn đề thường gặp nhất khi ta bắt đầu code JS là từ khóa **this**.

Như đã nói ở trên, 1 hàm trong JS cũng là 1 đối tượng, và đôi khi chúng ta không nhận thấy mình đang truyền 1 hàm đi vòng vòng. Lấy VD:

Code:

```
function buttonClicked(){
      alert('button ' + this.id + ' was clicked');
}

var myButton = document.getElementById('someButtonID');
var myButton2 = document.getElementById('someOtherButtonID');
myButton.onclick = buttonClicked;
myButton2.onclick = buttonClicked;
```

Bởi vì hàm buttonClicked được định nghĩa bên ngoài mọi đối tượng, ta có thể sợ rằng từ khóa this sẽ chứa tham chiếu tới đối tượng window hoặc document.

Nhưng khi chạy đoạn code trên ta sẽ thấy nó làm việc đúng như mong đợi, và hiển thị lên id của

button được click. Điều xảy ra ở đây là chúng ta đã khiến phương thức onclick của mỗi button chứa tham chiếu tới đối tượng buttonClicked. Và do đó mỗi khi một button được click, trình duyệt sẽ thực hiện công việc như dòng code sau:

Code:

```
myButton.onclick();
```

1 VD khác:

Code:

```
var myHelper = {

      formFields: [ ],

      emptyAllFields: function(){
            for(i=0; i<this.formFields.length; i++){
                  var elementID = this.formFields[i];
                  var field = document.getElementById(elementID);
                  field.value = '';
            }
      }
};

//tell which form fields we want to work with
myHelper.formFields.push('txtName');
myHelper.formFields.push('txtEmail');
myHelper.formFields.push('txtAddress');

//clearing the text boxes:
myHelper.emptyAllFields();

var clearButton = document.getElementById('btnClear');
clearButton.onclick = myHelper.emptyAllFields;
```

Bây giờ bạn click button Clear và 3 textbox sẽ bị xóa. Sau đó bạn tiếp tục click nó, nhưng sẽ chỉ nhận được lỗi runtime. Lỗi này liên quan đến từ khóa this. Vấn đề ở đây là this.formFields chưa được định nghĩa, bởi vì this chứa tham chiếu tới đối tượng button (chứ không phải tham chiếu tới myHelper). Giải pháp nhanh gọn là viết lại dòng cuối cùng như sau:

Code:

```
clearButton.onclick = function(){ myHelper.emptyAllFields(); };
```

Bằng cách này ta đã tạo một hàm mới để gọi phương thức của helper trong ngữ cảnh của nó (this tham chiếu tới myHelper).

---------- THE END ----------

Đây là kiến thức cơ bản cần có trước khi chúng ta có thể dấn thân vào thế giới của các JS framework, hy vọng bài viết trên sẽ giúp các bạn tiêu hóa chúng dễ dàng hơn

JavaScript Animation

Ví dụ dưới đây sẽ cho các bạn thấy rõ hơn về JavaScript Animation.
ở đây javascript sẽ thay đổi giữa hai ảnh khác nhau dựa trên hai sự kiện khác nhau của con trỏ.
Đoạn mã JavaScript như sau :

```
<script type="text/javascript">
function mouseOver()
{
document.getElementById("b1").src ="b_blue.gif";
}
function mouseOut()
{
document.getElementById("b1").src ="b_pink.gif";
}
</script>
```

Đoạn mã này sẽ thực hiện việc thay đổi các bức hình khi sự kiện con trỏ thay đổi.
Đoạn mã HTML như sau :

```
<a href="http://www.w3schools.com" target="_blank">
<img border="0" alt="Visit W3Schools!" src="b_pink.gif" id="b1"
onmouseOver="mouseOver()" onmouseOut="mouseOut()" /></a>
```

Code HTML hoàn chỉnh như sau :

```
<html>
<head>
```

```
<script type="text/javascript">
function mouseOver()
{
document.getElementById("b1").src ="b_blue.gif";
}
function mouseOut()
{
document.getElementById("b1").src ="b_pink.gif";
}
</script>
</head>
<body>
<a href="http://www.w3schools.com" target="_blank">
<img border="0" alt="Visit W3Schools!" src="b_pink.gif" id="b1"
onmouseover="mouseOver()" onmouseout="mouseOut()" /></a>
</body>
</html>
```

JavaScript Image map

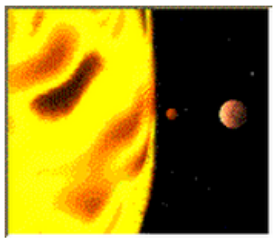
Một Image map là một ảnh mà trong đó có những vùng có thể click được,nghiã là chúng ta có thể đặt link ở các vùng của bức ảnh.

Chúng ta có thể thêm các event cho các thẻ area bên trong image-map.Các tag <area> hỗ trợ các OnClick, onDblClick, onMouseDown, onMouseUp, onmouseover, onMouseMove, onMouseOut, onKeyPress, onKeyDown, onKeyUp, onFocus, và các sự kiện onBlur.

Dưới đây là một ví dụ về Image-map có thêm mã JS các bạn có thể tham khảo.

```
<html>
<head>
<script type="text/javascript">
function writeText(txt)
{
document.getElementById("desc").innerHTML=txt;
}
</script>
```

```
</head>
<body>
<img src="planets.gif" width="145" height="126"
alt="Planets" usemap="#planetmap" />
<map name="planetmap">
<area shape ="rect" coords ="0,0,82,126"
onMouseOver="writeText('The Sun and the gas giant planets like Jupiter
are by far the largest objects in our Solar System.')"
href ="sun.htm" target ="_blank" alt="Sun" />
<area shape ="circle" coords ="90,58,3"
onMouseOver="writeText('The planet Mercury is very difficult to study
from the Earth because it is always so close to the Sun.')"
href ="mercur.htm" target ="_blank" alt="Mercury" />
<area shape ="circle" coords ="124,58,8"
onMouseOver="writeText('Until the 1960s, Venus was often considered a
twin sister to the Earth because Venus is the nearest planet to us, and
because the two planets seem to share many characteristics.')"
href ="venus.htm" target ="_blank" alt="Venus" />
</map>
<p id="desc"></p>
</body>
</html>
```



The planet Mercury is very difficult to study from the Earth because it is always so close to the Sun.

Các bạn hãy thử đưa con trỏ chuột đến vị trí của các hành tinh mà xem,mỗi hành tinh trên bức ảnh đã được đặt một link và bạn hoàn toàn có thể click vào đó.

JavaScript SetTimeOut và CleartimeOut

Thực ra thì setTimeOut và ClearTimeOut chúng ta đã tìm hiểu trước đây rồi.Bây giờ chỉ là nhắc lại mà thôi.

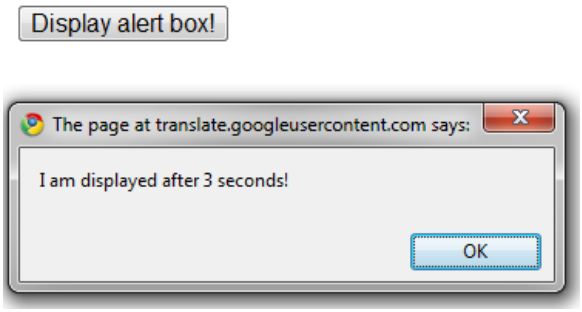
setTimeOut là hàm dùng để quy định một chức năng nào đó sau bao lâu sẽ bắt đầu được thực thi,ví dụ như bạn có một hàm hien_thi() và bạn muốn nó sau 5 giây thì bắt đầu thực thi thì bạn sử dụng hàm setTimeOut như sau :

setTimeOut("hien_thi()",5000);

ở ví dụ sau khi bạn click nút thì sau 5 giây một hộp thoại sẽ xuất hiện:

```
<html>
<head>
<script type="text/javascript">
function timedMsg()
{
var t=setTimeout("alert('I am displayed after 3 seconds!')",3000);
}
</script>
</head>
<body>
<form>
<input type="button" value="Display alert box!" onClick="timedMsg()" />
</form>
</body>
</html>
```

Khi bạn click button thì 5 giây sau hộp thoại sẽ hiện ra:

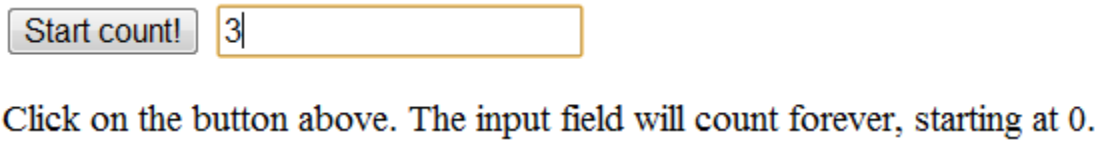


Một trong những ứng dụng quan trọng của setTimeOut là trong các vòng lặp vô tận,để làm được việc này chúng ta sẽ setTimeout lại chính hàm chứa SetTimeOut,để hiểu rõ hơn chúng ta tham khảo ví dụ sau ở đó khi bạn nhấn nút Start Count thì chương trình sẽ bắt đầu đếm từ 0.

```
<html>
<head>
```

```
<script type="text/javascript">
var c=0;
var t;
var timer_is_on=0;
function timedCount()
{
document.getElementById('txt').value=c;
c=c+1;
t=setTimeout("timedCount()",1000);
}
function doTimer()
{
if (!timer_is_on)
  {
  timer_is_on=1;
  timedCount();
  }
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!" onClick="doTimer()">
<input type="text" id="txt">
</form>
<p>Click on the button above. The input field will count forever, starting at 0.</p>
</body>
</html>
```

Kết quả hiện thị như sau :



ClearTimeOut

Hàm clearTimeOut dùng để dừng hàm setTimeOut ở trước đó,các bạn chú ý rằng nếu các bạn không dừng hàm setTimeOut lại trước khi tạo một setTimeOut mới thì cả hai setTimeOut sẽ chạy cùng nhau.Đó là lí do mà clearTimeOut được sinh ra với mục đích là dừng setTimeOut lại.

Các bạn cũng lưu ý rằng để có thể sử dụng ClearTimeOut lên một setTimeOut nào đó thì setTineout phải được gán vào một biến trước đó.Cụ thể là như thế này:

```
t=setTimeout("hien_thi()",5000);  /* gán biến t cho setTimeOut*/
clearTimeOut(t);                  /* dừng setTimeOut t */
```

ví dụ sau sẽ tạo ra một bộ đếm và một nút dừng đếm sử dụng clearTimeOut()

```
<html>
<head>
<script type="text/javascript">
var c=0;
var t;
var timer_is_on=0;

function timedCount()
{
document.getElementById('txt').value=c;
c=c+1;
t=setTimeout("timedCount()",1000);
}
function doTimer()
{
if (!timer_is_on)
  {
  timer_is_on=1;
  timedCount();
  }
}
```

```
function stopCount()
{
clearTimeout(t);
timer_is_on=0;
}
</script>
</head>
<body>
<form>
<input type="button" value="Start count!" onclick="doTimer()" />
<input type="text" id="txt" />
<input type="button" value="Stop count!" onclick="stopCount()" />
</form>
<p>
Click on the "Start count!" button above to start the timer. The input field will count forever, starting at 0. Click on the "Start count!" button to stop the counting. Click on the "Start count!" button to start the timer again.
</p>
</body>
</html>
```

Kết quả hiện thị như sau :

Start count! 3 Stop count!

Click on the "Start count!" button above to start the timer. The input field will count forever, starting at 0. Click on the "Stop count!" button to stop the counting. Click on the "Start count!" button to start the timer again.

Ví dụ sẽ tạo ra một đồng hồ điện tử các bạn có thể tham khảo:

```
<html>
<head>
<script type="text/javascript">
function startTime()
{
var today=new Date();
var h=today.getHours();
```

```
var m=today.getMinutes();
var s=today.getSeconds();
// add a zero in front of numbers<10
m=checkTime(m);
s=checkTime(s);
document.getElementById('txt').innerHTML=h+":"+m+":"+s;
t=setTimeout('startTime()',500);
}
function checkTime(i)
{
if (i<10)
  {
  i="0" + i;
  }
return i;
}
</script>
</head>
<body onload="startTime()">
<div id="txt"></div>
</body>
</html>
```

Kết quả hiện thị như sau :

11:21:28

JavaScript và lập trình hướng đối tượng- phần 1

OOP & JavaScript - Tại sao không? JavaScript là một ngôn ngữ tuyệt vời để viết các ứng dụng web theo phong cách hướng đối tượng. Nó có thể hỗ trợ OOP bởi vì nó hỗ trợ khả năng kế thừa và một số tính năng quan trọng của OOP thông qua các nguyên mẫu...

OOP & JavaScript - Tại sao không?

JavaScript là một ngôn ngữ tuyệt vời để viết các ứng dụng web theo phong cách hướng đối tượng. Nó có thể hỗ trợ OOP bởi vì nó hỗ trợ khả năng kế thừa và một số tính năng quan trọng của OOP thông qua các nguyên mẫu (prototype) cũng như các thuộc tính và phương thức, tuy không thực sự đầy đủ. Trên thực tế cách thực hiện kế thừa trong JS khác với các ngôn ngữ lập trình như C# hay Java. Sử dụng khả năng lập trình tiếp cận OOP sẽ làm cho chương trình của bạn có khả năng tái sử dụng cũng như bảo trì dễ dàng hơn rất nhiều.

Bài viết này tôi viết với mục đích hoàn toàn đơn giản là để tập hợp các kiến thức về JS đặc biệt là các cách thức lập trình JS tiếp cận OOP, thật ra là để chuẩn bị chủ yếu cho phần giao diện của ứng dụng web với công nghệ hiện đang khá nổi hiện nay là Ajax. Trong thực tế, việc lập trình web trên một số nền tảng bạn có thể không phải sử dụng nhiều đến JS, vậy nên mục đích chúng ta tìm hiểu về JS không ngoài việc hiểu và nắm bắt được cách thức tạo và xây dựng nên các framework hoặc các thư viện lập trình web.

Do vậy, bài viết này sẽ không phải là một bài trình bày chi tiết về JS, bởi điều đó đã có rất nhiều cuốn sách hay về JS mà các bạn có thể tìm đọc. Hầu hết những gì đề cập trong bài viết này đều dựa trên kinh nghiệm của cá nhân cũng như những kinh nghiệm mà tôi đã thu lượm được từ nhiều nguồn, vậy nên không có gì ngạc nhiên nếu các bạn thấy nó không có gì là mới mẻ, tuy nhiên, hi vọng không phải là toàn bộ bài viết này :).

Thực tế là JavaScript thật sự có thể làm được nhiều điều. Nếu bạn không tin, hãy bắt đầu với một cuốn sách về JS trên tay hoặc đơn giản chỉ cần đọc hết bài viết này !!:D.

JavaScript không phải là Java!

Đúng vậy! Nhưng các bạn có thể hỏi: Tại sao cả hai đều có chữ Java? Thật dễ hình dung điều này nếu chúng ta quan sát các nhãn hiệu, các sản phẩm trên thị trường ví dụ như Windows, Lindows, hay Bindows... Cái tên JavaScript đã được đổi tên từ LiveScript vào phút chót bởi phòng marketing của hãng Netscape. Trái ngược với nhận thức của mọi người, JavaScript không phải là ngôn ngữ theo dòng C/C++, nhưng thật không may cho nó vì ra đời sau Java và có tên gần giống cũng như một số cú pháp có kiểu tương tự Java nên khó tránh sự hiểu lầm.

Dưới đây là một số đặc điểm về JavaScript mà phần nào sẽ giúp bạn có thể hiểu hơn về JavaScript cũng như hiểu đúng về nó:

- *Code của JS được phiên dịch động (thông dịch):* điều này có nghĩa là tại thời điểm chạy thực, code được lưu dưới dạng text và dịch thẳng sang các chỉ thị máy khi chương chình thực thi, không giống như các ngôn ngữ khác như C hay Java là code phải được biên dịch trước khi chạy.

- *Các hàm (function) trong JavaScript là lớp đối tượng đầu tiên*:Các hàm trong JavaScript có thể gắn thêm vào các đối tượng như là các phương thức của đối tượng đó, tuy nhiên chúng có thể được gọi trong một ngữ cảnh khác hoặc có thể gắn vào một đối tượng khác trong thời điểm chạy thật. Khác với các ngôn ngữ khác như Java chẳng hạn, các phương thức của mỗi đối tượng đều chỉ được gọi bởi đối tượng đó và chỉ thuộc về đối tượng đó.
- *Đối tượng trong JavaScript là prototype-base:* Điều này nghĩa là bất cứ một đối tượng nào trong JavaScript được mô tả như là một đối tượng, cái thực ra là các mảng kết hợp dưới vỏ bọc như là các object. Sử dụng các prototype có thể giả lập các đối tượng tương tự như các đối tượng kiểu Java nhưng đó thực sự chỉ là bề ngoài mà thôi.

Có thể những điều nêu trên đây xem ra có vẻ khó hiểu, do vậy nếu không hiểu các bạn có thể xem nó như là đọc cho vui vậy!. Tuy nhiên trong các phần sau của bài viết, các bạn sẽ được hiểu rõ hơn qua các minh họa cụ thể. Còn bây giờ, chúng ta sẽ đi ngay vào chủ đề chính: JavaScript và lập trình hướng đối tượng (OOP).

Tạo một Object trong JavaScript.

1- Tại sao lại phải là Object?

Như các bạn đã biết, lập trình với các object thực sự tiện lợi, thứ nhất chúng có thể tương tác như các đối tượng trong thực tế, tiếp đến là khả năng tái sử dụng và bảo trì code dễ dàng hơn rất nhiều so với cách lập trình chỉ sử dụng các hàm (tất nhiên là chúng ta có thể sử dụng kết hợp cả hai phương pháp bởi hầu hết các đoạn mã viết bằng JavaScript thường không lớn). Một đặc điểm đặc trưng khi làm việc với các object là chúng ta làm việc với các thuộc tính và các phương thức của chúng, vì vậy nếu như chúng ta có một đối tượng là quả táo chẳng hạn thì thuộc tính có thể là màu sắc, giống táo, kích thước, và phương thức của quả táo có thể là "chín" và "rụng". Khi chín thuộc tính màu sắc và khối lượng có thể bị thay đổi, cũng như khi rụng quả táo có thể bị phân hủy. Như vậy việc lập trình với các đối tượng sẽ trở nên thật sự thú vị, giống như công việc mà chúa trời đã tạo ra mọi vật trên thế giới này vậy!

Sau đây chúng ta sẽ cùng nhau tìm hiểu một số cách mà JavaScript giúp chúa tạo ra thế giới:D

2- Tạo object sử dụng đối tượng được định nghĩa sẵn: Object

Trong JavaScript có rất nhiều các đối tượng được định nghĩa trong nó như String object, Array object, Function object, Object object .v.v... Tuy nhiên JavaScript không cung cấp cho chúng ta một quy tắc chuẩn để tự xây dựng một object như các ngôn ngữ hướng đối tượng khác, vì vậy việc tạo ra

các object cho riêng mình sẽ dựa trên các object cơ bản đó. Và vì lí do này mà các bạn sẽ thấy có khá nhiều cách để khai báo một object trong JavaScript.

Sau đây, chúng ta sẽ xây dựng một đối tượng dựa trên đối tượng có sẵn Object. Với câu lệnh**new** bạn sẽ tạo ra được một đối tượng theo cách đơn giản nhất:

```
1.  //Tạo object Quả táo của Newton
2.  apple = new Object();
3.
4.  //Khai báo thuộc tính cho object
5.  apple.color = "green";
6.  apple.mass = "90g";
7.
8.  //Khai báo các phương thức cho object
9.  apple.ripe = function() {
10.     this.color='red';
11.     this.status='This apple has ripen';
12. }
13. apple.fall=function(){
14.     this.status='Fall on Newton's nose'
15. }
16.
17. //Test object
18. alert(apple.color);
19. apple.ripe();
20. alert(apple.status);
21. apple.fall();
22. alert(apple.status);
```

Như các bạn thấy trong ví dụ trên, ta vừa tạo ra một đối tượng sử dụng new Object(). Nếu để ý các bạn sẽ thấy với cách này, các thuộc tính cũng như phương thức của đối tượng được tạo ra bất cứ khi nào ta muốn với các giá trị được gán trước. Nhưng cũng với cách này các bạn sẽ nhanh chóng nhận ra rằng các đối tượng gần như không thể kiểm soát khi số lượng thuộc tính và phương thức tăng lên. Vì vậy, JavaScript đã cung cấp thêm một phương pháp cài đặt đối tượng tiện lợi hơn mà chúng ta sẽ tìm hiểu sau đây.

3 – Tạo đối tượng sử dụng JavaScript Object Notation (JSON):

JSON là một nét đặc trưng của ngôn ngữ này. Nó cung cấp cho lập trình viên một khả năng tạo các mảng cũng như phác thảo ra các đối tượng một cách đơn giản và ngắn gọn. Để hiểu được JSON, trước hết ta tìm hiểu xem cách thức mà mảng trong JavaScript hoạt động.

JavaScript cung cấp cho chúng ta một đối tượng Array. Cách cài đặt một mảng mới khá đơn giản, ví dụ:

```
1.      myArray = new Array();
2.
3.          myLibrary.books = new Array();
```

Khá giống với các ngôn ngữ như Java hay C, JavaScript cho phép khai báo trước số phần tử của mảng cũng như gán một giá trị cho một phần tử của mảng:

```
1.      family=new Array(4);
2.
3.          family[0] = "Tom";
```

hoặc thậm trí ta có thể kết hợp với một giá trị khóa như sau:

```
1.      family["father"]="Tom";
```

Tất nhiên bạn có thể gán bất cứ giá trị nào cho mảng thậm trí là một object, bởi JavaScript là một ngôn ngữ có kiểu khá lỏng lẻo. Cách khởi tạo giá trị mảng như trên có vẻ nhanh gọn, nhưng nếu như mảng của chúng ta khá lớn và phức tạp thì việc khai báo như vậy sẽ rất tẻ nhạt. Một cách khác để để khởi tạo các giá trị ban đầu cho mảng là sử dụng cặp ngoặc vuông [ ]:

```
1.      family=["Tom","Jery","Jack","Timy"];
```

Hoặc nếu dùng cặp đấu { } kết hợp với giá trị khóa như sau:

```
1.      family={
2.
3.              farther: "Tom",
4.
```

```
5.          mother: "Jery",
6.
7.          first_son: "Jack",
8.
9.          second_son: "Timy"
10.
11.       };
```

Đây chính là cách mô tả đối tượng kiểu JSON.

Nếu muốn sử dụng dấu cách trống cho các khóa ta có thể dùng cặp dấu "" ví dụ:

```
1.      "first son": "Jack",
```

Chúng ta cũng có thể lồng các JSON vào nhau để khai báo các đối tượng phức tạp hơn, ví dụ:

```
1.      <font face="Courier New, monospace"><font size="2">var
2.      rectangle={</font></font>
3.      </p><p><font face="Courier New, monospace"><font size="2"><span>
        topleft:</span></font></font><font face="Courier New, monospace"><font size="2"><b>{
4.      x:1,      y:2 }</b></font></font><font      face="Courier      New,      monospace"><font
        size="2"><span>,</span></font></font></p>
5.      <p><font face="Courier New, monospace"><font size="2">          width:
6.      5,</font></font></p>
7.      <p><font face="Courier New, monospace"><font size="2">          height:
8.      10</font></font></p>
9.      <p><font face="Courier New, monospace"><font size="2">      };</font></font></p>
10.     <p><font face="Courier New, monospace"><font size="2">      //Khi
11.     đó</font></font></p>
12.     <p><font face="Courier New, monospace"><font size="2">      alert(rectangle.topleft.x);
13.     // sẽ đưa ra giá trị 1</font></font></p><p class="javascript">
```

Để thêm vào một phương thức cho đối tượng khai báo theo kiểu JSON ta làm khá đơn giản như sau:

```
1.      var rectangle={
2.
3.              //Thuộc tính
```

```
4.
5.              topleft:{ x:1, y:2 },
6.
7.              width: 5,
8.
9.              height: 10,
10.
11.             //Phương thức
12.
13.             area:function()
14.
15.             {
16.
17.                 return this.width*this.height;
18.
19.             }
20.
21.         };
22.
23.         //Khi đó
24.
25.         alert(rectangle.topleft.x); // sẽ đưa ra giá trị 1
26.
27.         alert(rectangle.area()); // sẽ đưa ra giá trị 50
```

Trong ví dụ trên, ta sử dụng từ khóa **this** để truy cập vào các thuộc tính vừa khai báo của bản thân đối tượng. Đây là một từ khóa đặc biệt và khá mạnh, ta sẽ có dịp đề cập đến nó nhiều hơn trong các phần sau.

Như vậy, bây giờ các bạn có thể sử dụng đối tượng Object hoặc JSON để khai báo một đối tượng trong JavaScript hoặc thậm trí sử dụng kết hợp cả hai phương pháp trên. Tuy nhiên, nếu bạn thường xuyên phải sử dụng, cũng như tạo mới rất nhiều đối tượng trong chương trình thì cả hai phương pháp trên tỏ ra kém hiệu quả, bởi lẽ với mỗi lần các thuộc tính thay đổi bạn lại phải thực hiện các khai báo khá phức tạp do vậy tính tái sử dụng của code không được đảm bảo. Thật may mắn, JavaScript cung

cấp một cách thức khác hiệu quả hơn rất nhiều để đáp ứng nhu cầu đó của những lập trình viên, cái mà chúng ta sẽ xem xét trong phần tiếp theo đây và cũng như phần còn lại của bài viết này.

JavaScript và lập trình hướng đối tượng- phần 2

Class và prototype 1- Xây dựng class trong JavaScript Trong lập trình hướng đối tượng, chúng ta thường tạo những đối tượng từ các lớp (class), trong đó các object được coi như là trường hợp cụ thể của một lớp. Cả hai ngôn ngữ Java và JavaScript đều cung...

Class và prototype

1- Xây dựng class trong JavaScript

Trong lập trình hướng đối tượng, chúng ta thường tạo những đối tượng từ các lớp (class), trong đó các object được coi như là trường hợp cụ thể của một lớp. Cả hai ngôn ngữ Java và JavaScript đều cung cấp cho chúng ta từ khóa new dùng để tạo một thể hiện của một đối tượng đã được định nghĩa trước đó.

Trong Java, mọi thứ đều là object, và được phát triển từ class nguyên thủy là java.lang.Object. Chắc hẳn nếu bạn là lập trình viên Java hẳn sẽ thấy câu lệnh sau khá quen thuộc:

[quote] MyObject myObj=new MyObject(arg1,arg2);[/quote]

Với JavaScript cũng có cách khai báo rất giống như vậy:

[quote] var myObj= new MyObject(arg1, arg2);[/quote]

Nhưng có một sự khác biệt lớn về bản chất đó là chúng ta hoàn toàn không định nghĩa lớp MyObject như trong Java mà thay vào đó là một hàm thật sự có cùng tên:

```
[quote]    function MyObject(){
        //do something here
    }

[/quote]
```

Để minh họa cho điều này, ta xét ví dụ sau khai báo một class Animal với các thuộc tính đơn giản như name, food và phương thức đơn giản là eat:

```
    function Animal(name, food){
```

```
    //Thuộc tính
    this.name=name; //Tên con vật, hoặc tên loài
    this.food=food; //Loại thức ăn có thể ăn

    //Phương thức
    this.eat=function(something){
 if(something===food){ //Nếu thức ăn phù hợp

        alert('This is very delicious');

      }else{

        alert("I don't like this");

      }

    }

  }

  var myAnimal = new Animal("Cat","Mouse");

  myAnimal.eat("Mouse");// This is very delicious
```

Trong ví dụ trên, một lần nữa ta sử dụng từ khóa this, với cấu trúc trên thì ta có thể hiểu this là đối tượng được tạo ra sau từ khóa new.

Tất nhiên với cách khai báo như vậy mọi việc đều OK, nhưng nếu đi sâu hơn một chút chúng ta sẽ thấy: đầu tiên, cho mỗi thể hiện của class Animal mà chúng ta tạo ra, chúng ta cũng tạo ra một hàm eat() mới, điều này dẫn đến vấn đề về bộ nhớ và tốc độ xử lý nếu bạn có ý định tạo ra rất nhiều đối tượng trong chương trình từ class Animal (ví dụ tạo một đàn kiến chẳng hạn!!!); thứ nữa, khi làm việc với các DOM (Document Object Model), chúng ta sẽ gặp phải vô số vấn đề rắc rối không mong đợi. Chính vì vậy bây giờ ta hãy thử một cách khác an toàn hơn, và nó được biết đến cái tên khá lạ tai: prototype-base.

2- Prototype

Prototype là gì vậy? Thực ra đây là một thuộc tính sẵn có trong mọi object trong JavaScript (trong JavaScrip thuộc tính cũng có thể là một đối tượng, ta sẽ đề cập chi tiết hơn về điều này sau), đây có thể coi là nét đặc trưng của JavaScript mà các ngôn ngữ hướng đối tượng khác không có. Các hàm cũng như các thuộc tính đều có thể kết hợp được với prototype. Prototype và từ khóa new có thể làm việc cùng nhau, khi một đối tượng được tạo bởi từ khóa new thì tất cả các thuộc tính và phương thức

prototype đều được gắn vào đối tượng mới đó. Điều này nghe có vẻ lạ và rắc rối, nhưng khá là hiệu quả:

Ta xây dựng lại class Animal theo phong cách prototype như sau:

```
function Animal(name, food){
    //Thuộc tính
    this.name=name; //Tên con vật, hoặc tên loài
    this.food=food; //Loại thức ăn có thể ăn
}
Animal.prototype.eat=function(something){
    if(something===this.food){//Nếu thức ăn phù hợp
        alert('This is very delicious');
    }else{
        alert("I don't like this");
    }
}
var myAnimal = new Animal("Cat","Mouse");
myAnimal.eat("Mouse");// This is very delicious
```

Như vậy đầu tiên chúng ta vẫn khai báo class với các thuộc tính bình thường, sau đó đối với các phương thức ta chỉ việc gắn nó vào prototype như trên. Khi chúng ta tạo một thể hiện của class thì phương thức được gắn vào object đó mà không phải tạo mới, và từ khóa this sẽ đảm bảo rằng trường food là của object vừa tạo.

Trong việc thao tác với prototype, có một lưu ý là mọi object được tạo ra sau prototype thì sẽ được gắn các thuộc tính hoặc phương thức được khai báo kiểu prototype trước đó, ví dụ:

```
myObject.prototype.color="red";
var myObj1=new myObject();
myObject.prototype.color="blue";
```

```
myObject.prototype.sound="boom!";
var myObj2=new myObject();
alert(myObj1.color); //red
alert(myObj2.color); //blue
alert(myObj2.sound); //boom!
alert(myObj1.sound); //erro!
```

Khi đó myObj1 chỉ có thuộc tính color với giá trị là red, còn myObj2 có thêm thuộc tính sound và giá trị color là blue.

**Mở rộng một object trong JavaScript*

Trên thực tế, cơ chế prototype còn có thể áp dụng cho các đối tượng xây dựng sẵn trong JavaScript, ta có thể dùng nó để mở rộng các đối tượng này. Một ví dụ rất hữu ích là mở rộng đối tượng Array như sau:

```
Array.prototype.indexof=function(obj){
        var result=-1;
        for(var i=0;i
            if(this[i]==obj){
            return i;
            break;
        }
    }
    return result;
}
var ary=new Array();
ary=["one","two","three"];
alert(ary.indexof("one")) //0
```

Ví dụ trên, ta mở rộng Array bằng cách thêm vào một phương thức indexof tính chỉ số của một phần tử trong mảng, nếu đối tượng không có trong mảng thì trả về giá trị là -1;

3- Cơ chế của prototype

Một câu hỏi sẽ được đặt ra là cơ chế hoạt động của prototype như thế nào? Tại sao với prototype ta có thể giả lập khá nhiều các đặc tính hướng đối tượng? Thật ra prototype lại có một cơ chế hoạt động khá đơn giản: Mỗi khi bạn thực hiện thao tác với thuộc tính và phương thức của một đối tượng nào đó trong JavaScript, thì trình thông dịch sẽ thực hiện các bước tuần tự sau để xác định thuộc tính hay phương thức nào được thực thi:

- Nếu thuộc tính hay phương thức của đối tượng đang xét có giá trị hoặc đã được gán giá trị thì thuộc tính hay phương thức đó được sử dụng.
- Nếu không thì kiểm tra giá trị của thuộc tính của prototype trong cấu trúc của object.
- Cứ tiếp tục như vậy cho đến khi tìm thấy thuộc tính phù hợp (thuộc tính đã được gán giá trị) hoặc giá trị tìm được là kiểu *Object*.

Chính vì vậy bất cứ khi nào muốn thêm một thuộc tính hay gắn thêm một phương thức mới cho một object bất kỳ ta chỉ việc khai báo nó như là thuộc tính của prototype.

Để thấy rõ hơn điều này ta sẽ cùng nhau tìm hiểu sâu hơn một chút về các thuộc tính và phương thức của các đối tượng cài đặt sẵn bên trong JavaScript là Object và Function

a) Object:

Thuộc tính:

constructor

prototype

Phương thưc:

hasOwnProperty()

isPrototypeOf()

toString()

valueOf()

toLocaleString()

Ở đây chúng ta chỉ cần lưu ý phương thức khá đặc biệt đó là hasOwnProperty();

Với hasOwnProperty() sẽ trả về giá trị true nếu object đang xét có thuộc tính nào đó nhưng không phải là thuộc tính được khai báo kiểu mở rộng prototype, ví dụ:

```
var myObj=new Object();
myObj.firstProperty="xyz";
myObj.prototype.secondProperty="abc";
alert(myObj.hasOwnProperty("firstProperty")) //true
alert(myObj.hasOwnProperty("fdasffsdf")) //false
alert(myObj.hasOwnProperty("secondProperty")) //false
```

b) Function

Thuộc tính:

constructor

prototype

arguments

arity

caller

length

Phương thức:

apply()

call()

toString()

valueOf()

Với đối tượng Function, chúng ta sẽ tìm hiểu kĩ hơn một chút bởi lẽ đây là đối tượng chính dùng trong lập trình OOP với JavaScript.

Thuộc tính constructor: thuộc tính này trả về một constructor mà từ đó object được tạo ra, ví dụ:

```
var myFunc=new Function();
```

```
alert(myFunc.constructor); // kết quả sẽ là Function
```

Lưu ý rằng thuộc tính này chỉ có trong các biến có kiểu object, nên với một biến bất kỳ muốn sử dụng constructor thì ta phải kiểm tra kiểu trước khi sử dụng, ví dụ.

```
if(typeof myVar =="object"){
    alert(myVar.constructor);
}
```

Thuộc tính arguments: arguments thực chất là một mảng chứa các tham số cho function, khi function được gán cho một đối số thì đối số này sẽ được đẩy vào mảng arguments, ta xét ví dụ sau:

```
function test(){
    var arg=test.arguments;
    for(var i=0;i
        alert(arg[i]);
    }
}
test(1,"a");// sẽ đưa ra giá trị 1 và sau đó là "a"
```

Để ý rằng hàm test ban đầu không khai báo tham số, nhưng khi chạy ta đã thêm hai tham số vào cho nó do vậy trong arguments sẽ có 2 giá trị.

Phương thức call() và apply(): Đây là hai phương thức của object Function, dùng để thực thi, hoặc gọi một function. Tham số đầu của hai phương thức trên thường là một object, cái mà sẽ được coi là object hiện thời và là tham số cho hàm thực thi phương thức call() hay apply(). Trong thực tế người ta thường dùng call thay cho apply bởi chức năng không có gì khác nhau, chỉ khác ở tham số truyền vào cho hai phương thức. Để minh họa điều này ta xét ví dụ sau:

```
function showInfo(){
    alert(this.name);
}
function myObject(name){
    this.name=name;
```

```
    this.show=showInfo;
}
var myObj=new myObject("AxitDN");
//Cách thông thường
myObj.show(); //kết quả cho AxitDN
//Sử dụng call
showInfo.call(myObj); //kết quả cho AxitDN
```

Ví dụ trên cho ta thấy rằng trong JavaScript, các hàm được sử dụng một cách khá tự do, thậm trí được sử dụng với một mục đích hoàn toàn khác so với lúc nó được tạo cho đến khi kết thúc chương trình. Không những thế với thuộc tính đặc biệt prototype càng làm cho chương trình viết bằng JavaScript trở nên vô cùng sinh động.

Tóm tắt những điều đã học

Qua những bài học trên chúng ta cũng đã phần nào hiểu được ít nhiều về JavaScript,chúng ta cũng đã biết cách làm thế nào để thêm một đoạn mã JS vào trang HTML cũng như là cách để nó tương tác với các sự kiện trong trang web của bạn.Vậy bước tiếp theo của chúng ta là gì ?
Câu trả lời của bạn là : Hãy tìm hiểu về DOM HTML và DHTML.
DOM HTML qui định các cách thức chuẩn để truy cập vào tài liệu HTML.
Còn DHTML là một sự kết hợp của HTML,CSS và JavaScript nó sẽ tạo ra sự năng động trong trang web của bạn.

Chương 4

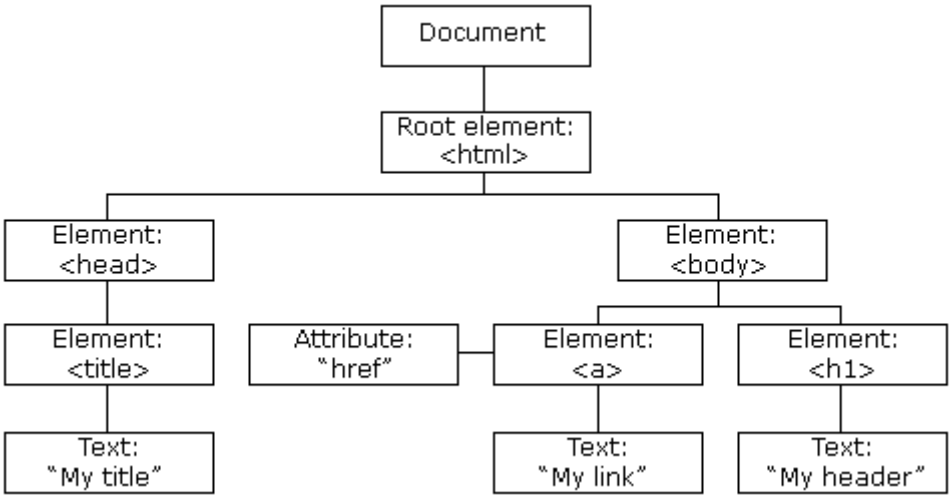
DOM HTML

Mở đầu về DOM HTML

Trước đây trong những bài thực hành về JavaScript có thể các bạn khó hiểu ở một số đoạn mã mà ở đó JavaScript tác động lên các thành phần HTML.

Nhiệm vụ của chương này sẽ giúp bạn hiểu rõ hơn về vấn đề đó,cụ thể là làm cách nào để JavaScript có thể tác động lên các thành phần HTML.

Dom HTML sẽ qui định những cách thức chuẩn để truy cập vào các phần tử HTML,để làm được việc này DOM xem một tài liệu HTML có dạng cây như sau :



Để nghiên cứu DOM trước hết các bạn phải có kiến thức về HTML và JavaScript

DOM được chia thành 3 thành phần có cấp độ khác nhau :

- Core DOM : đây là mô hình của Dom cho bất kì tài liệu có cấu trúc nào
- XML DOM : mô hình DOM cho các tài liệu XML
- HTML DOM : mô hình DOM cho các tài liệu HTML

DOM định nghĩa các đối tượng và thuộc tính của tất cả các thành phần của HTML và các phương thức để truy cập chúng.

HTML DOM là gì ?

HTML DOM là :

- Một mô hình đối tượng tiêu chuẩn của HTML
- Một giao diện lạp trình chuẩn của HTML
- Nó là mộ nền tảng và ngôn ngữ độc lập
- Nó là một tiêu chuẩn của W3C

Chúng ta có thể hình dung DOM HTML là một tiêu chuẩn cung cấp các cách thức để làm thế nào có thể thêm ,sửa,xóa,thay đổi các thành phần HTML.

HTML DOM nodes

DOM xem các thành phần của trang HTML là những nút.Nó cho biết :

- Toàn bộ trang HTML là một nút
- Mỗi phần tử HTML là một nút
- Các văn bản trong trang HTML là nút văn bản
- Mỗi thuộc tính HTML là một nút thuộc tính
- Những comment là những nút comment

Các bạn hãy xem ví dụ sau :

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

Trong tài liệu HTML trên thì nút gốc là <html>,tất cả những nút khác là chứa trong nút <html>.

Nút <html> này lại có hai nút con là <head> và <body>.

Trong khi nút <head> có nút con là <title> thì nút <body> cũng có hai nút con là <h1> và <p>.

Văn bản luôn được chứa trong các nút text

Một trong những vấn đề phổ biến là khi truy cập các nút có chứa văn bản.Để giải quyết vấn đề này các bạn phải lưu ý rằng các văn bản trong trong HTML luôn được chứa trong nút text.

Chẳng hạn như chúng ta có một đoạn HTML như sau :

```
<title> DOM HTML </title>
```

Như vậy thì nút <title> đang giữ một nút con là nút text,nút text này có giá trị là : “DOM HTML”,các bạn không được hiểu rằng giá trị của nút <title> là “DOM HTML “ !.

Nhưng DOM cung cấp cho ta một kĩ thuật có thể truy cập vào giá trị của các nút văn bản đó là sử dụng thuộc tính **innerHTML**,chẳng hạn như ví dụ trên để lấy ra giá trị “DOM HTML “ thì chúng ta chỉ cần viết :

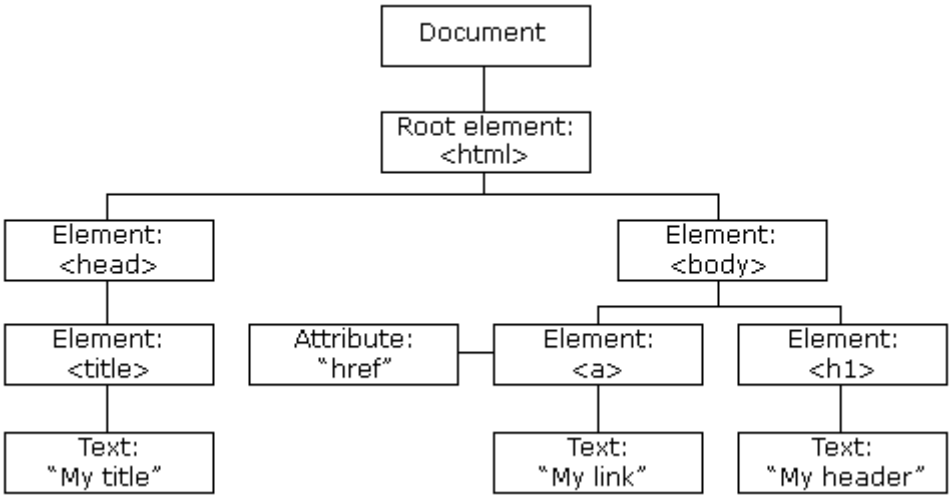
```
Document.getElementByTagName(“title”).innerHTML
```

Vấn đề này sẽ được đề cập nhiều hơn ở trong các bài sau.

HTML DOM node tree

HTML DOM hiện thị một tài liệu HTML như một cấu trúc cây,cấu trúc cây đó được gọi là node-tree,tất cả các nút có thể được truy cập thông qua cây nút (node-tree) và chúng ta có thể cập nhật hay xóa đi nội dung của nút và thậm chí bạn có thể tạo ra đối tượng mới trên cây nút đó.

Dưới đây là một ví dụ về cây nút (node-tree) trong đó nó đã chỉ ra mối quan hệ giữa các nút với nhau,cây nút bắt đầu từ nút gốc và tỏa dần ra đến vị trí kết thúc của cây nút là những text-node.

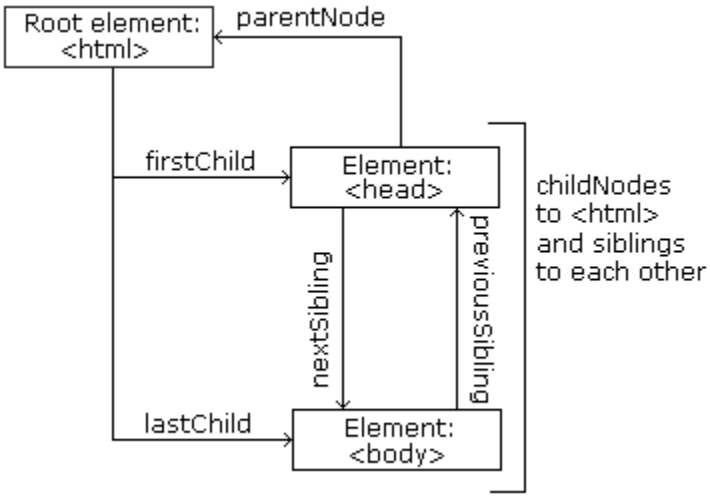


Nút cha,con và nút anh em ?

Những nút trong cùng một cây rõ rang là có quan hệ với nhau,và người ta đã chia ra thành nút cha,con và an hem.

- Nút đầu của cây nút được gọi là nút gốc
- Mỗi nút của cây (trù nút gốc) có duy nhất một cha
- Một nút có thể có nhiều nút con
- Nút lá là nút không có nút con
- Các nút có cùng nút cha thì gọi là nút anh em

Hình ảnh sau mô tả một phần của cây nút và mối quan hệ giữa các nút:



Các bạn hãy xem đoạn HTML sau đây :

```
<html>
  <head>
    <title>DOM Tutorial</title>
  </head>
```

```
  <body>
    <h1>DOM Lesson one</h1>
    <p>Hello world!</p>
  </body>
</html>
```

Như trên thì :

- Nút <html> không có nút cha,nó là nút gốc
- Nút cha của nút <head> và <body> là nút <html>
- Nút cha của text – node " Hello world" là nút <p>
- Nút <html> có hai nút con là <body> và <head>
- Nút <head> có một nút con là nút <title>
- Nút <title> lại có một nút con là text-node "DOM tutorial"
- <h1> và <p> là hai nút anh em,và chúng đều là con của nút <body>

First child – Last Child ?

ở ví dụ trên chúng ta có một số điều sau :

- Nút <head> là nút con đầu tiên first-child của nút <html> trong khi nút <body> lại là nút con cuối cùng last-child của nút <html>
- Nút <h1> là first-child của nút <body> , <p> là last-child của nút <body>

HTML DOM Methods

Dom cung cấp dao diện lập trình

Như đã nói ở các bài trước thì DOM coi tài liệu HTML là một tập hợp các nút,và chúng ta có thể sử dụng các ngôn ngữ lập trình (ở đây cụ thể là JavaScript) để truy cập cũng như là tác động vào các nút đó.

Dao diện lập trình ở đây được DOM sử dụng là các phương thức và thuộc tính.

HTML DOM Properties

Chúng ta có một số DOM properties như sau :

- x.innerHTML – giá trị trả về là giá trị của node- text con của x
- x.nodeName – trả về tên của x
- x.nodeValue – trả về giá trị của x

- x.parentNode - nút cha của x
- x.childNodes - các nút con của x
- x.attributes - các thuộc tính các nút của x

chú ý rằng ở trên thì x là một đối tượng nút.

HTML DOM Methods

Chúng ta có một số DOM Methods như sau :

- x.getElementById *(id)* - nhận được các phần tử với một id xác định
- x.getElementsByTagName *(name)* - nhận được tất cả các phần tử với một tên tag quy định
- x.appendChild *(node)* - chèn thêm một nút con để x
- x.removeChild *(node*)- loại bỏ một nút con từ x

chú ý rằng ở trên thì x là một đối tượng nút.

Thuộc tính innerHTML

Đây là một thuộc tính khá quan trọng cho nên chúng ta phải hiểu rõ về nó.

Nó cung cấp cách dễ nhất để thực hiện việc lấy hoặc sửa đổi nội dung của một phần tử.

innerHTML được hỗ trợ bởi tất cả các trình duyệt.

ví dụ dưới đây sẽ lấy một đoạn văn bản nằm trong cặp thẻ   có id=”intro”

```
<html>
<body>
<p id="intro">Hello World!</p>
<script type="text/javascript">
txt=document.getElementById("intro").innerHTML;
document.write("<p>The text from the intro paragraph: " + txt + "</p>");
</script>
</body>
</html>
```

Kết quả hiện thị như sau :

Hello World!

The text from the intro paragraph: Hello World!

ở ví dụ trên thì getElementById() là phương thức trong khi innerHTML lại là thuộc tính.

childNodes và nodeValue

chúng ta cũng có thể sử dụng childNodes và nodeValue để lấy nội dung của một đối tượng.

cũng là ví dụ trên nhưng chúng ta sẽ làm như sau :

```
<html>
<body>
<p id="intro">Hello World!</p>
<script type="text/javascript">
txt=document.getElementById("intro").childNodes[0].nodeValue;
document.write("<p>The text from the intro paragraph: " + txt + "</p>");
</script>
</body>
</html>
```

Cũng chú ý luôn rằng childNodes và nodeValue là những thuộc tính.

Ở đây chúng ta chủ yếu sử dụng innerHTML mà thôi ,bởi vì nó rất hữu ích cho việc tìm hiểu cấu trúc cây của DOM cũng như là xử lý các tập tin XML.

Truy cập vào nút trong DOM

Với DOM bạn có thể truy cập bất kì nút nào trong tài liệu HTML.

Bạn có thể truy cập nút của cây bằng ba cách sau :

- sử dụng getElementById() như đã học ở bài trước
- sử dụng getElementByTagName()
- sử dụng các mối quan hệ cha,con,anh em trong cây để truy cập nút

getElementById( )

phương thức này sẽ trả về đối tượng có id định sẵn.

cú pháp :

```
node.getElementById("id");
```

ví dụ :

```
<html>
<body>
<p id="intro">Hello World!</p>
<p>This example demonstrates the <b>getElementById</b> method!</p>
<script type="text/javascript">
x=document.getElementById("intro");
document.write("<p>The text from the intro paragraph: " + x.innerHTML + "</p>");
</script>
```

```
</body>
</html>
```

Kết quả hiện thị như sau :

Hello World!

This example demonstrates the **getElementById** method!

The text from the intro paragraph: Hello World!

Chú ý : getElementById( ) không làm việc trong XML

getElementsByTagName( )

getElementsByTagName( ) trả về các nút mang tag như đã chỉ ra,trong một trang HTML có thể có nhiều nút mang tag giống nhau,chính vì vậy cho nên phương thức này có thể trả về một mảng các giá trị.

Cú pháp :

```
node.getElementsByTagName(“ tag name”);
```

trong ví dụ dưới đây phương thức này sẽ trả về một mảng các nút mang tag là <p>

```
<html>
<body>
<p>Hello World!</p>
<p>The DOM is very useful!</p>
<p>This example demonstrates the <b>getElementsByTagName</b> method.</p>
<script type="text/javascript">
x=document.getElementsByTagName("p");
document.write("Text of first paragraph: " + x[0].innerHTML);
</script>
</body>
</html>
```

Kết quả hiện thị như sau :

Hello World!

The DOM is very useful!

This example demonstrates the **getElementsByTagName** method.

Text of first paragraph: Hello World!

Ví dụ dưới đây sẽ kết hợp việc sử dụng getElementById( ) và getElementsByTagName( ) :

```
<html>
<body>
<p>Hello World!</p>
<div id="main">
<p>The DOM is very useful.</p>
<p>This example demonstrates the <b>getElementsByTagName</b> method</p>
</div>
<script type="text/javascript">
x=document.getElementById("main").getElementsByTagName("p");
document.write("First paragraph inside the div: " + x[0].innerHTML);
</script>
</body>
</html>
```

Kết quả hiện thị như sau :

Hello World!

The DOM is very useful.

This example demonstrates the **getElementsByTagName** method

First paragraph inside the div: The DOM is very useful.

Các bạn chú ý rằng khi các bạn sử dụng getElementsByTagName( ) thì kết quả có thể sẽ trả về một mảng các nút con.Để xác định số phần tử trong mảng này chúng ta sử dụng thuộc tính **length**

ví dụ :

```
<html>
<body>
<p>Hello World!</p>
<p>The DOM is very useful!</p>
<p>This example demonstrates the <b>length</b> property.</p>
<script type="text/javascript">
x=document.getElementsByTagName("p");
for (i=0;i<x.length;i++)
  {
  document.write(x[i].innerHTML);
  document.write("<br />");
```

```
  }
</script>
</body>
</html>
```

Kết quả hiện thị sẽ như sau :

Hello World!

The DOM is very useful!

This example demonstrates the **length** property.

Hello World!
The DOM is very useful!
This example demonstrates the **length** property.

Trong ví dụ chúng đã lấy tất cả các nút mang tag <p> và đưa chúng vào một một mảng x,sau đó thông qua mảng x này để hiện thị tất cả các giá trị bên trong thẻ <p>.

Sử dụng mối quan hệ trong node-tree để truy cập nút

Đây cũng là một cách để chúng ta có thể truy cập vào một nút ngoài hai cách đã nêu ra ở trên.Phương pháp này sẽ sử dụng những thuộc tính sau :

- parentNode
- firstChild
- lastChild

các bạn hãy nhìn vào đoạn HTMl sau :

```
<html>
<body>
<p>Hello World!</p>
<div>
  <p>The DOM is very useful!</p>
  <p>This example demonstrates node relationships.</p>
</div>
</body>
</html>
```

Trong đoạn HTML trên thì cặp thẻ <p> xuất hiện đầu tiên trong mã là nút con đầu tiên (firstChild) của nút <body> và nút là nút con cuối cùng (lastChild) của nút <body>,nhưng nút này lại có hai nút con cùng mang tag <p>,và fistChild của của thẻ này là cặp thẻ <p>nằm ở phía trên,lastChild là cặp thẻ <p> nằm ở phía dưới.

Ví dụ :

```
<html>
<body>
<p id="intro">Hello World!</p>
<script type="text/javascript">
x=document.getElementById("intro");
document.write(x.firstChild.nodeValue);
</script>
</body>
</html>
```

Kết quả hiện thị như sau :

Hello World!

Hello World!

Chú ý : trong DOM có hai đối tượng đặc biệt các bạn có thể truy cập bằng phương pháp khác đó là

- document.documentElement – trả về nút gốc của một tài liệu
- document.body – cho phép truy cập trực tiếp đến các tag <body>

chú ý : nhân tiện đây tôi cũng muốn nhắc các bạn một vấn đề cần lưu ý như thế này.khi truy cập vào một nút nào đó trong cây nút và bạn lấy giá trị của nó để thực hiện các phép toán thì các bạn phải chú ý đến đơn vị của giá trị đó,vấn đề này tưởng như đơn giản nhưng nếu không để ý sẽ dẫn đến một số lỗi.Trong ví dụ sau chúng ta sẽ truy cập vào nút có tên tag là và điều chỉnh vị trí của nó :

```
<html>

<head>

<title>Untitled Document</title>

<script language="javascript">

function ham()

{

  x=document.getElementById('node').style.top +50;

  document.getElementById('node').style.top=x+"px";
```

```
}
</script>
</head>
<body>
<div id="node" style="position:absolute; top:10px"> Hello  </div>
<input type="button" value="click" onclick="ham()" />
</body>
</html>
```

Khi bạn nhấn button thì vị trí của đối tượng chẳng có thay đổi gì. Tại sao vậy ? đó là bởi vì function ham() không hoạt động,cụ thể hơn đó là do phép toán :

x=document.getElementById('node').style.top +5;

không thể thực hiện được bởi **document.getElementById('node').style.top** sẽ trả về một giá trị có đơn vị là px (pixel) khác đơn vị với 5, chúng ta không thể thực hiện các phép toán trên hai giá trị khác đơn vị được (ví như chẳng thể lấy 5m cộng với 6 kg được).

Vậy để giải quyết bài toán trên ta phải làm như thế nào ? câu trả lời là các bạn đâu tiên phải chuyển **document.getElementById('node').style.top** thành số (dùng hàm parseInt()) rồi sau đó mới thực hiện phép cộng:

```
<script language="javascript">
function ham()
{
  x=parseInt(document.getElementById('node').style.top) +50;
  document.getElementById('node').style.top=x+"px";
 }
</script>
```

Sử dụng DOM để xác định thông tin node

Trong HTML DOM mỗi một nút là một đối tượng (Object).

Đối tượng thì có phương thức và thuộc tính và chúng ta có thể truy cập chúng bằng nhiều cách thông qua JavaScript.

Ba thuộc tính quan trọng chứa thông tin về nút gồm :

- nodeName
- nodeValue
- nodeType

nodeName

Thuộc tính cho biết tên của một nút.

Nó là một thuộc tính chỉ đọc, bạn không thể chỉnh sửa với nó.

nodeName của một nút cơ sở là giống như tên của <tag>.

nodeName của một thuộc tính là tên của thuộc tính đó.

nodeName của một text node luôn luôn là #text.

nodeName của một document node luôn luôn là #document.

nodeValue

Thuộc tính này trả về giá trị của một nút.

nodeValue của một nút cơ sở là không có (undefine)

nodeValue của một text node là đoạn text đó

nodeValue của một nút thuộc tính là giá trị của thuộc tính đó.

Ví dụ :

```
<html>
<body>
<p id="intro">Hello World!</p>
<script type="text/javascript">
x=document.getElementById("intro");
document.write(x.firstChild.nodeValue);
</script>
</body>
</html>
```

Kết quả hiện thị là :

Hello World!

Hello World!

nodeType

Thuộc tính nodeType trả về loại giá trị của node,nodeType có giá trị chỉ đọc mà thôi.

Các bạn có thể tham khảo bảng sau :

Element type	NodeType
Element	1
Attribute	2
Text	3
Comment	8
Document	9

Bài thực hành về DOM

Sau đây chúng ta sẽ thực hành một số kiến thức đã học,ở đây chúng ta sẽ kết hợp việc sử dụng JavaScript và DOM.

Bài 1

Ở bài thực hành này chúng ta sẽ tạo ra một trang web có chứa một button mà khi nhấn vào button đó màu nền trang web sẽ được thay đổi.

Ý tưởng giải thuật của bài tập này là chúng ta sẽ sử dụng HTML để tạo ra một button,và tạo một event click chuột vào button đó thì một đoạn mã JavaScript sẽ được kích hoạt,đoạn JavaScript này sẽ thay đổi màu nền cho trang web.

Đoạn JavaScript sẽ sử dụng mô hình DOM để truy cập vào nút thuộc tính qui định màu của trang web để tiến hành những thay đổi thích hợp.

Bài 2

Yêu cầu của bài thực hành này là các bạn hãy tạo ra một trang web có một button và một vài kí tự nào đó.khi nhấn vào button thì ngay lập tức dòng kí tự đã có sẽ biến đổi thành dòng kí tự khác.

Ý tưởng bài tập này cũng như bài tập trên chỉ có điều đoạn mã JavaScript sử dụng DOM để truy cập vào text-node và thay đổi đoạn text đó .

Bài 3

Cũng là bài tập 2 nhưng dòng văn bản sẽ bị biến đổi thành một dòng kí tự khác màu.

Bài làm

Bài 1

```
<html>
<head>
```

```
<title> hieu </title>
<script language="javascript">
function ham()
{
document.body.bgColor='red';
}
</script>
</head>
<body>
<input type="button" value="Change background color" onclick="ham();"/>
</body>
</html>
```

Bài 2

```
<html>
<head>
<title> hieu </title>
<script language="javascript">
function ham()
{
document.getElementById('text').innerHTML="I LOVE YOU";
}
</script>
</head>
<body>
<p id="text"> I Hate You </p>
<input type="button" value="click here" onclick="ham();"/>
</body>
</html>
```

Bài 3

```
<html>
<head>
<title> hieu </title>
```

```
<script language="javascript">
function ham()
{
document.getElementById('text').innerHTML="I LOVE YOU";
document.getElementById('text').style.color="red";
}
</script>
</head>
<body>
<p id="text"> I Hate You </p>
<input type="button" value="click here" onclick="ham();"/>
</body>
</html>
```

HTML DOM – event

Event là những sự kiện mà nó có thể được nhận biết bằng các ngôn ngữ lập trình (trong trường hợp này là JavaScript).

Những ví dụ về sự kiện là :

- click chuột
- submit form
- nhập kí tự

  …….

Event thường được đi kèm với các hàm và các bạn phải chú ý rằng hàm sẽ không được thực thi nếu như event chưa xảy ra.

onload và onUnLoad

onload và onUnLoad là hai sự kiện diễn ra khi người dùng mở hoặc quay lại một trang.

Hai sự kiện này thường được sử dụng để kiểm tra trình duyệt của người dùng đang sử dụng (nó là trình duyệt gì ? phiên bản bao nhiêu ?...),căn cứ vào đó để có những biến đổi phù hợp với những loại và phiên bản trình duyệt.

Ngoài ra onload và onUnLoad còn được sử dụng với cookies,nó sẽ tạo hoặc lấy cookies khi người dùng vào hoặc quay lại một trang.Chẳng hạn như khi người dùng lần đầu vào trang web của bạn,lúc đó sẽ có một hộp thoại hiện lên yêu cầu người dùng nhập tên vào,trong những lần ghé thăm khác thì

hộp thoại ấy không hiện lên nữa mà thay vào đó là một lời chào ví dụ như " xin chào Jone". Đó là một ví dụ về sử dụng cookies.

onFocus,onBlur và onChange

onFocus,onBlur,onChange là những sự kiện thường được sử dụng kết hợp với việc xác nhận những trường form.

Dưới đây là một ví dụ về việc sử dụng onChange.Ở đó hàm checkEmail() sẽ luôn được gọi mỗi khi người dùng thay đổi nội dung của trường Email:

```
E-mail: <input type="text" id="email" onchange="checkEmail()" />
```

BÀI TẬP:

Bây giờ sau khi nghiên cứu về DOM ,JavaScript và HTML chúng ta sẽ bắt đầu thực hiện một số bài luyện tập hay ho hơn một chút,những lý thuyết trên sẽ bộ trợ cho nhau giúp bạn tạo ra những trang web có tính tương tác cao với người dùng.

Bài 1:

Hãy tạo ra một trang web trong đó có một button và khi các bạn di chuột đến button đó thì vị trí của nó sẽ được thay đổi ngẫu nhiên đến vị trí khác.

Bài 2:

Hãy tạo ra một trang web(sử dụng JavaScript) có 1500 chữ " you", các chữ " you" này sẽ được đặt một cách ngẫu nhiên trên màn hình trình duyệt,và mỗi chữ cũng sẽ có một màu ngẫu nhiên nào đó.

Bài 3:

hãy tạo ra một trang web có một chiếc lá đang rơi từ góc trên màn hình rơi xuống.

Bài 4:

Cũng như bài tập 3 nhưng có điều là thêm chuyển động theo chiều ngang của chiếc lá nữa,chiếc lá khi rơi xuống nó sẽ lắc lư theo chiều gió.

LỜI GIẢI:

Bài 1:

để làm được bài tập này gợi ý các bạn là chúng ta cần tìm hiểu lại về các hàm ngẫu nhiên random trong JavaScript,sử dụng DOM để truy cập vào các thuộc tính vị trí của button và thay đổi vị trí của nó.Đoạn mã sau mang tính chất tham khảo,thú thật là đoạn mã này không hoạt động trên firefox và Crome nó chỉ hoạt động trên IE và Opera mà thôi :

```
<html>
<head>
```

```
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1" />
<title>Untitled Document</title>
<script language="javascript">
function ham()
{
  document.getElementById("nut").style.top=Math.floor(Math.random()*600);
  document.getElementById("nut").style.left=Math.floor(Math.random()*600);
  document.getElementById("nut").value="hieu";
}
</script>
</head>
<body>
<input id="nut" type="button" value="click here" style="position:relative; width:100px; height:100px; top:300px; left:300px" onmouseover="ham();" />
</body>
</html>
```

Các bạn có thể xem demo ở trong file b1-1 đính kèm với cuốn giáo trình.

Bây giờ vấn đề đặt ra là như thế này :

Rõ ràng đoạn mã trên khi các bạn chạy trên Crome và Firefox thì không hoạt động,vậy có cách nào làm bài bài tập trên mà có thể chạy trên mọi trình duyệt hay không ?

Vấn đề này nảy sinh khi chúng ta tiến hành thực hiện phép gán sau :

```
document.getElementById("nut").style.top=500;
document.getElementById("nut").style.left=500;
```

có một số người lại gán như sau :

```
document.getElementById("nut").style.top=”500px”;
document.getElementById("nut").style.left=”500px”;
```

khác biệt ở hai trường hợp trên là ở chỗ có ngoặc kép hay không có ngoặc kép.Các bạn hãy tự tìm hiểu vấn đề này(thực ra thì cách sử dụng có ngoặc kép vẫn an toàn hơn,cho nên mình khuyên bạn nên sử dụng cách này),sau đây là đoạn code khác của bài tập trên chạy được trên tất cả các trình duyệt.

```
<html>
<head>
<title>Untitled Document</title>
<script language="javascript">
```

```
function ham()
{
  x=Math.floor(Math.random()*600);
  y=Math.floor(Math.random()*600);
  document.getElementById("nut").style.top=x.toString()+"px" ;
  document.getElementById("nut").style.left=y.toString()+"px";
  document.getElementById("nut").value="hieu";
}
</script>
</head>
<body>
<input id="nut" type="button" value="click here" style="position:relative; width:100px; height:100px; top:300px; left:300px" onmouseover="ham();" />
</body>
</html>
```

Các bạn có thể xem demo đoạn code này ở file d1-2 đính kèm với cuốn giáo trình.

Bài 2:

Gợi ý bài này chúng ta sử dụng hàm ngẫu nhiên random trong JavaScript,code nguồn của bài như sau,các bạn hãy tự tìm hiểu :

```
<html>
<head>
<title>Untitled Document</title>
</head>
<body>
<script language="javascript">
var i=0;
while(i<1500)
{
 l=Math.floor(Math.random()*999999);
 document.write("<p>you</p>");
 k=document.body.lastChild;
 k.style.position="absolute";
 x=Math.floor(Math.random()*900);
 y=Math.floor(Math.random()*800);
```

```
k.style.top=y.toString()+"px";
k.style.left=x.toString()+"px";
k.style.color="#"+l.toString();
i=i+1;
}
</script>
</body>
</html>
```

Các bạn hãy xem demo ở file d2 đính kèm với cuốn giáo trình.

Bài 3:

ở bài tập này trước hết các bạn phải có một cái hình chiếc lá.Giải thuật của chúng ta là sẽ lập trình điều khiển vị trí của đối tượng img này thay đổi sao cho tạo cảm giác là chiếc lá đang rơi.

Vị trí của chiếc lá bao gồm :

- top: khoảng cách tới lề trên trình duyệt
- left: khoảng cách tới lề trái của trình duyệt.

lúc đầu chiếc lá sẽ xuất hiện với **top** là một giá trị nào đó,và cứ sau một khoảng thời gian rất nhỏ giá trị thuộc tính **top** này sẽ tăng lên một lượng, cứ như vậy sẽ tạo ra hình ảnh một chiếc lá đang rơi.

Tất nhiên để có thể điều khiển vị trí của chiếc lá liên tục như vậy thì chúng ta phải bọc đối tượng image trong một thẻ với position là absolute.

Code nguồn như sau :

```
<html >
<head>
<title>Untitled Document</title>
<script language="javascript">
var top1 = Math.random()*40;
var left1 = Math.random()*400;
document.write('<div id="anh1" style="position:absolute; top:'+top1+'px; left:'+left1+'px"><img
src="tuyet.gif" /> </div>');
function ham()
{
  var k= parseInt(document.getElementById("anh1").style.top) + 5;
  document.getElementById("anh1").style.top= k+"px";
  if(parseInt(document.getElementById("anh1").style.top)>600)
```

```
  document.getElementById("anh1").style.top=0;
  setTimeout("ham()",100);
}

</script>
</head>
<body>
<input type="button" onclick="ham();" value="click" />
</body>
</html>
```

Các bạn có thể xem demo ở b3 trong file đính kèm với tài liệu.

Bài 4:

Ở bài tập này chúng ta có thể thừa hưởng code của bài tập 3,có điều chúng ta sẽ thêm chuyển động theo chiều ngang cho chiếc lá nữa.Để chiếc lá chuyển theo hình lượn sóng thì tọa độ ngang của nó phải biến đổi theo hình sin.

Sử dụng hàm Math.sin( );

Code nguồn như sau :

```
<html >
<head>
<title>Untitled Document</title>
<script language="javascript">
var top1 = Math.random()*40;
var left1 = Math.random()*360;
document.write('<div id="anh1" style="position:absolute; top:'+top1+'px; left:'+left1+'px"><img
src="tuyet.gif" /> </div>');
function ham()
{
  var k= parseInt(document.getElementById("anh1").style.top) + 5;
  var l= Math.sin(left1)*10+200;
  left1=left1+1;
  document.getElementById("anh1").style.top= k+"px";
  document.getElementById("anh1").style.left= l+"px";
  if(parseInt(document.getElementById("anh1").style.top)>600)
  document.getElementById("anh1").style.top=0;
```

```
  setTimeout("ham()",100);
}
</script>
</head>
<body>
<input type="button" onclick="ham();" value="click" />
</body>
</html>
```

Xem Demo ở file b4.