Giáo Trình

JAVASCRIPT

Biên soạn : **Nguyễn Minh Thành**

Tp.HCM, 2010

Mục Lục

Bài 1 TỔNG QUAN VỀ JAVASCRIPT

I. Giới thiệu

Với HTML bạn đã biết cách tạo ra trang Web - tuy nhiên chỉ mới ở mức biểu diễn thông tin chứ chưa phải là các trang Web động có khả năng đáp ứng các sự kiện từ phía người dùng. Hãng Netscape đã đưa ra ngôn ngữ script có tên là LiveScript để thực hiện chức năng này. Sau đó ngôn ngữ này được đổi tên thành JavaScript để tận dụng tính đại chúng của ngôn ngữ lập trình Java. Mặc dù có những điểm tương đồng giữa Java và JavaScript, nhưng chúng vẫn là hai ngôn ngữ riêng biệt.

JavaScript là ngôn ngữ dưới dạng script có thể gắn với các file HTML. Nó không được biên dịch mà được trình duyệt diễn dịch. Không giống Java phải chuyển thành các mã dễ biên dịch, trình duyệt đọc JavaScript dưới dạng mã nguồn. Chính vì vậy bạn có thể dễ dàng học JavaScript qua ví dụ bởi vì bạn có thể thấy cách sử dụng JavaScript trên các trang Web.

JavaScript là ngôn ngữ dựa trên đối tượng, có nghĩa là bao gồm nhiều kiểu đối tượng, ví dụ đối tượng ***Math*** với tất cả các chức năng toán học. Tuy vậy JavaScript không là ngôn ngữ hướng đối tượng như C++ hay Java do không hỗ trợ các lớp hay tính thừa kế.

JavaScript có thể đáp ứng các sự kiện như tải hay loại bỏ các form. Khả năng này cho phép JavaScript trở thành một ngôn ngữ script động.

Giống với HTML và Java, JavaScript được thiết kế độc lập với hệ điều hành. Nó có thể chạy trên bất kỳ hệ điều hành nào có trình duyệt hỗ trợ JavaScript. Ngoài ra JavaScript giống Java ở khía cạnh an ninh: JavaScript không thể đọc và viết vào file của người dùng.

Các trình duyệt web như Nescape Navigator 2.0 trở đi có thể hiển thị những câu lệnh JavaScript được nhúng vào trang HTML. Khi trình duyệt yêu cầu một trang, server sẽ gửi đầy đủ nội dung của trang đó, bao gồm cả HTML và các câu lệnh JavaScript qua mạng tới client. Client sẽ đọc trang đó từ đầu đến cuối, hiển thị các kết quả của HTML và xử lý các câu lệnh JavaScript khi nào chúng xuất hiện.

Các câu lệnh JavaScript được nhúng trong một trang HTML có thể trả lời cho các sự kiện của người sử dụng như kích chuột, nhập vào một form và điều hướng trang. Ví dụ bạn có thể kiểm tra các giá trị thông tin mà người sử dụng đưa vào mà không cần đến bất cứ một quá trình truyền trên mạng nào. Trang HTML với JavaScript được nhúng sẽ kiểm tra các giá trị được đưa vào và sẽ thông báo với người sử dụng khi giá trị đưa vào là không hợp lệ.

Mục đích giáo trình là giới thiệu về ngôn ngữ lập trình JavaScript để bạn có thể viết các script vào file HTML của mình.

II. Nhúng JavaScript vào trang web

Bạn có thể nhúng JavaScript vào một File HTML theo một trong các cách sau đây:

- Sử dụng các câu lệnh và các hàm trong cặp thẻ <script>…</script>
- Sử dụng các File nguồn JavaScript
- Sử dụng một biểu thức JavaScript làm giá trị của một thuộc tính cho thẻ HTML
- Sử dụng thẻ sự kiện (event handlers) trong một thẻ HTML nào đó

Trong đó, sử dụng cặp thẻ <script>...</script> và nhúng một File nguồn JavaScript là được sử dụng nhiều hơn cả.

1. Nhúng nguồn Javascript vào trang

Script được đưa vào File HTML bằng cách sử dụng cặp thẻ <script> và <\script>. Các thẻ <script> có thể xuất hiện trong phần <head> hay <body> của File HTML. Nếu đặt trong phần <head>, nó sẽ được tải và sẵn sàng trước khi phần còn lại của văn bản được tải.

Thuộc tính duy nhất được định nghĩa hiện thời cho thẻ <script> là "language=…" dùng để xác định ngôn ngữ script được sử dụng. Có hai giá trị được định nghĩa là "JavaScript" hay "VBScript". Với chương trình viết bằng JavaScript bạn sử dụng cú pháp sau :

```
<script language="JavaScript">
// Chèn các mã Javacript vào đây
</script>
```

2. Sử dụng file nguồn javascript

Dùng phương pháp này hay hơn nhúng trực tiếp lệnh JavaScript vào trang HTML.

Cú pháp:

```
<script src="file_name.js">
</script>
```

Thuộc tính SRC có thể được định rõ bằng địa chỉ URL, các liên kết hoặc các đường dẫn tuyệt đối, VD:

```
<script src=" http://cse.com.vn/... ">
```

Các File JavaScript bên ngoài không được chứa bất kỳ thẻ HTML nào. Chúng chỉ được chứa các câu lệnh JavaScript và định nghĩa hàm. Tên File của các hàm JavaScript bên ngoài cần có đuôi .js,

3. Ẩn javascript khi trình duyệt không hỗ trợ

JavaScript là cho phép bạn ẩn các mã JavaScript trong các ghi chú của File HTML, để các trình duyệt cũ không hỗ trợ cho JavaScript có thể đọc được nó như trong VD sau đây:

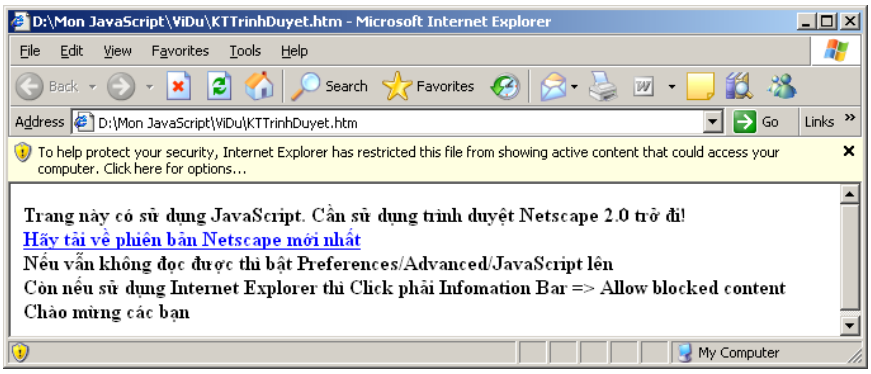
```
<SCRIPT LANGUAGE="JavaScript">
<!—
// Chèn các mã Javacript vào đây
-->
</SCRIPT>
```

Thẻ <noscript> và </noscript>

Cặp thẻ này dùng để định rõ nội dung thông báo cho người sử dụng biết trình duyệt không hỗ trợ JavaScript. Khi đó trình duyệt sẽ không hiểu thẻ <noscript> và nó bị lờ đi, còn đoạn mã nằm trong cặp thẻ này sẽ được Navigator hiển thị. Ngược lại, nếu trình duyệt có hỗ trợ JavaScript thì đoạn mã trong cặp thẻ <noscript> sẽ được bỏ qua. Tuy nhiên, điều này cũng có thể xảy ra nếu người sử dụng không sử dụng JavaScript trong trình duyệt của mình bằng cách tắt nó đi trong hộp Preferences/Advanced.

Vd: Tạo trang (KTTrinhDuyet)

```
<html><body>
<noscript><b> Trang này có sử dụng JavaScript. Cần sử dụng trình duyệt Netscape 2.0 trở đi! <br>
<a href="http://home.netscape.com/comprd/mirror/index.html">
Hãy tải về phiên bản Netscape mới nhất</a> </br>
Nếu vẫn không đọc được  thì bật Preferences/Advanced/JavaScript lên<br>
Còn nếu sử dụng Internet Explorer thì Click phải Infomation Bar => Allow blocked content<br> </noscript>
Chào mừng các bạn
</body></html>
```



III. Các lệnh cơ bản

1. Cú pháp cơ bản của lệnh

Câu lệnh của JavaScript bao gồm các hàm, các phát biểu, các toán tử và các biểu thức trên cùng một dòng và kết thúc bằng **;**

Cách gọi một phương thức của một đối tượng như sau:

```
object_name.property_name;
```

vd: **document.writeln("Chào các bạn!
");**

2. Hiển thị một dòng text

Trong hầu hết các ngôn ngữ lập trình, khả năng cơ sở là hiển thị ra màn hình một dòng text. Trong JavaScript, người lập trình cũng có thể điều khiển việc xuất ra màn hình của client một dòng text tuần tự trong File HTML. JavaScript sẽ xác định điểm mà nó sẽ xuất ra trong File HTML và dòng text kết quả sẽ được dịch như các dòng HTML khác và hiển thị trên trang.

Trong phần này, ta sẽ học lệnh xuất văn bản write() và writeln() của đối tượng document (chi tiết ở bài 5) .

Đối tượng document trong JavaScript được thiết kế sẵn hai cách thức để xuất một dòng text ra màn hình client: write() và writeln(). Cách gọi một cách thức của một đối tượng như sau:

```
document.write("Chuỗi văn bản");
```

vd: document.write("Chào các bạn");

document.writeln("Chào các bạn");

Cách thức write() xuất ra màn hình xâu Text nhưng không xuống dòng, còn cách thức writeln() sau khi viết xong dòng Text tự động xuống dòng. Hai cách thức này đều cho phép xuất ra thẻ HTML.

Ghi chú: có thể dùng "+" để ghép nhiều chuỗi ký tự.

Cho phép dùng các kí tự đặc biệt trong chuỗi:

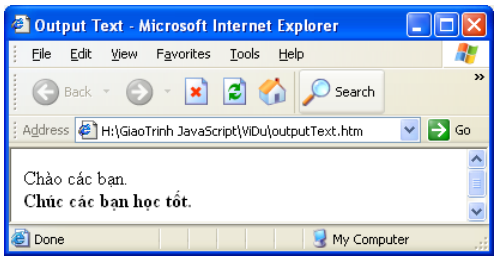
\n : Xuống dòng

\t : Tab

Khi có dùng các ký tự đặc biệt hoặc khoảng trắng thì phải đặt khối JavaScript trong cặp thẻ <pre> . . </pre>

vd: Tạo trang (OutputText.htm) dùng cách thức write() xuất ra thẻ HTML.

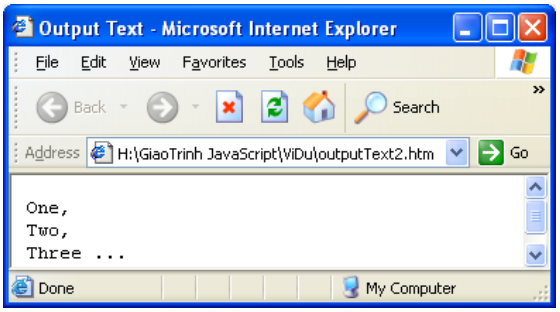
```
<body>
  Chào các bạn.<BR>
```



```
 <script language="javascript">
 <!--
      document.write("<B>Chúc các bạn học tốt.</B>");
 -->
 </script>
</body>
```

vd: Tạo trang (OutputText2.htm) phân biệt sự khác nhau của write() và writeln():

```
<body>
<pre>
<script language="javascript">
<!--
    document.writeln("One,");
    document.write("Two,\n");
    document.write("Three ");
    document.write("...");
-->
</script>
</pre>
<body>
```



vd: Tạo trang (OutputText3.htm) xuất các thẻ html từ JavaScript:

```
<body>
<script language="JavaScript">
<!--
document.write('<img src="logo.jpg">');
document.write("<br><h3>Welcome To Javascript Language</h3>");
-->
</script>
</body>
```

3. Hiển thị hộp thoại thông báo – alert()

JavaScript hỗ trợ khả năng cho phép người lập trình tạo ra một hộp hội thoại thông báo. Cách đơn giản để làm việc đó là sử dụng lệnh alert(). Để sử dụng được cách thức này, bạn phải đưa vào một dòng văn bản

Cú pháp:

```
alert("Câu thông báo");
```

Khi đó File sẽ chờ cho đến khi người sử dụng nhấn vào nút OK rồi mới tiếp tục thực hiện.

Thông thường, cách thức alert() được sử dụng trong các trường hợp thông báo:

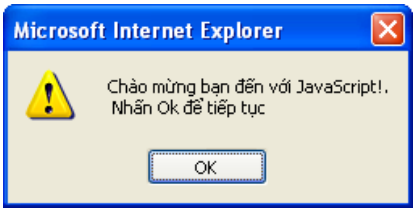
Thông tin đưa và form không hợp lệ

Kết quả sau khi tính toán không hợp lệ

Khi dịch vụ chưa sẵn sàng để truy nhập dữ liệu, hay khi có lỗi xảy ra…

vd: Tạo trang (Thongbao.htm)

```
<body>
 <script language="javascript">
 <!--
 alert("Chào mừng bạn đến với JavaScript!. \n Nhấn Ok để tiếp tục");
 -->
 </script>
 Chúc bạn thành công!!!
</body>
```



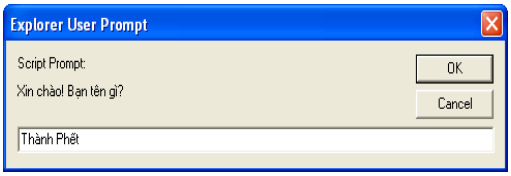
4. Giao tiếp với người sử dụng – Lệnh Prompt()

Lệnh alert() mới chỉ cho phép thông báo với người sử dụng chứ chưa thực sự giao tiếp với người sử dụng. JavaScript cung cấp lệnh **prompt()** để giao tiếp với người sử dụng là tạo ra một hộp hội thoại gồm một dòng thông báo, một trường nhập dữ liệu, một nút OK và một nút Cancel. Người sử dụng có thể nhập vào trường đó rồi kích vào OK. Khi đó, ta có thể xử lý dữ liệu do người sử dụng vừa đưa vào.

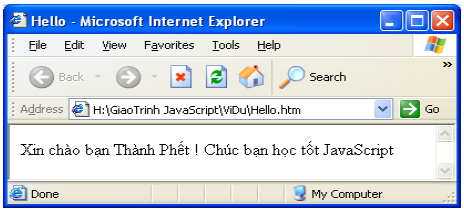
Cú pháp:

```
prompt("Câu thông báo","nội dung mặc định");
```

vd: Tạo trang (Hello1.htm) hiện thị hộp thoại hỏi tên người dùng và sau đó sẽ hiển thị một thông báo chào tên mới đưa vào.

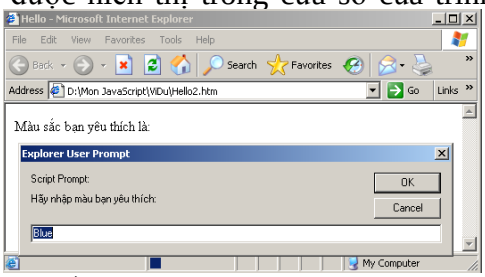



```
<body>
 <script language="JavaScript">
 <!--
 var name = prompt("Xin chào! Bạn tên gì?","");
 document.write("Xin chào bạn " + name + " ! Chúc bạn học tốt JavaScript ");
 -->
 </script>
</body>
```



Ví dụ này hiển thị dấu nhắc nhập vào tên với phương thức window.prompt. Giá trị đạt được sẽ được ghi trong biến có tên là name.

Biến name được kết hợp với các chuỗi khác và được hiển thị trong cửa sổ của trình duyệt nhờ phương thức document.write.

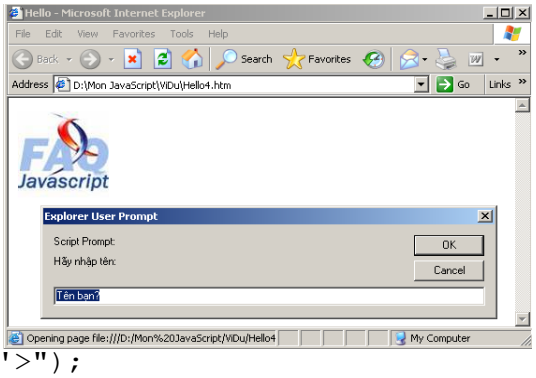


vd: Tạo trang (Hello2.htm) như sau.

```
<body>
 <script language="Javascript">
 <!--
 document.write("Màu sắc bạn yêu thích là:");
 document.writeln(prompt("Hãy nhập màu bạn yêu thích:","Blue"));
 -->
 </script>
</body>
```

vd: Tạo trang (Hello3.htm) như sau.

```
<body>
<script language="JavaScript">
<!--
document.write("<img src='logo.jpg'> <BR>");
document.write("Xin chào bạn <B>");
document.writeln(prompt("Hãy nhập tên:","Tên bạn?"));
document.write("</B> Đã đến với Ngôn ngữ JavaScript ");
-->
</script>
</body>
```



vd: Tạo trang (Hello4.htm) như sau.

```
<body>
 <script language="JavaScript">
 <!--
 document.write("<img src='logo.jpg'>");
 document.write("\n<H3>Chào ban " + prompt("Hãy nhập tên:","Tên bạn?") + " đã đến với JavaScript</H3>");
 -->
 </script> </body>
```

5. Hỏi đáp người sử dụng

Lệnh confirm cho phép xuấ hiện hộp thoại thông báo tùy theo hành động quyết định click chuột của người sử dụng mà lệnh tiếp theo sẽ được thực hiện.

Lệnh **confirm()** tạo ra một hộp hội thoại gồm một dòng thông báo, một nút OK và một nút Cancel. Người sử dụng có thể click vào OK. Khi đó, ta có thể xử lý thực hiện hành động theo yêu cầu, ngược lại khi Click vào Cancel sẽ bỏ đóng hộp thoại thông bao.

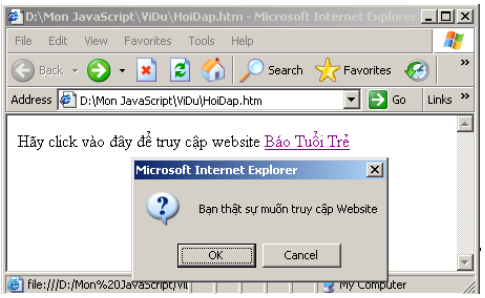
Thường sử dụng trong các trường hỏi đáp, xác nhận quyết định xử lý thông tin từ phía người dùng.

Cú pháp:

confirm("Câu thông báo hỏi ?");

vd: Tạo trang (HoiDap.htm) như sau.

```
<script>
function Hoidap(){
 question = confirm("Bạn thật sự muốn truy cập Website")
 if (question !="0"){
      top.location = "http://www.tuoitre.com.vn/"
 }
}
</script>
Hãy click vào đây để truy cập website :<a href="" onClick="Hoidap(); return false;">Báo Tuổi Trẻ </a>
```



Bài 2 NGÔN NGỮ JAVASCRIPT

I. Biến

Cũng như các ngôn ngữ lập trình khác javascript dùng biến để lưu trữ các giá trị nhập vào, các giá trị tính toán . . . nói cách khác biến là vùng nhớ sử dụng để lưu trữ các giá trị khác nhau trong quá trình chương trình hoạt động.

Mỗi biến có một tên, Tên biến trong JavaScript phải bắt đầu bằng chữ hay dấu gạch dưới. Các chữ số không được sử dụng ở đầu tên biến nhưng có thể sử dụng sau ký tự đầu tiên.

Phạm vi của biến có thể là một trong hai kiểu sau:

Biến toàn cục: Có thể được truy cập từ bất kỳ đâu trong ứng dụng. được khai báo:

x = 0;

Biến cục bộ: Chỉ được truy cập trong phạm vi chương trình mà nó khai báo. Biến cục bộ được khai báo trong một hàm với từ khoá var, và được định nghĩa trong một hàm (function) nào đó:

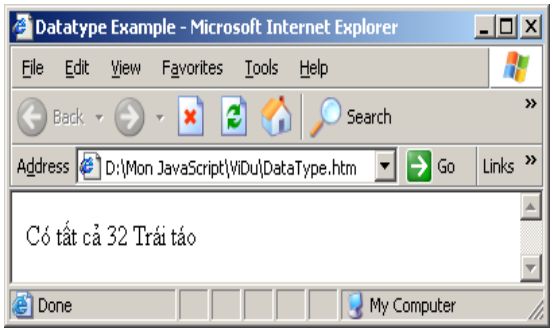
var x = 0;

II. Kiểu dữ liệu

Khác với C++ hay Java, JavaScript là ngôn ngữ có tính định kiểu thấp. Điều này có nghĩa là không phải chỉ ra kiểu dữ liệu khi khai báo biến. Kiểu dữ liệu được tự động chuyển thành kiểu phù hợp khi cần thiết.

vd : Tạo trang (DataType.htm) như sau

```
<HTML>
<HEAD>
<TITLE> Datatype Example </TITLE>
</HEAD>
<BODY>
<SCRIPT LANGUAGE= "JavaScript">
var a="Trái táo";
var n=12;
n = n + 20;
var tb ="Có tất cả " + n + " " + a;
document.write(tb);
</SCRIPT>
</BODY>
</HTML>
```



Trình diễn dịch JavaScript sẽ xem biến n có kiểu nguyên khi cộng với 20 và khi kết hợp với biến tb sẽ chuyển thành kiểu chuỗi.

Trong JavaScript, có bốn kiểu dữ liệu sau đây: kiểu số nguyên, kiểu dấu phẩy động, kiểu logic và kiểu chuỗi.

1. Kiểu số nguyên (Integer)

Số nguyên có thể được biểu diễn theo ba cách:

- Hệ cơ số 10 (hệ thập phân) - có thể biểu diễn số nguyên theo cơ số 10, chú ý rằng chữ số đầu tiên phải khác 0.(vd: 450)

- Hệ cơ số 8 (hệ bát phân) - số nguyên có thể biểu diễn dưới dạng bát phân với chữ số đầu tiên là số 0. (vd: 056)
- Hệ cơ số 16 (hệ thập lục phân) - số nguyên có thể biểu diễn dưới dạng thập lục phân với hai chữ số đầu tiên là 0x. (vd: 0x5F)

2. Kiểu dấu phẩy động (Float Point)

Một biến có kiểu dấu phẩy động có 4 thành phần sau:

- Phần nguyên thập phân.
- Dấu chấm thập phân (.).
- Phần dư.
- Phần mũ.

Để phân biệt kiểu dấu phẩy động với kiểu số nguyên, phải có ít nhất một chữ số theo sau dấu chấm hay E.

vd: 9.87 -0.85E4 9.87E14 0.98E-3

3. Kiểu Logic (Boolean)

Kiểu logic được sử dụng để chỉ hai điều kiện : đúng hoặc sai. Miền giá trị của kiểu này chỉ có hai giá trị

true

false

4. Kiểu chuỗi (String)

Một biến kiểu chuỗi được biểu diễn bởi không hay nhiều ký tự được đặt trong cặp dấu " ... " hay '... '.

vd: “The dog ran up the tree”

‘The dog barked’

“100”

Bảng kí tự loại trừ "\"			
\'	Dấu nháy đơn	\e	Esc
\"	Dấu nháy kép	\\	Dấu \
\t	Tab	\XXX	XXX mã thập phân của kí tự Latin
\n	Xuống dòng	\xXX	XX mã thập lục phân của kí tự Latin
\b	Quay lùi (Backspace)	\uXXXX	XXXX mã thập lục phân của kí tự Unicode

5. null và undefined

null là giá trị đặc biệt để chỉ không có giá trị, có nghĩa là một giá trị rỗng.

undefined là giá trị để chỉ một giá trị chưa được định nghĩa trong Javascript.

6. Lệnh typeof

Để xác định được kiểu giữ liệu của một biến, hay một giá trị nào đó, ta sử dụng phương thức (hàm) *typeof*, hàm sẽ trả về kiểu dữ liệu của giá trị đó :

```
typeof(value)
```

7. Chuyển đổi kiểu dữ liệu

Để chuyển đổi giữa các kiểu dữ liệu cơ bản, Javascript cung cấp 3 lệnh :

- `String()` : chuyển đổi các kiểu dữ liệu về kiểu chuỗi
- `Number()` : chuyển đổi kiểu chuỗi thành số
- `Boolean()` : chuyển đổi các kiểu dữ liệu về kiểu boolean

III. Lệnh và khối lệnh

Các câu lệnh trong JavaScript kết thúc bằng một dấu chấm phẩy (;).

Một khối lệnh là đoạn chương trình gồm hai lệnh trở lên và được bắt đầu bằng dấu mở ngoặc nhọn ({) và kết thúc bằng dấu đóng ngoặc nhọc (}).

Bên trong một khối lệnh có thể chứa một hay nhiều lệnh hoặc chứa các khối lệnh khác.

```
{ // khối 1
 { // khối 2
       lệnh 2.1
       lệnh 2.2
       …
 } // kết thúc khối lệnh 2
 lệnh 1.1
 lệnh 1.2
} // kết thúc khối lệnh 1
```

IV. Toán tử & biểu thức

1. Định nghĩa và phân loại biểu thức

Tập hợp các biến và các toán tử nhằm đánh giá một giá trị nào đó được gọi là một biểu thức (expression). Có ba kiểu biểu thức trong JavaScript:

- Số học: nhằm để lượng giá giá trị số. vd (3+4)+(84.5/3) được đánh giá bằng 197.1666666667.
- Chuỗi: nhằm để đánh giá chuỗi. vd "The dog barked" + barktone + "!" là The dog barked ferociously!.
- Logic: nhằm đánh giá giá trị logic. vd temp>32 có thể nhận giá trị sai.

Ngòai ra JavaScript cũng hỗ trợ biểu thức điều kiện, cú pháp như sau:

```
(condition) ? valTrue : valFalse
```

Nếu điều kiện condition được đánh giá là đúng, biểu thức nhận giá trị valTrue, ngược lại nhận giá trị valFalse. VD:

```
state = (temp>32) ? "liquid" : "solid"
```

Trong vd này biến state được gán giá trị "liquid" nếu giá trị của biến temp lớn hơn 32; trong trường hợp ngược lại nó nhận giá trị "solid".

2. Các toán tử

Toán tử được sử dụng để thực hiện một phép toán nào đó trên dữ liệu. Một toán tử có thể trả lại một giá trị kiểu số, kiểu chuỗi hay kiểu logic. Các toán tử trong JavaScript có thể được nhóm thành các loại sau đây: gán, so sánh, số học, chuỗi, logic và logic bitwise.

a. Gán

Toán tử gán là dấu bằng (=) nhằm thực hiện việc gán giá trị của toán hạng bên phải cho toán hạng bên trái. Bên cạnh đó JavaScript còn hỗ trợ một số kiểu toán tử gán rút gọn.

Kiểu gán thông thường	Kiểu gán rút gọn
x = 5	

x = x + y	x + = y
x = x - y	x - = y
x = x * y	x * = y
x = x / y	x / = y
x = x % y	x % = y

b. So sánh

Người ta sử dụng toán tử so sánh để so sánh hai toán hạng và trả lại giá trị đúng hay sai phụ thuộc vào kết quả so sánh. Sau đây là một số toán tử so sánh trong JavaScript:

==	Trả lại giá trị đúng nếu toán hạng bên trái bằng toán hạng bên phải
!=	Trả lại giá trị đúng nếu toán hạng bên trái khác toán hạng bên phải
>	Trả lại giá trị đúng nếu toán hạng bên trái lớn hơn toán hạng bên phải
>=	Trả lại giá trị đúng nếu toán hạng bên trái lớn hơn hoặc bằng toán hạng bên phải
<	Trả lại giá trị đúng nếu toán hạng bên trái nhỏ hơn toán hạng bên phải
<=	Trả lại giá trị đúng nếu toán hạng bên trái nhỏ hơn hoặc bằng toán hạng bên phải

c. Số học

Bên cạnh các toán tử cộng (+), trừ (-), nhân (*), chia (/) thông thường, JavaScript còn hỗ trợ các toán tử sau đây:

`var1 % var2`	Toán tử phần dư, trả lại phần dư khi chia var1 cho var2
`-`	Toán tử phủ định, có giá trị phủ định toán hạng
`var++`	Toán tử này tăng var lên 1 (có thể biểu diễn là ++var)
`var--`	Toán tử này giảm var đi 1 (có thể biểu diễn là --var)

Chú ý: Nếu bạn gán giá trị của toán tử ++ hay -- vào một biến, như y= x++, có thể có các kết quả khác nhau phụ thuộc vào vị trí xuất hiện trước hay sau của ++ hay -- với tên biến (là x trong trường hợp này). Nếu ++ đứng trước x, x sẽ được tăng hoặc giảm trước khi giá trị x được gán cho y. Nếu ++ hay -- đứng sau x, giá trị của x được gán cho y trước khi nó được tăng hay giảm.

d. Chuỗi : Khi được sử dụng với chuỗi, toán tử + được coi là kết hợp hai chuỗi,

vd: "abc" + "xyz" được "abcxyz"

e. Logic : JavaScript hỗ trợ các toán tử logic sau đây:

`expr1 && expr2`	Là toán tử logic AND, trả lại giá trị đúng nếu cả expr1 và expr2 cùng đúng.
`expr1 || expr2`	Là toán tử logic OR, trả lại giá trị đúng nếu ít nhất một trong hai expr1 và expr2 đúng.
`! expr`	Là toán tử logic NOT phủ định giá trị của expr.

f. Bitwise

Với các toán tử thao tác trên bit, đầu tiên giá trị được chuyển dưới dạng số nguyên 32 bit, sau đó lần lượt thực hiện các phép toán trên từng bit.

`&`	Toán tử bitwise AND, trả lại giá trị 1 nếu cả hai bit cùng là 1.
`|`	Toán tử bitwise OR, trả lại giá trị 1 nếu một trong hai bit là 1.
`^`	Toán tử bitwise XOR, trả lại giá trị 1 nếu hai bit có giá trị khác nhau

Ngoài ra còn có một số toán tử dịch chuyển bitwise. Giá trị được chuyển thành số nguyên 32 bit trước khi dịch chuyển. Sau khi dịch chuyển, giá trị lại được chuyển thành kiểu của toán hạng bên trái. Sau đây là các toán tử dịch chuyển:

<<	Toán tử dịch trái. Dịch chuyển toán hạng trái sang trái một số lượng bit bằng toán hạng phải. Các bit bị chuyển sang trái bị mất và 0 thay vào phía bên phải. VD: 4<<2 trở thành 16 (số nhị phân 100 trở thành số nhị phân 10000)
>>	Toán tử dịch phải. Dịch chuyển toán hạng trái sang phải một số lượng bit bằng toán hạng phải. Các bit bị chuyển sang phải bị mất và dấu của toán hạng bên trái được giữ nguyên. VD: 16>>2 trở thành 4 (số nhị phân 10000 trở thành số nhị phân 100)
>>>	Toán tử dịch phải có chèn 0. Dịch chuyển toán hạng trái sang phải một số lượng bit bằng toán hạng phải. Bit dấu được dịch chuyển từ trái (giống >>). Những bit được dịch sang phải bị xoá đi. VD: -8>>>2 trở thành 1073741822 (bởi các bit dấu đã trở thành một phần của số). Tất nhiên với số dương kết quả của toán tử >> và >>> là giống nhau.

Có một số toán tử dịch chuyển bitwise rút gọn:

Kiểu bitwise thông thường	Kiểu bitwise rút gọn
x = x << y	x << = y
x = x >> y	x - >> y
x = x >>> y	x >>> = y
x = x & y	x & = y
x = x ^ y	x ^ = y
x = x | y	x | = y

V. Cấu trúc lập trình

Có thể chia các cấu trúc lập trình của JavaScript thành ba nhóm sau:

- Cấu trúc rẽ nhánh (Điều kiện).
- Cấu trúc lặp.

1. Cấu trúc lập trình rẽ nhánh

Câu lệnh này cho phép bạn kiểm tra điều kiện và thực hiện một nhóm lệnh nào đấy dựa trên kết quả của điều kiện vừa kiểm tra. Nhóm lệnh sau else không bắt buộc phải có, nó cho phép chỉ ra nhóm lệnh phải thực hiện nếu điều kiện là sai.

Cú pháp:

```
if ( <điều kiện> ) {
   //Các câu lệnh với điều kiện đúng
}else{
   //Các câu lệnh với điều kiện sai
}
```

vd:

```
if (x==10){
document.write("x bằng 10, đặt lại x bằng 0.");
x = 0;
}else  document.write("x không bằng 10.");
```

Chú ý: Ký tự { và } được sử dụng để nhóm khối mã lệnh.

vd: Tạo trang (CauTrucDK1.htm), sử dụng lệnh confirm() với phát biểu if

```
<script language="javascript">
   var question="What is 10+10 ?";
```

```
        var answer=20;
        var correct="<IMG SRC='vui.gif'>";
        var incorrect="<IMG SRC='buon.gif'>";
        var response=prompt(question,"0");
        if (response != answer) {
             if (confirm("Wrong ! press OK for a second change"))
                  response=prompt(question,"0");
             }
        var output = (response ==answer ) ?
        correct:incorrect;
        document.write(output);
    </script>
```

vd: Tạo trang (CauTrucDK1.htm) Sử dụng lệnh confirm() với phát biểu if

```
<script language="javascript">
var question="What is 10+10 ?";
var answer=20;
var correct="<IMG SRC='vui.gif'>";
var incorrect="<IMG SRC='buon.gif'>";
var response=prompt(question,"0");
if (response != answer) {
     if (confirm("Wrong ! press OK for a second change"))
          response=prompt(question,"0");
     }else {
      if (confirm("Correct ! press OK for a second question"))
       {        question="What is 10*10";
                answer=100;
                response=prompt(question,"0");  }
     }
     var output = (response ==answer ) ? correct:incorrect ;
     document.write(output);
</script>
```

2. Cấu trúc lặp

Câu lệnh lặp thể hiện việc lặp đi lặp lại một đoạn mã cho đến khi biểu thức điều kiện được đánh giá là đúng. JavaScipt cung cấp hai kiểu câu lệnh lặp:

- for
- while

a. Vòng lặp for

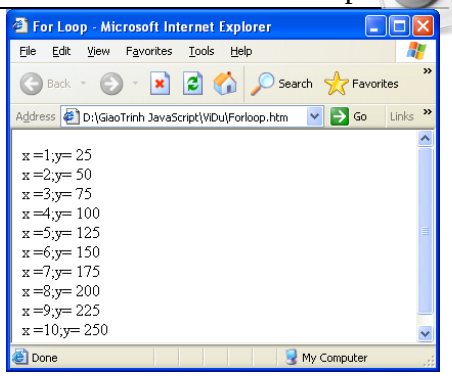
Vòng lặp for thiết lập một biểu thức khởi đầu - initExpr, sau đó lặp một đoạn mã cho đến khi biểu thức <điều kiện> được đánh giá là đúng. Sau khi kết thúc mỗi vòng lặp, biểu thức incrExpr được đánh giá lại.

Cú pháp:

```
for (initExpr; <điều kiện>; incrExpr){
//Các lệnh được thực hiện trong khi lặp
}
```

vd: Tạo trang (ForLoop.htm) như sau

```
for (x=1; x<=10 ; x++) {
  y=x*25;
  document.write("x ="+ x +";y= "+ y +
"<BR>");
}
```



b. Vòng lặp While

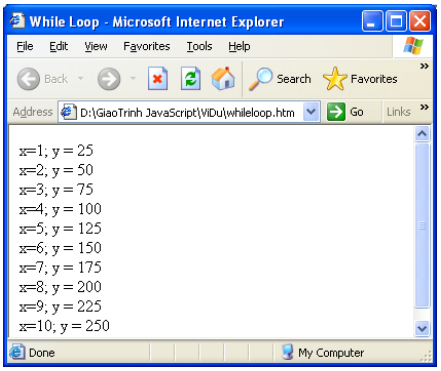
Vòng lặp while lặp khối lệnh chừng nào <điều kiện> còn được đánh giá là đúng

Cú pháp:

```
while (<điều kiện>){
  //Các câu lệnh thực hiện trong khi lặp
}
```

vd: Tạo trang (WhileLoop.htm) như sau

```
x=1;
while (x<=10){
  y=x*25;
  document.write("x="+x +"; y = "+ y +
"<BR>");
  x++;
} //Kết quả của VD này giống như VD trước.
```



c. Lệnh Break

Câu lệnh break dùng để kết thúc việc thực hiện của vòng lặp for hay while. Chương trình được tiếp tục thực hiện tại câu lệnh ngay sau chỗ kết thúc của vòng lặp.

Cú pháp: ***break;***

vd: Đoạn mã sau lặp cho đến khi x lớn hơn hoặc bằng 100. Tuy nhiên nếu giá trị x đưa vào vòng lặp nhỏ hơn 50, vòng lặp sẽ kết thúc

```
while (x<100){
  if (x<50) break;
  x++;}
```

d. Lệnh Continue

Lệnh continue giống lệnh break nhưng khác ở chỗ việc lặp được kết thúc và bắt đầu từ đầu vòng lặp. Đối với vòng lặp while, lệnh continue điều khiển quay lại <điều kiện>; với for, lệnh continue điều khiển quay lại incrExpr.

Cú pháp: ***continue;***

vd: Đoạn mã sau tăng x từ 0 lên 5, nhảy lên 8 và tiếp tục tăng lên 10

```
x=0;
while (x<=10) {
  document.write("Giá trị của x là:"+ x+"<BR>");
  if (x=5){  x=8;  continue;  }
  x++; }
```

VI. Mảng – Array

Mảng là một tập hợp các thành phần giống nhau (vd mảng các số nguyên). Mỗi phần tử của mảng được truy xuất bởi một chỉ số (index). Có hay loại chỉ số : số nguyên không âm và chuỗi.

Để định nghĩa một mảng, ta sử dụng các phương thức sau :

- Định nghĩa một mảng không chỉ định kích thước

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

- Định nghĩa một mảng có chỉ định kích thước

```
var array_name = new Array(100);
```

- Định nghĩa mảng với các giá trị khởi tạo

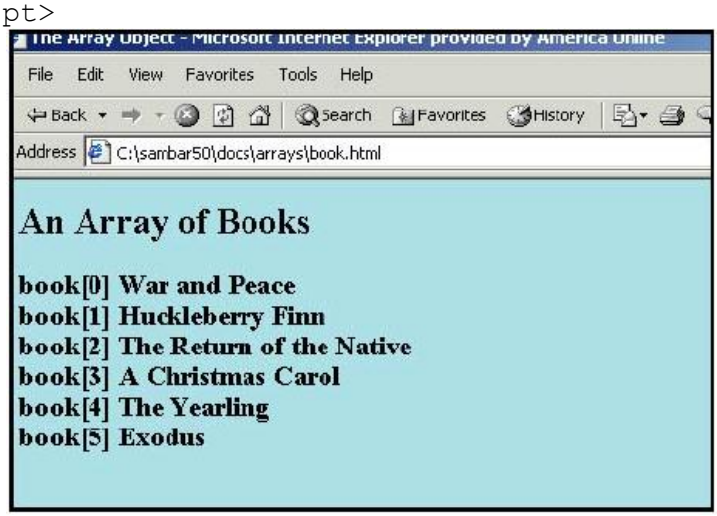
```
var array_name = new Array("red","green","blue");
```

Ghi chú : mảng là một đối tượng được định nghĩa sẵn trong Javascript (chi tiết xem bài 6), do đó khi tạo đối tượng phải sử dụng từ khoá *new*.

Ta có thể xử dụng vòng lặp để duyệt qua các phần tử trong mảng khi số lượng phần tử quá nhiều. (xem ví dụ sau)

Vd : Tạo trang web xuất mảng các loại sách

```
<script language="JavaScript">
   var book = new Array(6);   // Create an Array object
   book[0] = "War and Peace"; // Assign values to its
elements
   book[1] = "Huckleberry Finn";
   book[2] = "The Return of the Native";
   book[3] = "A Christmas Carol";
   book[4] = "The Yearling";
   book[5] = "Exodus";
       document.write("<h3>");
   for(i=0;i<book.length;i++){
    document.write("book[" + i + "] "+ book[i]  + "<br>");
   }
</script>
```



Các thuộc tính và phương thức của mảng :

Thuộc tính / phương thức	Mô tả
length	Số phần tử của mảng

prototype	Mở rộng định nghĩa thuộc tính / phương thức cho mảng
concat()	Nối các phần tử của mảng vào một mảng khác
join()	Ghép các phần tử của mảng cách nhau bởi dấu phẩy
pop()	Xoá và trả về phần tử cuối mảng
push()	Thêm các phần tử vào cuối mảng
reverse()	Đảo ngược thứ tự của các phần tử trong mảng
shift()	Xoá và trả về phần tử đầu tiên của mảng
slice()	Tạo một mảng mới từ những phần tử của 1 mảng đã tồn tại
sort()	Sắp xếp mảng
unshift()	Thêm các phần tử vào đầu mảng
toString()	Trả về chuỗi thể hiện các phần tử trong mảng

Bài 3 HÀM - FUNCTION

I. Giới thiệu

Trong kỹ thuật lập trình các lập trình viên thường sử dụng hàm để thực hiện một đoạn chương trình nào đó để thực hiện một công việc nào đó. Trong Javascript có các hàm được xây dựng sẵn để giúp bạn thực hiện một chức năng ví dụ như hàm alert(), document.write(), parseInt() và bạn cũng có thể định nghĩa ra các hàm khác của mình để thực hiện một công việc nào đó.

Các hàm có thể có một hay nhiều tham số truyền vào và một giá trị trả về. Bởi vì JavaScript là ngôn ngữ có tính định kiểu thấp nên không cần định nghĩa kiểu tham số và giá trị trả về của hàm. Hàm có thể là thuộc tính của một đối tượng, trong trường hợp này nó được xem như là phương thức của đối tượng đó.

II. Định nghĩa hàm

Lệnh function được sử dụng để tạo ra hàm trong JavaScript.

Cú pháp

```
function fnName([param1],[param2],...,[paramN]){
//function statement
}
```

Truyền tham số:

VD:
```
function printName(name) {
    document.write("<HR>tên của bạn là: <B><I>");
    document.write(name);
    document.write("</B></I><HR>");
}
```

- Khi gọi hàm printName() với lệnh là : printName("Bob"); Thì hàm printName() được thi hành giá trị của name là "Bob"
- Nếu gọi hàm printName() với đối số là một biến
  var user = "John";
  printName(user);
  Khi đó name là "John".
- Nếu bạn muốn thay đổi giá trị của name bạn có thể làm như sau :
  name = "Mr. " + name;

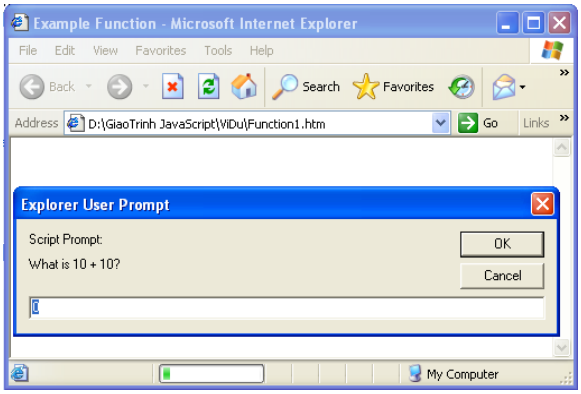
Trả về các giá trị:

Dùng return để trả về giá trị của biến.

```
function luythua3(n) {
var luythua3 = n * n * n;
return luythua3;
}
```

VD Tạo trang (Function1.htm)

```
<HTML>
<HEAD>
```



```
<TITLE>Example Function</TITLE>
<SCRIPT LANGUAGE="JavaScript">
function testQuestion(question){
      var answer=eval(question);
      var output="What is " + question + "?";
      var correct="<IMG SRC='vui.gif'>";
      var incorrect="<IMG SRC='buon.gif'>";
      var response=prompt(output,"0");
      return(response == answer)?correct:incorrect;
}
</SCRIPT>
</HEAD>
<BODY>
<SCRIPT LANGUAGE="JavaScript">
var result=testQuestion("10 + 10");
document.write(result);
</SCRIPT>
</BODY> </HTML>
```

Ghi chú: Hàm eval dùng chuyển đổi giá trị chuỗi số thành giá trị số

eval("10*10") trả về giá trị là 100

Hàm gọi lại hàm:

Ta có thể gọi lại một hàm đã định nghĩa bằng cách gọi lại tên hàm và truyền các tham số trong hai dấu ngoặc đơn ngày trong cặp thẻ <script>

```
<script language="javascript">
      Function_name(argument…);
</script>
```

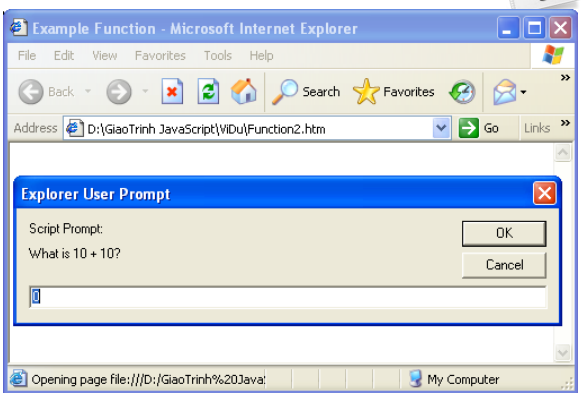
Hoặc ta cũng có thể gọi hàm tại các sự kiện của các thành tố HTML, hay trên một liên kết để đáp ứng lại các yêu cầu của người dùng.

```
<a href="javascript:function_name(argument…);" />
<input type="button" value="Click here" onclick=
"function_name(argument…);" />
```

VD Tạo trang (Function2.htm)

```
<html>
<head>
<title>example function</title>
<script language="JavaScript">
      function testQuestion(question) {
            var answer=eval(question);
            var output="What is " + question + "?";
            var correct="<IMG SRC='vui.gif'>";
            var incorrect="<IMG SRC='incorrect.gif'>";
            var response=prompt(output,"0");
            return (response == answer) ? correct :
      testQuestion(question);
}
```

```
        </script>
        </head>
        <body>
        <script
    language="JavaScript">
    var  result=testQuestion("10  +
10");
    document.write(result);
        </script>
        </body> </html>
```



III. Các hàm có sẵn

JavaScript có một số hàm có sẵn, gắn trực tiếp vào chính ngôn ngữ và không nằm trong một đối tượng nào:

- eval
- parseInt
- parseFloat

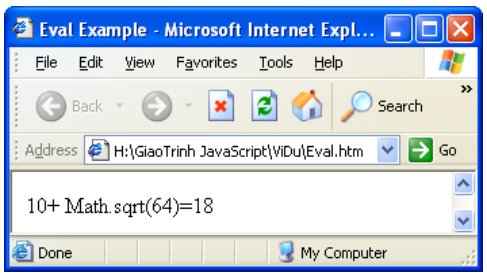
1. Hàm eval

Hàm này được sử dụng để đánh giá các biểu thức hay lệnh. Biểu thức, lệnh hay các đối tượng của thuộc tính đều có thể được đánh giá. Đặc biệt hết sức hữu ích khi đánh giá các biểu thức do người dùng đưa vào (ngược lại có thể đánh giá trực tiếp).

Cú pháp: return val=eval (bất kỳ biểu thức hay lệnh hợp lệ trong Java)

VD: Tạo trang (Eval.htm)

```
<html>
 <head>
<title>eval example </title>
<script language= "JavaScript">
var string="10+
Math.sqrt(64)";
document.write(string+ "="+
eval(string));
</script>
</head>
<body> </body></html>
```



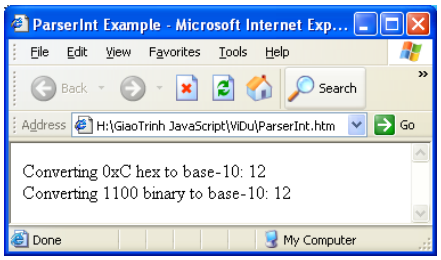
2. Hàm parseInt

Hàm này chuyển một chuỗi số thành số nguyên với cơ số là tham số thứ hai (tham số này không bắt buộc). Hàm này thường được sử dụng để chuyển các số nguyên sang cơ số 10 và đảm bảo rằng các dữ liệu được nhập dưới dạng ký tự được chuyển thành số trước khi tính toán. Trong trường hợp dữ liệu vào không hợp lệ, hàm parseInt sẽ đọc và chuyển dạng chuỗi đến vị trí nó tìm thấy ký tự không phải là số. Ngoài ra hàm này còn cắt dấu phẩy động.

Cú pháp:

```
parseInt (string [, radix])
```

VD: Tạo trang (ParserInt.htm)




```
<html>
 <head>
<title>parserint example
</title>
<body>
<script language= "JavaScript">
document.write("Converting 0xC hex to base-10: " +
parseInt(0xC,10) + "<br/>");
document.write("Converting 1100 binary to base-10: " +
parseInt(1100,2) + "<br/>");
</script>
</body></html>
```

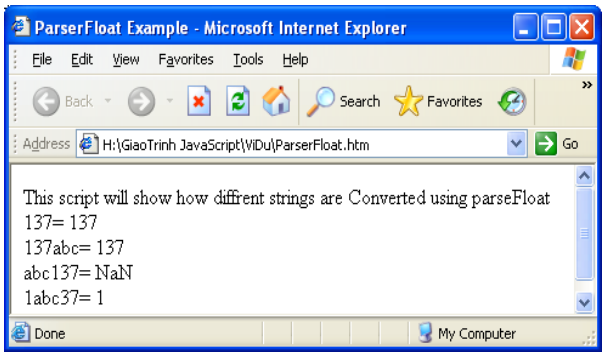
3. Hàm parseFloat

Hàm này giống hàm parseInt nhưng nó chuyển chuỗi thành số biểu diễn dưới dạng dấu phẩy động.

Cú pháp: parseFloat (string)

VD: Tạo trang (ParserFloat.htm)

```
<body>
<script language= "JavaScript">
document.write("This script will show how diffrent strings
are ");
document.write("Converted using parseFloat<BR>");
document.write("137= " + parseFloat("137") + "<BR>");
document.write("137abc= " + parseFloat("137abc") + "<BR>");
document.write("abc137= " + parseFloat("abc137") + "<BR>");
document.write("1abc37= " + parseFloat("1abc37") + "<BR>");
</script></body>
```



Bài 4 ĐỐI TƯỢNG VÀ SỰ KIỆN

I. Giới thiệu

Đối tượng là một loại biến phức hợp có thể chứa nhiều thành phần dữ liệu và phương thức (hàm). Đối tượng thường đại diện cho một thành phần, một loại đối tượng thực tế (hay trừu tượng) bất kì nào đó.

Ví dụ đối tượng person chỉ người, đối tượng animal chỉ thú vật, đối tượng window chỉ của sổ trình duyệt…

II. Các câu lệnh thao tác lên đối tượng

JavaScript cũng là một ngôn ngữ làm việc trên đối tượng, do đó nó có một số câu lệnh làm việc với các đối tượng.

1. Định nghĩa đối tượng

Javascript là một ngôn ngữ định tính thấp nên sử dụng phương thức function để định nghĩa một đối tượng.

```
function object_name(param_1 [, param_2…])
{
      this.property_1 = para_1;
      this.property_2 = para_2;
      ….
}
```

2. Khởi tạo đối tượng – lệnh new

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

Cú pháp :

```
objectvar = new object_name ( param1 [,param2...])
```

vd Tạo trang (New.htm) 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

```
<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>
```

3. Từ khoá this

Từ khoá this được sử dụng để chỉ đối tượng hiện thời đang gọi (hay chứa) thuộc tính hoặc phương thức (hàm) đó.

Cú pháp

```
this.property
this.method()
```

Có thể xem VD của lệnh new.

VD: Tạo trang (This.htm) minh hoạ cách thức tạo ra và sử dụng hàm như là thành viên của một đối tượng. Hàm printStats được tạo ra là phương thức của đối tượng person

```
<HTML>
<HEAD><TITLE>Function 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;
    this.printStats=printStats;
}
function printStats() {
    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();
person2.printStats();
person3.printStats();
```

```
person4.printStats();
</SCRIPT>
</HEAD>
<BODY> </BODY>
</HTML>
```

4. Lệnh For...in

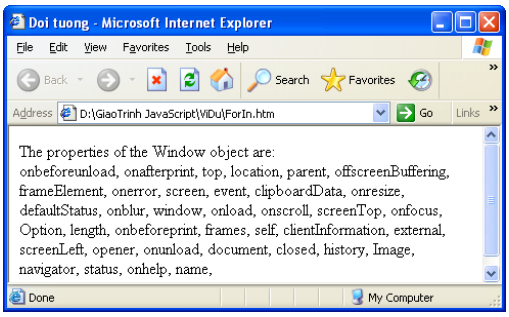
Câu lệnh này được sử dụng để lặp tất cả các thuộc tính (property) của một đối tượng. Tên biến có thể là một giá trị bất kỳ, chỉ cần thiết khi bạn sử dụng các thuộc tính trong vòng lặp. VD sau sẽ minh hoạ điều này

Cú pháp

```
for (<variable> in <object>) {
 //Các câu lệnh
}
```

vd: Tạo trang (ForIn.htm) sau sẽ lấy tất cả các thuộc tính của đối tượng Window và In tên của mỗi thuộc tính.

```
<Body>
<SCRIPT LANGUAGE= "JavaScript"><BODY>
document.write("The properties of the Window object are: <BR>");
for (var x in window)
document.write(" "+ x + ", ");
</SCRIPT>
</Body>
```



5. Lệnh 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.

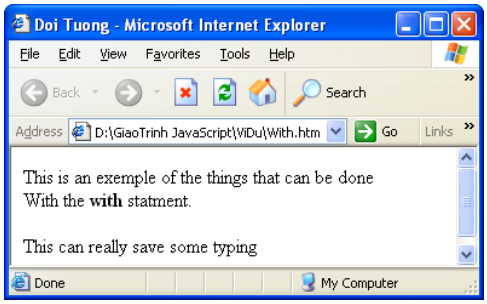
Cú pháp

```
with(object){
 // statement
}
```

vd Tạo trang (With.htm) 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>Doi Tuong</TITLE>
<SCRIPT LANGUAGE= "JavaScript">
with (document){
```

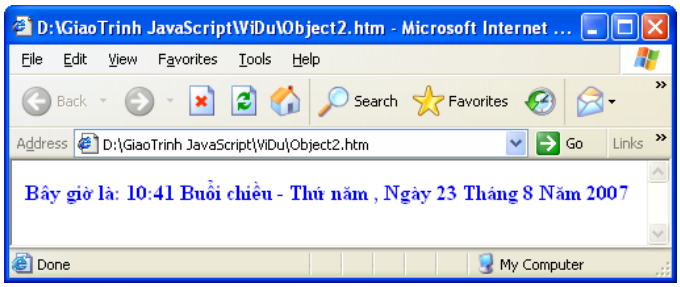
```
    write("This is an exemple of the things that can be done <BR>");
    write("With the <B>with<B> statment. <P>");
    write("This can really save some typing");
    }
    </SCRIPT>
    </HEAD>
    <BODY> </BODY>
    </HTML>
```



vd Tạo trang (Object1.htm)

```
    <HTML>
    <HEAD>
    <TITLE>Doi Tuong</TITLE>
    <SCRIPT LANGUAGE="JavaScript">
    function displayInfo() {
    document.write("<H1>Employee Profile: " + this.name + "</H1><HR><PRE>");
    document.writeln("Employee Number: " + this.number);
    document.writeln("Social Security Number: " + this.socsec);
    document.writeln("Annual Salary: " + this.salary);
    document.write("</PRE>");
    }
    function employee() {
    this.name=prompt("Enter Employee's Name","Name");
    this.number=prompt("Enter Employee Number for " + this.name,"000-000");
    this.socsec=prompt("Enter Social Security Number for " + this.name,"000-00-0000");
    this.salary=prompt("Enter Annual Salary for " + this.name,"$00,000");
    this.displayInfo=displayInfo;
    }
    newEmployee=new employee();
    </SCRIPT>
    </HEAD>
    <BODY>
    <SCRIPT LANGUAGE="JavaScript">
    newEmployee.displayInfo();
    </SCRIPT>
    </BODY> </HTML>
```

vd Tạo trang (Object2.htm)




```
<head><meta content="charset=unicode"></head>
<script LANGUAGE="JavaScript">
var day="";
var month="";
var ampm="";
var ampmhour="";
var myweekday="";
var year="";
mydate = new Date();
myday = mydate.getDay();
mymonth = mydate.getMonth()+1;
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 , ";
</script>
<body>
<script>
document.write("<b>" +"Bây giờ là: "+ ampmhour + "" + myminutes +" "+ ampm)
document.write(" - " + day + " Ngày " + myweekday +" ");
document.write( " Tháng " + mymonth + " Năm " + year + "</font>");
</script>
</body>
```

III. Sự kiện

JavaScript là ngôn ngữ định hướng sự kiện, nghĩa là sẽ phản ứng trước các sự kiện xác định trước như kích chuột hay tải một văn bản. Một sự kiện có thể gây ra việc thực hiện một đoạn mã lệnh (gọi là các chương triình xử lý sự kiện) giúp cho chương trình có thể phản ứng một cách thích hợp.

Chương trình xử lý sự kiện (Event handler): Một đoạn mã hay một hàm được thực hiện để phản ứng trước một sự kiện gọi là chương trình xử lý sự kiện. Chương trình xử lý sự kiện được xác định là một thuộc tính của một thẻ HTML:

```
<tagName  eventHandler = "JavaScript Code or Function">
```

vd sau gọi hàm CheckAge() mỗi khi giá trị của trường văn bản thay đổi:

```
<input type=text name="age" onChange="CheckAge()">
```

Đoạn mã của chương trình xử lý sự kiện không là một hàm; nó là các lệnh của JavaScript cách nhau bằng dấu chấm phẩy. Tuy nhiên cho mục đích viết thành các module nên viết dưới dạng các hàm.

Một số chương trình xử lý sự kiện trong JavaScript:

onBlur	Xảy ra khi input focus bị xoá từ thành phần form
onClick	Xảy ra khi người dùng kích vào các thành phần hay liên kết của form.
onChange	Xảy ra khi giá trị của thành phần được chọn thay đổi
onFocus	Xảy ra khi thành phần của form được focus(làm nổi lên).
onLoad	Xảy ra trang Web được tải.
onMouseOver	Xảy ra khi di chuyển chuột qua kết nối hay anchor.
onSelect	Xảy ra khi người sử dụng lựa chọn một trường nhập dữ liệu trên form.
onSubmit	Xảy ra khi người dùng đưa ra một form.
onUnLoad	Xảy ra khi người dùng đóng một trang

Sau đây là bảng các chương trình xử lý sự kiện có sẵn của một số đối tượng. Các đối tượng này sẽ được trình bày kỹ hơn trong phần sau.

Đối tượng	Chương trình xử lý sự kiện có sẵn
Selection list	onBlur, onChange, onFocus
Text	onBlur, onChange, onFocus, onSelect
Textarea	onBlur, onChange, onFocus, onSelect
Button	onClick
Checkbox	onClick
Radio button	onClick
Hypertext link	onClick, onMouseOver, onMouseOut
Clickable Imagemap area	onMouseOver, onMouseOut
Reset button	onClick
Submit button	onClick

Document	onLoad, onUnload, onError
Window	onLoad, onUnload, onBlur, onFocus
Framesets	onBlur, onFocus
Form	onSubmit, onReset
Image	onLoad, onError, onAbort

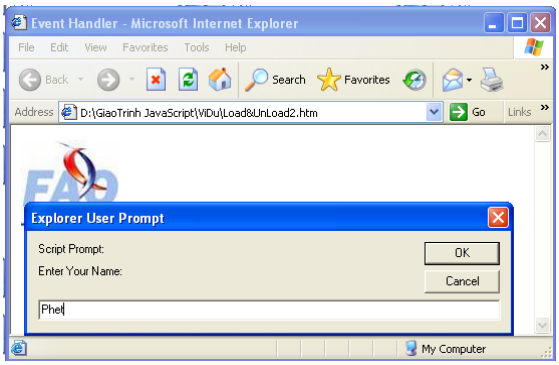
vd (Trang EventHander.htm) sau là một đoạn mã script đơn giản của chương trình xử lý sự kiện thẩm định giá trị đưa vào trong trường text. Tuổi của người sử dụng được nhập vào trong trường này và chương trình xử lý sự kiện sẽ thẩm định tính hợp lệ của dữ liệu đưa vào. Nếu không hợp lệ sẽ xuất hiện một thông báo yêu cầu nhập lại. Chương trình xử lý sự kiện được gọi mỗi khi trường AGE bị thay đổi và focus chuyển sang trường khác.

```
<HTML><HEAD>
<TITLE> An Event Handler Exemple </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
function CheckAge(form) {
if ( (form.AGE.value<0)||(form.AGE.value>120) ) {
alert("Tuổi nhập vào không hợp lệ! Mời bạn nhập  lại");
     form.AGE.value=0;
     }
}
</SCRIPT>
</HEAD>
<BODY>
<FORM NAME="PHIEU_DIEU_TRA">
Nhập vào tên của bạn:<BR>
Tên <INPUT TYPE=TEXT NAME="TEN" MAXLENGTH=10 SIZE=10><BR>
Đệm <INPUT TYPE=TEXT NAME="DEM" MAXLENGTH=15 SIZE=10><BR>
Họ  <INPUT TYPE=TEXT NAME="HO" MAXLENGTH=10 SIZE=10><BR>
Age <INPUT TYPE=TEXT NAME="AGE" MAXLENGTH=3 SIZE=2
onChange="CheckAge(PHIEU_DIEU_TRA)"><BR>
<P>
Bạn thích mùa nào nhất:<BR>
Mùa xuân<INPUT TYPE=RADIO NAME="MUA" VALUE="Mua xuan">
Mùa hạ<INPUT TYPE=RADIO NAME="MUA" VALUE="Mua ha">
Mùa thu<INPUT TYPE=RADIO NAME="MUA" VALUE="Mua thu">
Mùa đông<INPUT TYPE=RADIO NAME="MUA" VALUE="Mua dong">
<P>
Hãy chọn những hoạt động ngoài trời mà bạn yêu thích:<BR>
Đi bộ<INPUT TYPE=CHECKBOX NAME="HOAT_DONG" VALUE="Di bo">
Trượt tuyết<INPUT TYPE=CHECKBOX NAME="HOAT_DONG" VALUE="Truot
tuyet">
Thể thao dưới nước<INPUT TYPE=CHECKBOX NAME="HOAT_DONG"
VALUE="Duoi nuoc">
Đạp xe<INPUT TYPE=CHECKBOX NAME="HOAT_DONG" VALUE="Dap xe">
<P><INPUT TYPE=SUBMIT>
```

```
<INPUT TYPE=RESET>
</FORM></BODY></HTML>
```

vd (Trang Load&UnLoad.htm)

```
<HTML>
<HEAD>
<TITLE>Event Handler</TITLE>
</HEAD>
<BODY onLoad="alert('Welcome to my page!');"
onUnload="alert('Goodbye! ');">
 <IMG SRC="Logo.jpg">
</BODY>
</HTML>
```



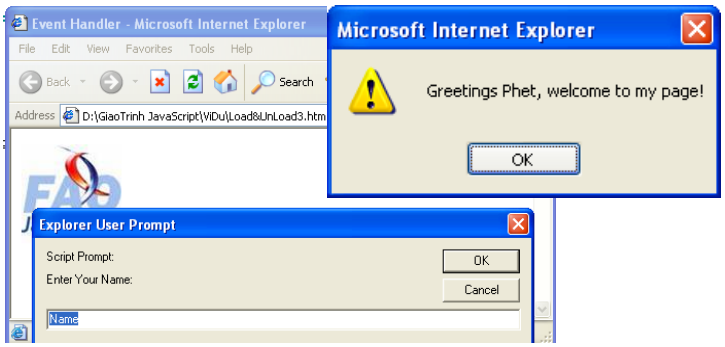
vd (Trang Load&UnLoad2.htm)

```
<HTML>
<HEAD>
<TITLE>Event Handler </TITLE>
<SCRIPT LANGUAGE="JavaScript">
 var name = "";
</SCRIPT>
</HEAD>
<BODY onLoad="name = prompt('Enter Your Name:','Name');
      alert('Greetings ' + name + ', welcome to my page!');"
           onUnload=" alert(Goodbye ' + name + ', sorry to see
      you go!');">
<IMG SRC="logo.jpg ">
</BODY>
</HTML>
```

VD (Trang Load&UnLoad3.htm)

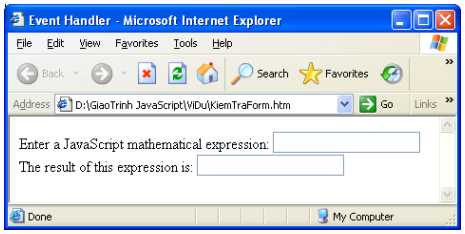
```
<HTML>
<HEAD>
<TITLE>Event Handler </TITLE>
<SCRIPT LANGUAGE="JavaScript">
var name = "";
function hello() {
 name = prompt('Enter Your Name:','Name');
 alert('Greetings ' + name + ', welcome to my page!');
}
function goodbye() {
 alert('Goodbye ' + name + ', sorry to see you go!');
}
</SCRIPT>
</HEAD>
<BODY onLoad="hello();" onUnload="goodbye();">
```

```
 <IMG SRC="logo.jpg">
</BODY> </HTML>
```



vd (KiemtraForm.htm)

```
<html>
<head>
<title>event handler</title>
<script language="javascript">
      function calculate(form) {
       form.results.value = eval(form.entry.value);}
      function getexpression(form) {
       form.entry.blur();
       form.entry.value = prompt("please enter a javascript
 mathematical expression","");
       calculate(form); }
</script>
</head>
<body>
<form method=post>
 enter a javascript mathematical expression:
 <input type="text" name="entry" value=""
onfocus="getexpression(this.form);"><br>
      the result of this expression is: <input type="text"
 name="results" value="" onfocus="this.blur();">
</form> </body> </html>
```



Ghi chu:

formObjectName.fieldname:Dùng để chỉ tên trường của hiện hành trong Form.

formObjectName.fieldname.value: dùng lấy giá trị của trường form hiện hành.

Bài 5 CÁC ĐỐI TƯỢNG TRONG JAVASCRIPT

I. Giới thiệu

Như đã nói JavaScript là ngôn ngữ lập trình dựa trên đối tượng, nhưng không hướng đối tượng bởi vì nó không hỗ trợ các lớp cũng như tính thừa kế. Phần này nói về các đối tượng có sẵn trong JavaScript và sơ đồ phân cấp các đối tượng.

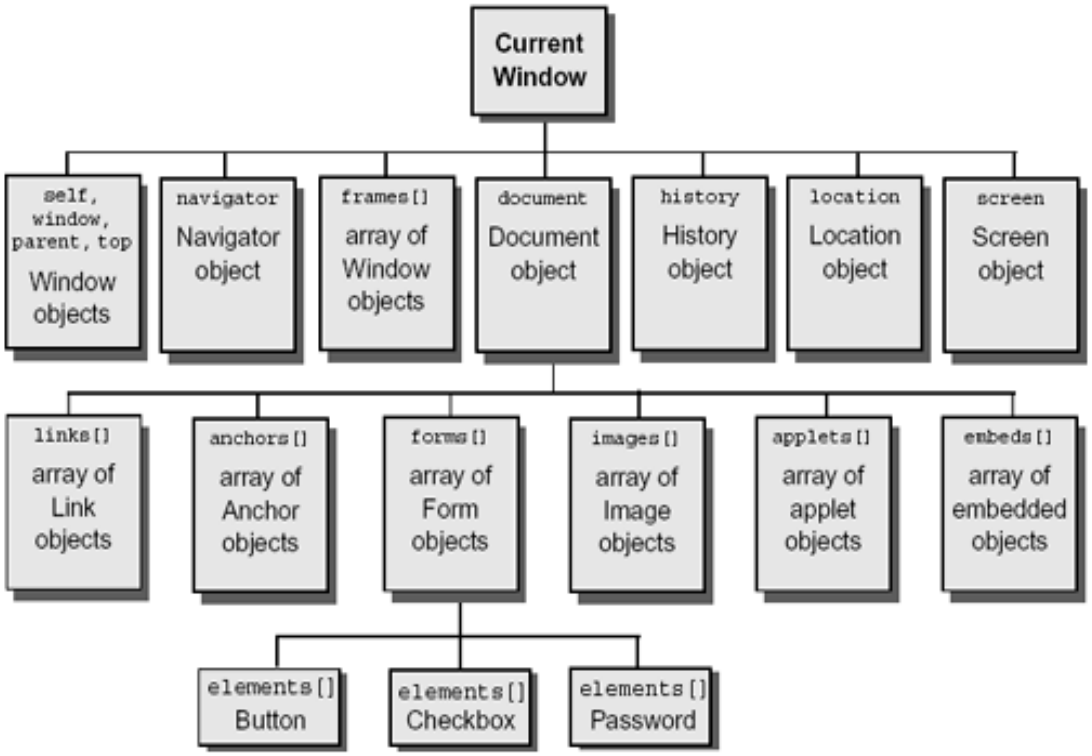
Trong sơ đồ phân cấp các đối tượng của JavaScript, các đối tượng con thực sự là các thuộc tính của các đối tượng bố mẹ. Trong VD về chương trình xử lý sự kiện trước đây form tên PHIEU_DIEU_TRA là thuộc tính của đối tượng document và trường text AGE là thuộc tính của form PHIEU_DIEU_TRA. Để tham chiếu đến giá trị của AGE, bạn phải sử dụng:

document.PHIEU_DIEU_TRA.AGE.value

Các đối tượng có thuộc tính (properties), phương thức (methods), và các chương trình xử lý sự kiện (event handlers) gắn với chúng. VD đối tượng document có thuộc tính title phản ánh nội dung của thẻ <TITLE> của document. Bên cạnh đó bạn thấy phương thức document.write được sử dụng trong nhiều VD để đưa văn bản kết quả ra document.

Đối tượng cũng có thể có các chương trình xử lý sự kiện. VD đối tượng link có hai chương trình xử lý sự kiện là onClick và onMouseOver. onClick được gọi khi có đối tượng link được kích, onMouseOver được gọi khi con trỏ chuột di chuyển qua link.

Khi bạn tải một document xuống Navigator, nó sẽ tạo ra một số đối tượng cùng với những giá trị các thuộc tính của chúng dựa trên File HTML của document đó và một vài thông tin cần thiết khác. Những đối tượng này tồn tại một cách có cấp bậc và phản ánh chính cấu trúc của File HTML đó. Sơ đồ sau sẽ minh hoạ sự phân cấp của các đối tượng này



Trong sơ đồ phân cấp này, các đối tượng con chính là các thuộc tính của một đối tượng cha. Vd như một form tên là form1 chính là một đối tượng con của đối tượng document và được gọi tới là document.form1

Tất cả các trang đều có các đối tượng sau đây:

- navigator: có các thuộc tính tên và phiên bản của trình duyệt (Browser) đang được sử dụng, dùng cho MIME type được hỗ trợ bởi client và plug-in được cài đặt trên client.
- window: là đối tượng ở mức cao nhất, có các thuộc tính thực hiện áp dụng vào toàn bộ cửa sổ.
- document: chứa các thuộc tính dựa trên nội dung của document như tên, màu nền, các kết nối và các forms.
- location: có các thuộc tính dựa trên địa chỉ URL hiện thời
- history: Chứa các thuộc tính về các URL mà client yêu cầu trước đó.

II. CÁC ĐỐI TƯỢNG

Sau đây sẽ mô tả các thuộc tính, phương thức cũng như các chương trình xử lý sự kiện cho từng đối tượng trong JavaScript.

1. Đối tượng navigator

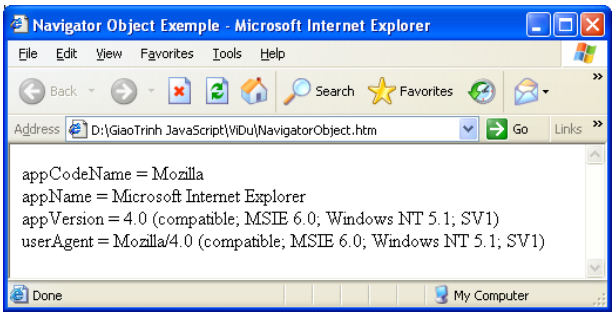
Đối tượng này được sử dụng để đạt được các thông tin về trình duyệt như số phiên bản. Đối tượng này không có phương thức hay chương trình xử lý sự kiện.

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.

Vd (Trang NavigatorObject.htm) sau sẽ hiển thị các thuộc tính của đối tượng navigator

```
<html>
<head>
<title> navigator object exemple </title>
<script language= "javascript">
 document.write("appcodename = "+navigator.appCodename +
"<br>");
 document.write("appname = "+navigator.appName + "<br>");
 document.write("appversion = "+navigator.appVersion + "<br>");
 document.write("useragent
= "+navigator.userAgent +
"<br>");
</script>
</head>
<body></body>
</html>
```



2. Đối tượng window

Đối tượng window (parent, seft, top) như đã nói ở trên là đối tượng ở mức cao nhất. Các đối tượng document, frame, vị trí đều là thuộc tính của đối tượng window.

Các thuộc tính:

closed	Xác định cửa sổ đã đóng hay chưa (true / false)
defaultStatus	Thông báo ngầm định hiển thị lên trên thanh trạng thái của cửa sổ
frames	Mảng xác định tất cả các frame trong cửa sổ.
document	Thể hiện đối tượng document
length	Số lượng các frame trong cửa sổ cha.
name	Tên của cửa sổ hiện thời.
parent	Đối tượng cửa sổ cha
self	Cửa sổ hiện thời.
status	Được sử dụng cho thông báo tạm thời hiển thị lên trên thanh thạng thái cửa sổ. Được sử dụng để lấy hay đặt lại thông báo trạng thái và ghi đè lên defaultStatus.
top	Cửa sổ ở trên cùng (chứa cửa sổ chỉ định)
window	Cửa sổ hiện thời.

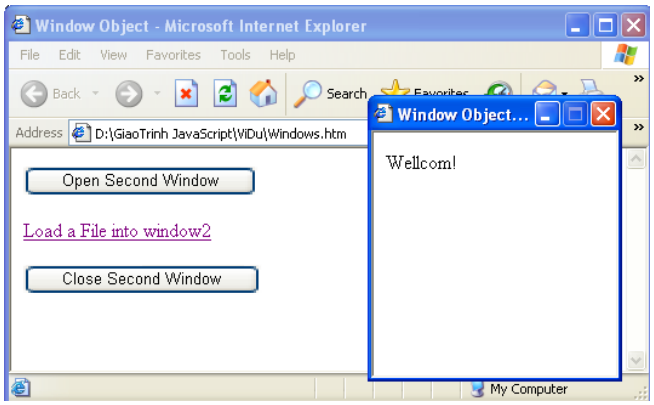
Các phương thức

alert ("message")	Hiển thị hộp hội thoại với chuỗi "message" và nút OK.
blur()	Xoá trỏ chuột khỏi cửa sổ
focus()	Đặt trỏ chuột vào cửa sổ
clearTimeout(timeoutID)	Xóa timeout do setTimeout đặt. SetTimeout trả lại timeoutID
clearInterval(intervalID)	Xóa interval do setInterval đặt. SetInterval trả lại IntervalID
windowReference.close()	Đóng cửa sổ windowReference.
confirm("message")	Hiển thị hộp hội thoại với chuỗi "message", nút OK và nút Cancel. Trả lại giá trị True cho OK và False cho Cancel.
open(url, name, [options])	Mở cửa sổ mới.
prompt ("message","defaultInput")	Mở một hộp hội thoại để nhận dữ liệu vào trường text.

setTimeout(expression/function, msec)	Đánh giá biểu thức expresion sau thời gian msec, trả về ID của timeout được set.
setInterval(expression/function, msec)	Đánh giá biểu thức expresion sau thời gian msec, trả về ID của interval được set.

VD (TrangWindows.htm) Sử dụng tên cửa sổ khi gọi tới nó như là đích của một form submit hoặc trong một Hypertext link (thuộc tính TARGET của thẻ FORM và A). Trong VD tạo ra một tới cửa sổ thứ hai, như nút thứ nhất để mở một cửa sổ rỗng, sau đó một liên kết sẽ tải File doc2.html xuống cửa sổ mới đó rồi một nút khác dùng để đóng của sổ thứ hai lại,

```
<html>
<head>
<title>window object </title>
</head>
<body>
<form>
<input type="button" value="open second window"
onclick="msgwindow=window.open('','window2','resizable=no,width=200,
height=200')"><p>
<a href="doc.htm"
target="window2"> load a
file into window2 </a></p>
<input type="button"
value="close second window"
onclick="msgwindow.close()">
</form>
</body>
</html>
```



Các chương trình xử lý sự kiện

onLoad	Xuất hiện khi cửa sổ kết thúc việc tải.
onUnLoad	Xuất hiện khi cửa sổ được loại bỏ.

3. Đối tượng location

Các thuộc tính của đối tượng location duy trì các thông tin về URL của document hiện thời. Đối tượng này hoàn toàn không có các phương thức và chương trình xử lý sự kiện đi kèm.

VD: http:// www.abc.com/ chap1/page2.html#topic3

Các thuộc tính

hash	Tên anchor của vị trí hiện thời (VD topic3).
host	Phần hostname:port của URL (VD www.abc.com). Chú ý rằng đây thường là cổng ngầm định và ít khi được chỉ ra.
hostname	Tên của host và domain (VD www.abc.com).

href	Toàn bộ URL cho document hiện tại.
pathname	Phần đường dẫn của URL (VD /chap1/page2.html).
port	Cổng truyền thông được sử dụng cho máy tính host, thường là cổng ngầm định.
protocol	Giao thức được sử dụng (cùng với dấu hai chấm) (VD http:).
search	Câu truy vấn tìm kiếm có thể ở cuối URL cho các script CGI.

Các phương thức

reload()	Tải lại đường dẫn hiện tại.
replace()	Thay thế trang hiện tại bằng một trang mới

4. Đối tượng Frame

Một cửa sổ có thể có một vài frame, frame là một cửa sổ con trong một của sổ lớn hơn. Các frame có thể cuộn một cách độc lập với nhau và mỗi frame có URL riêng. frame không có các chương trình xử lý sự kiện. Sự kiện onLoad và onUnLoad là của đối tượng window.

Các thuộc tính

frames	Mảng tất cả các frame trong cửa sổ.
name	Thuộc tính NAME của thẻ <FRAME>
length	Số lượng các frame con trong một frame.
parent	Cửa sổ hay frame chứa nhóm frame hiện thời.
self	frame hiện thời.
window	frame hiện thời.
top	cửa sổ trên cùng.

Các phương thức

clearTimeout (timeoutID)	Xoá timeout do setTimeout lập. SetTimeout trả lại timeoutID.
setTimeout (expression,msec)	Đánh giá expression sau khi hết thời gian msec.
blur()	Xoá trỏ chuột khỏi frame
focus()	Đặt trỏ chuột vào frame

Cách sử dụng

- **Tạo một frame(create)**

Để tạo một frame, ta sử dụng thẻ FRAMESET. Mục đích của thẻ này là định nghĩa một tập các frame trong một trang.

Lưu ý: Trang tạo Frame không sử dụng Thẻ <Body> . . .</Body>

VD: Tạo frame (Trang Frame1.htm)

```
<html>
<head>
<title>frame example </title>
</head>
<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 >
</html>
```

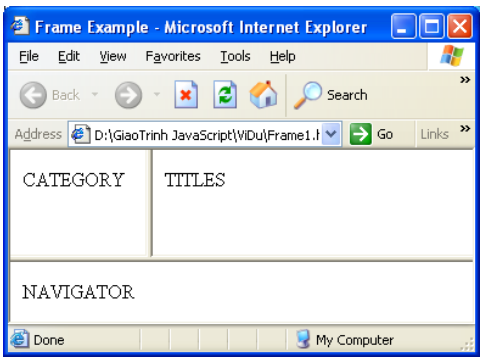
Sơ đồ sau hiển thị cấu trúc của các frame: Cả 3 frame đều trên cùng một cửa sổ cha, mặc dù 2 trong số các frame đó nằm trong một frameset khác.

Top

- listFrame (category.html)
- contentFrame (titles.html)
- navigatorFrame (navigator.html)

Bạn có thể gọi tới những frame trước đó bằng cách 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]



Vd Tạo Frame (Trang Frame2.htm) Cũng giống như một sự lựa chọn, bạn có thể tạo ra một cửa sổ giống như VD trước nhưng trong mỗi đỉnh của hai frame lại có một cửa sổ cha riêng từ navigateFrame. Mức frameset cao nhất có thể được định nghĩa như sau:

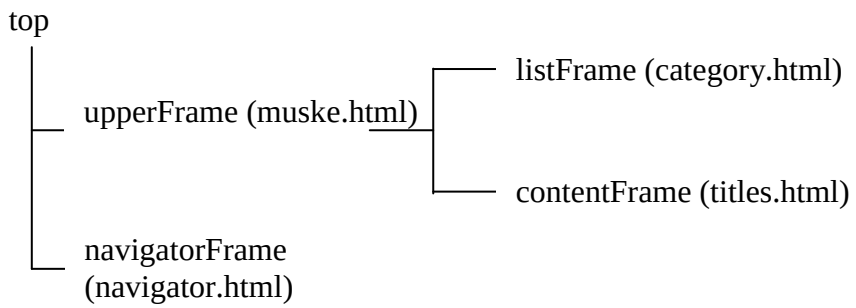
```
<html>
<head>
<title>frame example </title>
</head>
<frameset rows="90%,10%">
```

```
<frame src=muske.htm name="upperframe">
<frame src=navigator.htm name="navigateframe">
</frameset >
</html>
```

trong file muske.html lại tiếp tục đặt một frameset:

```
<html>
<head>
<title>frame example </title>
</head>
<frameset cols="30%,70%">
<frame src=category.htm name="listframe">
<frame src=titles.htm name="contentframe">
</frameset >
</html>
```

Khi đó kết quả hiển thị của VD 2 giống VD 1 nhưng sự phân cấp của các frames lại khác hẳn:



Bạn có thể gọi tới các frame trên bằng cách sử dụng thuộc tính mảng frames như sau:

upperFrame	chính là top.frames[0]
navigatorFrame	chính là top.frames[1]
listFrame	chính là upperFrame.frames[0] hoặc top.frames[0].frames[0]
contentFrame	chính là upperFrame.frames[1] hoặc top.frames[0].frames[1]

- **Cập nhật một frame (update)**

Bạn có thể cập nhật nội dung của một frame bằng cách sử dụng thuộc tính location để đặt địa chỉ URL và phải định chỉ rõ vị trí của frame trong cấu trúc.

Trong VD trên, nếu bạn thêm một dòng sau vào trong trang Navigator.htm:

```
<input type="button" value="Titles only"
onClick="top.frames[0].location='artist.htm'">
```

thì khi nút “Titles only” được nhấn, File artist.html sẽ được tải vào upperFrame, và hai frame listFrame, contentFrame sẽ bị đóng lại như chúng chưa bao giờ tồn tại.

5. Đối tượng document

Đối tượng này chứa các thông tin về document hiện thời và cung cấp các phương thức để đưa thông tin ra màn hình. Đối tượng document được tạo ra bằng cặp thẻ <BODY> và </BODY>. Một số các thuộc tính gắn với thẻ <BODY>.

Các thuộc tính

alink	Màu của các liên kết khi active
anchor	Mảng tất cả các anchor trong document.
bgColor	Màu nền trang web
cookie	Sử dụng để xác định cookie.
fgColor	Màu của văn bản
forms	Mảng tất cả các form trong document.
lastModified	Ngày cuối cùng văn bản được sửa.
link	Màu của các liên kết chưa sử dụng
links	Mảng tất cả các link trong document.
location	URL đầy đủ của văn bản.
referrer	URL của văn bản gọi nó.
title	Nội dung của thẻ <TITLE>.
vlink	Màu của liên kết đã sử dụng
url	URL đầy đủ của văn bản.
images	Mảng các hình ảnh trong trang.

Các phương thức

clear ()	Xoá document hiện thời.
close ()	Đóng dòng dữ liệu vào và đưa toàn bộ dữ liệu trong bộ đệm ra màn hình.
open (["mineType"])	Mở một stream để thu thập dữ liệu vào của các phương thức write và writeln.
write(expression1 [,expression2]...[,expressionN])	Viết biểu thức HTML lên văn bản trông một cửa sổ xác định.
writeln (expression1 [,expression2] ... [,expressionN])	Giống phương thức trên nhưng khi hết mỗi biểu thức lại xuống dòng.

6. Đối tượng anchors

anchor là một đoạn văn bản trong document có thể dùng làm đích cho một siêu liên kết. Nó được xác định bằng cặp thẻ <A> và </A>. Đối tượng anchor không có thuộc tính, phương thức cũng như chương trình xử lý sự kiện. Mảng anchor tham chiếu đến mỗi anchor có tên trong document. Mỗi anchor được tham chiếu bằng cách:

```
document.anchors[index]
```

Mảng anchor có một thuộc tính duy nhất là length xác định số lượng các anchor trong document, nó có thể được xác định như sau:

```
document.anchors.length.
```

7. Đối tượng forms

Các form được tạo ra nhờ cặp thẻ <FORM> và </FORM>. Phần lớn các thuộc tính của đối tượng form phản ánh các thuộc tính của thẻ <FORM>. Có một vài phần tử (elements) là thuộc tính của đối tượng forms:

Button, checkbox, hidden, password, radio, reset, select, submit, text, textarea

Các phần tử này sẽ được trình bày sau.

Nếu document chứa một vài form, chúng có thể được tham chiếu qua mảng forms. Số lượng các form có thể được xác định như sau:

```
document.forms.length
```

Mỗi một form có thể được tham chiếu như sau:

```
document.forms[index]
```

Các thuộc tính

action	thuộc tính action của thẻ form.
elements	Mảng chứa tất cả các thành phần trong một form (như checkbox, trường text, danh sách lựa chọn
encoding	Xâu chứa kiểu MIME được sử dụng để mã hoá nội dung của form gửi cho server.
length	Số lượng các thành phần trong một form.
method	Thuộc tính method.
target	Xâu chứa tên của cửa sổ đích khi submit form

Các phương thức

Phương thức `submit()` - Xuất dữ liệu của một form tên formName tới trang xử lý. Phương thức này mô phỏng một click vào nút submit trên form.

Phương thức `reset()` – Thiết lập các trường trên form về giá trị mặc định.

Các chương trình xử lý sự kiện

onSubmit - Chương trình xử lý sự kiện này được gọi khi người sử dụng chuyển dữ liệu từ form đi.

Các phần tử của đối tượng Form

Form được tạo bởi các phần tử cho phép người sử dụng đưa thông tin vào. Khi đó, nội dung (hoặc giá trị) của các phần tử sẽ được chuyển đến một chương trình trên server qua một giao diện được gọi là Common Gateway Interface(Giao tiếp qua một cổng chung) gọi tắt là CGI

Sử dụng JavaScript bạn có thể viết những đoạn scripts chèn vào HTML của bạn để làm việc với các phần tử của form và các giá trị của chúng.

Bảng Các phần tử của form

Phần tử	Mô tả
button	Là một nút bấm hơn là nút submit hay nút reset (<input type="button">)
checkbox	Một checkbox (<input type="checkbox">)
fileupload	Là một phần tử tải File lên cho phép người sử dụng gửi lên một File (<input type="File">)
hidden	Một trường ẩn (<input type="hidden">)
password	Một trường text để nhập mật khẩu mà tất cả các ký tự nhập vào đều hiển thị là dấu (*)(<input type="password">)
radio	Một nút bấm (<input type="radio">)
reset	Một nút reset(<input type="reset">)
select	Một danh sách lựa chọn (<select><option>option1</option> <option>option2</option></select>)
submit	Một nút submit (<input type="submit">)
text	Một trường text (<input type="text">)
textarea	Một trường text cho phép nhập vàp nhiều dòng <textarea>default text</textarea>)

Thuộc tính **name** : Mỗi phần tử có thể được đặt tên để JavaScript truy nhập đến chúng qua tên.

Thuộc tính **form** : chỉ định đối tượng form chứa phần tử.

Thuộc tính **value** : chỉ định đối tượng giá trị của phần tử.

Thuộc tính **type** : Trong mỗi phần tử của form đều có thuộc tính type, đó là một xâu chỉ định rõ kiểu của phần tử được đưa vào như nút bấm, một trường text hay một checkbox... Xâu đó có thể là một trong các giá trị sau:

- Text field: "text"
- Radio button: "radio"
- Checkbox: "checkbox"
- Hidden field: "hidden"
- Submit button: "submit"
- Reset button: "reset"
- Password field: "password"
- Button: "button"
- Select list: "select-one"
- Multiple select lists: "select-multiple"
- Textarea field: "textarea"

a. Phần tử button

Trong một form HTML chuẩn, chỉ có hai nút bấm có sẵn là submit và reset bởi vị dữ liệu trong form phải được gửi tới một vài địa chỉ URL (thường là CGI-BIN script) để xử lý và lưu trữ. Một phần tử button được chỉ định rõ khi sử dụng thẻ INPUT:

<input type="button" name="name" value= "buttonname">

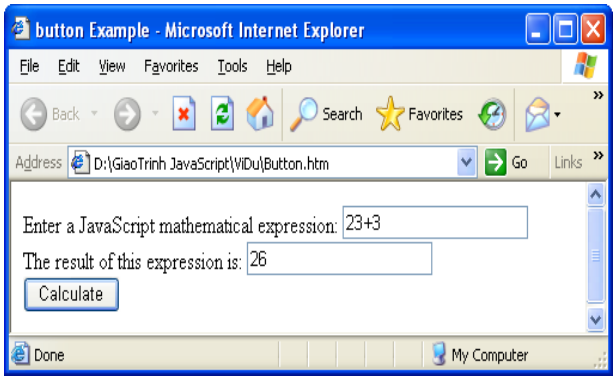
Trong thẻ INPUT, name là tên của button, thuộc tính VALUE có chứa nhãn của button sẽ được hiển thị trên Navigator của browser.

Chỉ có một thẻ sự kiện duy nhất đối với button là onClick. Kết hợp với nó là cách thức duy nhất click. Phần tử buttton có khả năng mở rộng cho phép người lập trình JavaScript có thể viết được một đoạn mã lệnh JavaScript để thực thi việc thêm vào một nút bấm trong một script.

Trong VD sau, thay vì sử dụng onChange, bạn có thể chỉnh sửa script để định giá biểu thức khi button được bấm.

VD (Trang Button.htm) Định giá một form sử dụng phần tử button.

```
<head>
<title>button example</title>
<script language="javascript">
function calculate(form) {
 form.results.value =
eval(form.entry.value);
}
</script>
</head>
<body>
  <form method=post>
       enter a javascript
     mathematical expression:
      <input type="text" name="entry" value=""> <br>
      the result of this expression is:
      <input type=text name="results" onfocus="this.blur();">
     <br>
      <input type="button" value="calculate"
     onclick="calculate(this.form);">
```



```
  </form>
</body>
```

b. Phần tử checkbox

Các phần tử checkbox có khả năng bật tắt dùng để chọn hoặc không chọn một thông tin. Các checkbox có nhiều thuộc tính và cách thức hơn button. Bảng dưới đây là danh sách các thuộc tính và các cách thức của phần tử checkbox.

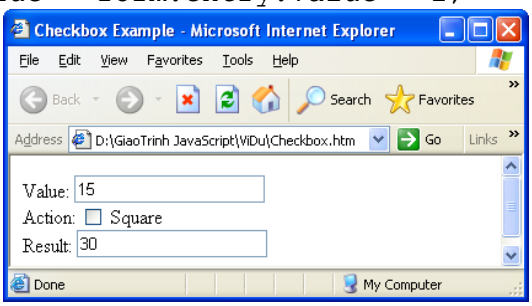
Cách thức và thuộc tính	Mô tả
checked	Cho biết trạng thái hiện thời của checkbox (thuộc tính)
defaultChecked	Cho biết trạng thái mặc định của phần tử (thuộc tính)
name	Cho biết tên của phần tử được chỉ định trong thẻ INPUT (thuộc tính)
value	Cho biết giá trị hiện thời của phần tử được chỉ định trong thẻ INPUT (thuộc tính)
click()	Mô tả một click vào checkbox (Cách thức)

Phần tử checkbox chỉ có một thẻ sự kiện là onClick

VD (Trang Checkbox.htm) Tạo hộp checkbox để nhập vào một số rồi lựa chọn tính nhân đôi và bình phương:

```
<head>
<title>checkbox example</title>
<script>
function calculate(form,callingfield) {
if (callingfield == "result") {
    if (form.square.checked){
          form.entry.value = math.sqrt(form.result.value);
    }else{
          form.entry.value = form.result.value / 2;
          }     //end if(2)
    }else{
          if (form.square.checked){

form.result.value=form.entry.value*form.entry.value;
          }else {
                form.result.value = form.entry.value * 2;
          }
    }
}
</script>
</head>
<body>
<form method=post>
value: <input type="text" name="entry" value="0"
```



```
onchange="calculate(this.form,this.name);"><br>
action: <input type="checkbox" name="square"
 onclick="calculate(this.form,this.name);"> square<br>
result: <input type="text" name="result" value="0"
 onchange="calculate(this.form,this.name);">
</form></body>
```

Trong script này, bạn đã thấy cách sử dụng thẻ sự kiện onClick cũng như thuộc tính checked là một giá trị kiểu Boolean có thể dùng làm điều kiện trong câu lệnh if...else

Bạn có thể thêm một checkbox tên là square vào form. Nếu hộp này được check, chương trình sẽ lấy giá trị của nó, nếu không, một thực thi được mặc định sẽ nhân đôi giá trị của nó. Thẻ sự kiện onClick trong checkbox được định nghĩa:

(<INPUT TYPE=checkbox NAME=square onClick= "calculate(this.form, this.name);">)

Khi đó nếu người dùng thay đổi một câu lệnh khác, form sẽ được tính toán lại. Để tạo ra sự mở rộng cho checkbox. bạn có thể thay đổi hàm calculate() như sau:

```
function calculate(form,callingField) {
if (callingField == "result") {
if (form.square.checked) {
   form.entry.value = Math.sqrt(form.result.value);
   }else { form.entry.value = form.result.value / 2; }
}else {
   if (form.square.checked) {
   form.result.value=form.entry.value*form.entry.value;
  }else {  form.result.value = form.entry.value * 2;  }
 } }
```

c. *Phần tử File Upload*

Phần tử này cung cấp cho form một cách để người sử dụng có thể chỉ rõ một File đưa vào form xử lý. Phần tử File Upload được chỉ định rõ trong JavaScript bằng đối tượng FileUpload.

Đối tượng chỉ có hai thuộc tính là name và value, cả hai đều là giá trị xâu như các đối tượng khác. Không có cách thức hay thẻ File cho đối tượng này.

d. Phần tử hidden

Phần tử hidden là phần tử duy nhất trong số tất cả các phần tử của form không được hiển thị trên Web browser. Trường hidden có thể sử dụng để lưu các giá trị cần thiết để gửi tới server song song với sự xuất ra từ form (form submission) nhưng nó không được hiển thị trên trang. Mọi người có thể sử dụng trong JavaScript để lưu các giá trị trong suốt một script và để tính toán không cần form.

Đối tượng hidden chỉ có hai thuộc tính là name và value, đó cũng là những giá trị xâu giống các đối tượng khác. Không có cách thức hay thẻ sự kiện nào cho đối tượng này.

e. Phần tử Password

Đối tượng Password là đối tượng duy nhất trong các đối tượng của form mà khi gõ bất kỳ ký tự nào vào cũng đều hiển thị dấu sao(*). Nó cho phép đưa vào những thông tin bí mật như đăng ký mật khẩu...

Đối tượng Password có 3 thuộc tính giống trường text là: defaultValue, name và value. Không giống với hai phần tử ở trên, trường Password có nhiều cách thức hơn(focus(), blur(), and select()) và tương ứng với các thẻ sự kiện: onFocus, onBlur, and onSelect. Phần này sẽ được nói kỹ hơn trong đối tượng text.

f. Phần tử radio

Đối tượng radio gần giống sự bật tắt checkbox khi có hai nút radio kết hợp thành một nhóm. Khi nhiều radio được kết hợp thành một nhóm, chỉ có một nút được chọn trong bất kỳ một thời điểm nào. VD dòng lệnh sau tạo ra một nhóm radio có ba nút tên là test:

```
<input type="radio" name="test" value="1"
checked="checked">1<br/>
<input type="radio" name="test" value="2">2<br>
<input type="radio" name="test" value="3">3<br>
```

Nhóm các nút radio lại bằng cách đặt cho chúng có cùng một tên trong các thẻ INPUT. Có một vài thuộc tính để kiểm tra trạng thái hiện thời của một nhóm nút radio. Bảng sau hiển thị các thuộc tính và cách thức của đối tượng radio.

Thuộc tính và cách thức	**Mô tả**
checked	Mô tả trạng thái hiện thời của phần tử radio (thuộc tính)
defaultChecked	Mô tả trạng thái mặc định của phần tử (thuộc tính)
index	Mô tả thứ tự của nút radio được chọn hiện thời trong một nhóm
length	Mô tả tổng số nút radio trong một nhóm
name	Mô tả tên của phần tử được chỉ định trong thẻ INPUT (thuộc tính)
value	Mô tả giá trị hiện thời của phần tử được định ra trong thẻ INPUT (thuộc tính)
click()	Mô phỏng một click trên nút radio (cách thức)

Cũng như checkbox, radio chỉ có một thẻ sự kiện là onClick.

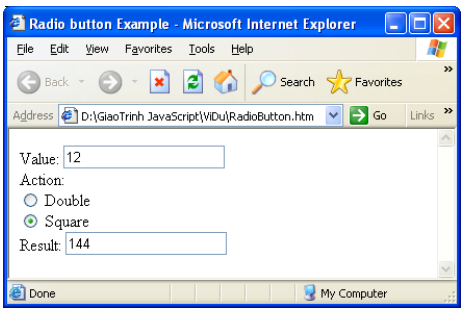
Không có bất kỳ một đối tượng form nào có thuộc tính index và length. Do một nhóm radio gồm nhiều phần tử radio, nên chúng được đặt trong một mảng các nút radio và được đánh số từ 0. Trong VD nhóm radio có tên test ở trên, nếu nhóm đó nằm trong một form có tên là "testform", bạn có thể gọi tới nút radio thứ hai bằng tên "testform.test[1]" và có thể kiểm tra giá trị của nó bằng "testform.test[1].checked"

VD (Trang RadioButton.htm):

```
<html>
<head>
<title>radio button example</title>
```

```
<script>
function calculate(form,callingfield) {
        if (callingfield == "result") {
          if (form.action[1].checked) {
            form.entry.value = math.sqrt(form.result.value);
          } else {
            form.entry.value = form.result.value / 2;
          }
        } else {
          if (form.action[1].checked) {

form.result.value=form.entry.value*form.entry.value;
          } else {
            form.result.value = form.entry.value * 2;
          }
        }
      }
</script>
</head>
<body>
<form method=post>
value: <input type="text"
name="entry" value="0"
onchange="calculate(this.form,this.name);"> <br>
action:<br>
<input type="radio" name="action" value="twice"
onclick="calculate(this.form,this.name);"> double<br>
<input type="radio" name="action" value="square"
onclick="calculate(this.form,this.name);"> square <br>
result: <input type=text name="result" value="0"
onchange="calculate(this.form,this.name);">
</form>
</body>
</html>
```



Trong VD này, sự thay đổi từ checkbox ở trên là rất khó nhận biết. ở đây ta sử dụng hai nút radio với hai giá trị khác nhau: double và square

Như ta đã biết có thể truy nhập đến các nút radio qua một mảng, do đó hai nút này có thể truy nhập bằng action[0] và action[1]. Bằng cách này, bạn chỉ cần thay đổi tham chiếu đến hàm calculate() từ form.square.checked thành form.action[1].checked.

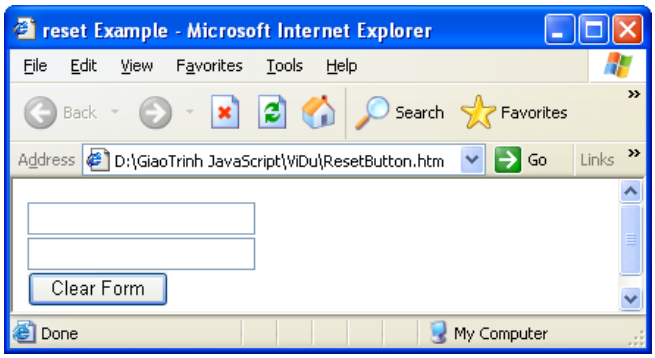
g. Phần tử reset

Sử dụng đối tượng reset, cũng giống đối tượng button, đối tượng reset có hai thuộc tính là name và value, và một cách thức click(), một thẻ sự kiện onClick.

Hầu hết những người lập trình không sử dụng thẻ sự kiện onClick của nút reset để kiểm tra giá trị của nút này, đối tượng reset thường dùng để xoá form.

vd (Trang ResetButton.htm) minh hoạ cách sử dụng nút reset để xoá các giá trị của form.

```
<head>
<title>reset example</title>
<script language="javascript">
function clearform(form) {
    form.value1.value = "form";
    form.value2.value = "cleared";
}
</script>
</head>
<body>
<form method="post">
<input type="text" name="value1"><br>
<input type="text" name="value2"><br>
<input type="reset" value="clear form" onclick="clearform(this.form);">
</form>
</body>
```



h. Phần tử select

Danh sách lựa chọn trong các form HTML xuất hiện menu drop-down hoặc danh sách cuộn được của các đối tượng có thể được lựa chọn. Các danh dách được xây dựng bằng cách sử dụng hai thẻ SELECT và OPTION.

```
<select name="test">
    <option selected="selected">1</option>
    <option>2</option>
    <option>3</option>
</select>
```

Tạo ra ba thành phần của menu thả drop-down với ba lựa chọn 1,2 và 3. Sử dụng thuộc tính SIZE bạn có thể tạo ta một danh sách cuộn với số phần tử hiển thị ở lần thứ nhất. Để bật menu drop-down trong một menu cuộn với hai thành phần hiển thị, bạn có thể sử dụng như sau:

```
<select name="test" size="2">
    <option selected="selected">1</option>
    <option>2</option>
    <option>3</option>
</select>
```

Trong cả hai VD trên, người sử dụng chỉ có 1 lựa chọn. Nếu sử dụng thuộc tính MULTIPLE, bạn có thể cho phép người sử dụng lựa chọn nhiều hơn 1 giá trị trong danh sách lựa chọn:

```
<select name="test" size="2" multiple="multiple">
    <option selected="selected">1</option>
```

```
            <option>2</option>
            <option>3</option>
        </select>
```

Danh sách lựa chọn trong JavaScript là đối tượng select. Đối tượng này tạo ra một vài thành phần tương tự các button và radio.

Với các thành phần lựa chọn, danh sách các lựa chọn được chứa trong một mảng được đánh số từ 0. Trong trường hợp này, mảng là một thuộc tính của đối tượng select gọi là options.

Cả việc lựa chọn các option và từng phần tử option riêng biệt đều có những thuộc tính. Bổ sung thêm vào mảng option, phần tử select có thuộc tính selectedIndex, có chứa số thứ tự của option được lựa chọn hiện thời.

Mỗi option trong danh sách lựa chọn đều có một vài thuộc tính:

defaultSelected	Cho biết option có mặc định là chọn trong thẻ OPTION hay không.
index	Chứa giá trị số thứ tự của option hiện thời trong mảng option.
selected	Cho biết trạng thái hiện thời của option
text	Có chứa giá trị của dòng text hiển thị trên menu cho mỗi option, và thuộc tính value mọi giá trị chỉ ra trong thẻ OPTION.

Đối tượng select không có các cách thức được định nghĩa sẵn. Tuy nhiên, đối tượng select có ba thẻ sự kiện, đó là onBlue, onFocus, onChange, chúng đều là những đối tượng text.

VD bạn có danh sách lựa chọn sau:

```
<select name="example" onfocus="react();">
    <option selected value="number one">1</option>
    <option value="the second">2</option>
    <option value="three is it">3</option>
</select>
```

Khi lần đầu tiên hiển thị bạn có thể truy nhập tới các thông tin sau:

```
example.options[1].value = "The Second"
example.options[2].text = "3"
example.selectedIndex = 0
example.options[0].defaultSelected = true
example.options[1].selected = false
```

Nếu người sử dụng kích vào menu và lựa chọn option thứ hai, thì thẻ onFocus sẽ thực hiện, và khi đó giá trị của thuộc tính sẽ là:

```
example.options[1].value = "The Second"
example.options[2].text = "3"
example.selectedIndex = 1
example.options[0].defaultSelected = true
example.options[1].selected = true
```

Sửa các danh sách lựa chọn

Navigator 3.0 cho phép thay đổi nội dung của danh sách lựa chọn từ JavaScript bằng cách liên kết các giá trị mới cho thuộc tính text của các thực thể trong danh sách.

Trong VD trước, bạn đã tạo ra một danh sách lựa chọn như sau:

```
<select name="example" onfocus="react();">
     <option selected value="number one">1</option>
     <option value="the second">2</option>
     <option value="three is it">3</option>
</select>
```

Có thể thay đổi được dòng text hiển thị trên nút thứ hai thành "two" bằng:

```
example.options[1].text = "two";
```

Có thể thêm các lựa chọn mới vào danh sách bằng cách sử dụng đối tượng xây dựng Option() theo cú pháp:

```
     newOptionName = new Option(optionText, optionValue,
defaultSelected, selected);
     selectListName.options[index] = newOptionName;
```

Việc tạo đối tượng option() này với dòng text được chỉ trước, defaultSelected và selected như trên đã định ra những giá trị kiểu Boolean. Đối tượng này được liên kết vào danh sách lựa chọn được thực hiện bằng index.

Các lựa chọn có thể bị xoá trong danh sách lựa chọn bằng cách gắn giá trị null cho đối tượng muốn xoá

```
     selectListName.options[index] = null;
```

i. Phần tử submit

Nút Submit là một trường hợp đặc biệt của button, cũng như nút Reset. Nút này đưa thông tin hiện tại từ các trường của form tới địa chỉ URL được chỉ ra trong thuộc tính ACTION của thẻ form sử dụng cách thức METHOD chỉ ra trong thẻ FORM.

Giống như đối tượng button và reset, đối tượng submit có sẵn thuộc tính name và value, cách thức click() và thẻ sự kiện onClick.

j. Phần tử Text

Phần tử này nằm trong những phần tử hay được sử dụng nhất trong các form HTML. Tương tự như trường Password, trường text cho phép nhập vào một dòng đơn, nhưng các ký tự của nó hiện ra bình thường.

Đối tượng text có ba thuộc tính:defautValue, name và value. Ba cách thức mô phỏng sự kiện của người sử dụng: focus(), blur() và select(). Có 4 thẻ sự kiện là: obBlur, onFocus, onChange, onSelect. Chú ý các sự kiện này chỉ thực hiện khi con trỏ đã được kích ra ngoài trường text.

Bảng sau mô tả các thuộc tính và cách thức của đối tượng text.

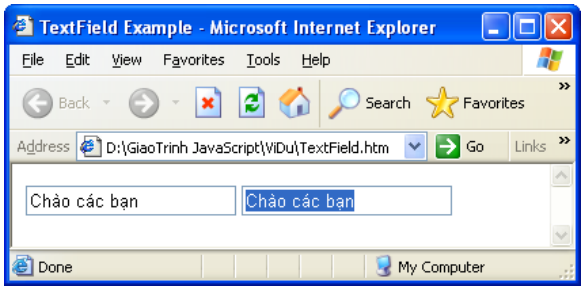
Cách thức và thuộc tính	**Mô tả**
defaultValue	Chỉ ra giá trị mặc định của phần tử được chỉ ra trong thẻ INPUT (thuộc tính)

name	Tên của đối tượng được chỉ ra trong thẻ INPUT (thuộc tính)
value	Giá trị hiện thời của phần tử (thuộc tính)
focus()	Mô tả việc con trỏ tới trường text (cách thức)
blur()	Mô tả việc con trỏ rời trường text (cách thức)
select()	Mô tả việc lựa chọn dòng text trong trường text (cách thức)

Một chú ý quan trọng là có thể gán giá trị cho trường text bằng cách liên kết các giá trị với thuộc tính value. Trong VD sau đây, dòng text được đưa vào trường đầu tiên được lặp lại trong trường text thứ hai, và mọi dòng text được đưa vào trường text thứ hai lại được lặp lại trong trường texxt thứ nhất. Khả năng này của nó có thể áp dụng để tự động cập nhật hoặc thay đổi dữ liệu.

VD (Trang TextField.htm) Tự động cập nhật các trường text .

```
<HTML>
<HEAD>
<TITLE>TextField Example</TITLE>
<SCRIPT LANGUAGE="JavaScript">
    function echo(form,currentField){
        if (currentField == "first")
          form.second.value = form.first.value;
        else
          form.first.value = form.second.value;
      }
</SCRIPT>
</HEAD>
<BODY>
<FORM>
    <INPUT TYPE=text
NAME="first"
onChange="echo(this.form,th
is.name);">
    <INPUT TYPE="text" NAME="second"
onChange="echo(this.form,this.name);">
</FORM>
</BODY>
</HTML>
```



k. Phần tử Textarea

Thẻ TEXTAREA cung cấp một hộp cho phép nhập số dòng text do người thiết kế định trước. VD:

```
<TEXTAREA NAME="fieldName" ROWS="10" COLS="25">
     Default Text Here
</TEXTAREA>
```

VD này tạo ra một trường text cho phép đưa vào 10 hàng ,mỗi hàng 25 ký tự. Dòng "Defautl Text Here"sẽ xuất hiện trong trường này vào lần hiển thị đầu tiên.

Cũng như phần tử text , JavaScript cung cấp cho bạn các thuộc tính defaultValue, name, và value, các cách thức focus(), select(), và blur(), các thẻ sự kiện onBlur, onForcus, onChange, onSelect.

Mảng elements[]

Các đối tượng của form có thể được gọi tới bằng mảng elements[]. VD bạn tạo ra một form sau:

```
<form method="post" name="testform">
     <input type="text" name="one">
     <input type="text" name="two">
     <input type="text" name="three">
</form>
```

Bạn có thể gọi tới ba thành phần này như sau:

```
document.elements[0], document.elements[1],
document.elements[2],
```

Hơn nữa còn có thể gọi

```
document.testform.one,
document.testform.two,
document.testform.three.
```

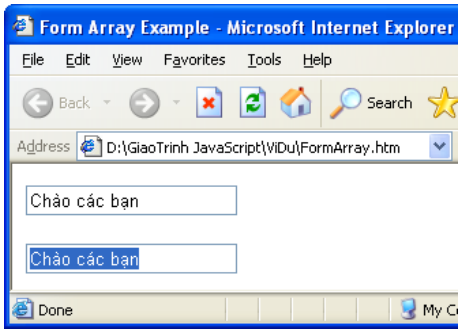
Thuộc tính này thường được sử dụng trong các mối quan hệ tuần tự của các phần tử hơn là dùng tên của chúng.

l. Mảng form[]

Các thẻ sự kiện được thiết kế để làm việc với các form riêng biệt hoặc các trường ở một thời điểm, nó rất hữu dụng để cho phép gọi tới các form có liên quan trong cùng một trang.

Mảng form[] đề cập đến ở đây có thể có nhiều xác định các nhân của form trên cùng một trang và have information in a single field match in all three forms. Có thể gọi bằng document.forms[] thay vì gọi bằng tên form. Trong script này, bạn có hai trường text để nhập và nằm trên hai form độc lập với nhau. Sử dụng mảng form bạn có thể tương tác trên các giá trị của các trường trong hai form cùng một lúc khi người sử dụng thay đổi giá trị trên một form.

```
<body>
<form method=post>
<input type=text
onchange="document.forms[1].elemen
ts[0].value = this.value;">
</form>
<form method=post>
<input type=text
onchange="document.forms[0].elements[0].value = this.value;">
</form>
</body>
```



Mặt khác, bạn cũng có thể truy nhập đến form bằng tên form được đặt trong thẻ FORM:

```
<form method="post" name="name">
```

Khi đó bạn có thể gọi là document.forms["name"] hoặc document.name

8. Đối tượng history

Đối tượng này được sử dụng để lưu giữ các thông tin về các URL trước được người sử dụng sử dụng. Danh sách các URL được lưu trữ theo thứ tự thời gian. Đối tượng này không có chương trình xử lý sự kiện.

Các thuộc tính

length - Số lượng các URL trong đối tượng.

Các phương thức

history.back() - Được sử dụng để tham chiếu tới URL mới được thăm trước đây.

history.forward() - Được sử dụng để tham chiếu tới URL kế tiếp trong danh sách. Nó sẽ không gây hiệu ứng gì nếu đã đến cuối của danh sách.

history.go (delta | "location") - Được sử dụng để chuyển lên hay chuyển xuống delta bậc hay di chuển đến URL xác định bởi location trong danh sách. Nếu delta được sử dụng thì việc dịch chuyển lên phía trên khi delta dương và xuống phía dưới khi delta âm. nếu sử dụng location, URL gần nhất có chứa location là chuỗi con sẽ được tham chiếu.

9. Đối tượng links

Đối tượng link là một đoạn văn bản hay một ảnh được xem là một siêu liên kết. Các thuộc tính của đối tượng link chủ yếu xử lý về URL của các siêu liên kết. Đối tượng link cũng không có phương thức nào.

Mảng link chứa danh sách tất cả các liên kết trong document. Có thể xác định số lượng các link qua : document.links.length()

Có thể tham chiếu tới một liên kết qủa: document.links [index]

Để xác định các thuộc tính của đối tượng link, có thể sử dụng URL tương tự:

http://www.abc.com/chap1/page2.html#topic3

Các thuộc tính

hash	Tên anchor của vị trí hiện thời (VD topic3).
Host	Phần hostname:port của URL (VD www.abc.com). Chú ý rằng đây thường là cổng ngầm định và ít khi được chỉ ra.
hostname	Tên của host và domain (VD ww.abc.com).
href	Toàn bộ URL cho document hiện tại.
pathname	Phần đường dẫn của URL (VD /chap1/page2.html).
port	Cổng truyền thông được sử dụng cho máy tính host, thường là cổng ngầm định.

protocol	Giao thức được sử dụng (cùng với dấu hai chấm) (VD http:).
search	Câu truy vấn tìm kiếm có thể ở cuối url cho các script CGI.
target	Giống thuộc tính target của <link>

Các chương trình xử lý sự kiện

onclick	Xảy ra khi người sử dụng nhấn vào link.
onmouseover	Xảy ra khi con chuột di chuyển qua link.

10. Đối tượng Math

Đối tượng Math là đối tượng nội tại trong JavaScript. Các thuộc tính của đối tượng này chứa nhiều hằng số toán học, các hàm toán học, lượng giác phổ biến. Đối tượng Math không có chương trình xử lý sự kiện.

Việc tham chiếu tới number trong các phương thức có thể là số hay các biểu thức được đnáh giá là số hợp lệ.

Các thuộc tính

`E`	Hằng số Euler, khoảng 2,718.
`LN2`	logarit tự nhiên của 2, khoảng 0,693.
`LN10`	logarit tự nhiên của 10, khoảng 2,302.
`LOG2E`	logarit cơ số 2 của e, khoảng 1,442.
`PI`	Giá trị của □, khoảng 3,14159.
`SQRT1_2`	Căn bậc 2 của 0,5, khoảng 0,707.
`SQRT2`	Căn bậc 2 của 2, khoảng 1,414.
`Log10E`	logarit cơ số 10 của e, khoảng 1,442.

Các phương thức

`abs (number)`	Trả lại giá trị tuyệt đối của number.
`acos (number)`	Trả lại giá trị arc cosine (theo radian) của number. Giá trị của number phải năng giữa 1 và 1.
`asin (number)`	Trả lại giá trị arc sine (theo radian) của number. Giá trị của number phải năng giữa 1 và 1.
`atan (number)`	Trả lại giá trị arc tan (theo radian) của number.
`ceil (number)`	Trả lại số nguyên nhỏ nhất lớn hơn hoặc bằng number.
`cos (number)`	Trả lại giá trị cosine của number.
`exp (number)`	Trả lại giá trị e^ number, với e là hằng số Euler.
`floor (number)`	Trả lại số nguyên lớn nhất nhỏ hơn hoặc bằng number.

`log (number)`	Trả lại logarit tự nhiên của number.
`max (num1,num2)`	Trả lại giá trị lớn nhất giữa num1 và num2
`min (num1,num2)`	Trả lại giá trị nhỏ nhất giữa num1 và num2.
`pos (base,exponent)`	Trả lại giá trị base luỹ thừa exponent.
`random (r)`	Trả lại một số ngẫu nhiên giữa 0 và 1.
`round (number)`	Trả lại giá trị của number làm tròn tới số nguyên gần nhất.
`sin (number)`	Trả lại sin của number.
`sqrt (number)`	Trả lại căn bậc 2 của number.
`tan (number)`	Trả lại tag của number.
`toString(number)`	Chuyển số thành chuỗi.

11. Đối tượng Date

Đối tượng Date là đối tượng có sẵn trong JavaScript. Nó cung cấp nhiều phương thức có ích để xử lý về thời gian và ngày tháng. Đối tượng Date không có thuộc tính và chương trình xử lý sự kiện.

Phần lớn các phương thức date đều có một đối tượng Date đi cùng. Các phương thức giới thiệu trong phần này sử dụng đối tượng Date dateVar, VD:

```
dateVar = new Date ('August 16, 1996 20:45:04');
```

Các phương thức

`getDate()`	Trả lại ngày trong tháng (1-31) cho dateVar.
`getDay()`	Trả lại ngày trong tuần (0=chủ nhật,...6=thứ bảy) cho dateVar.
`getMonth()`	Trả về giá trị tháng
`getFullYear`	Trả về giá trị năm (4 chữ số)
`getHours()`	Trả lại giờ (0-23) cho dateVar.
`getMinutes()`	Trả lại phút (0-59) cho dateVar.
`getSeconds()`	Trả lại giây (0-59) cho dateVar.
`getTime()`	Trả lại số lượng các mili giây từ 00:00:00 ngày 1/1/1970.
`getTimeZoneOffset()`	Trả lại độ dịch chuyểnbằng phút của giờ địa phương hiện tại so với giờ quốc tế GMT.
`getYear()`	Trả lại năm cho dateVar.
`parse (dateStr)`	Phân tích chuỗi dateStr và trả lại số lượng các mili giây tính từ 00:00:00 ngày 01/01/1970.

`setDate(date)`	Đặt ngày trong tháng là date
`setHours(hours)`	Đặt giờ là hours
`setMinutes(minutes)`	Đặt phút là minutes cho dateVar.
`setMonths(months)`	Đặt tháng là months cho dateVar.
`setSeconds(seconds)`	Đặt giây là seconds cho dateVar.
`setTime(value)`	Đặt thời gian là value, trong đó value biểu diễn số lượng mili giây từ 00:00:00 ngày 01/01/10970.
`setYear(years)`	Đặt năm là years
`toGMTString()`	Trả lại chuỗi biểu diễn ngày dưới dạng GMT.
`toLocaleString()`	Trả lại chuỗi biểu diễn dateVar theo khu vực thời gian hiện thời.
`getUTCDate ()`	Trả lại số lượng mili giây từ 00:00:00 01/01/1970 GMT.

12. Đối tượng String

Đối tượng String là đối tượng được xây dựng nội tại trong JavaScript cung cấp nhiều phương thức thao tác trên chuỗi. Đối tượng này có thuộc tính duy nhất là độ dài (length) và không có chương trình xử lý sự kiện.

Các phương thức

`anchor(name)`	Được sử dụng để tạo ra thẻ <A> (một cách động). Tham số name là thuộc tính NAME của thẻ <A>.
`big()`	Kết quả giống như thẻ <BIG> trên chuỗi str.
`blink()`	Kết quả giống như thẻ <BLINK> trên chuỗi str.
`bold()`	Kết quả giống như thẻ <BOLD> trên chuỗi str.
`charAt(index)`	Trả lại ký tự thứ index trong chuỗi str.
`fontcolor()`	Kết quả giống như thẻ <FONTCOLOR = color>.
`fontsize(size)`	Kết quả giống như thẻ <FONTSIZE = size>.
`indexOf(srchStr [,index])`	Trả lại vị trí trong chuỗi str vị trí xuất hiện đầu tiên của chuỗi srchStr. Chuỗi str được tìm từ trái sang phải. Tham số index có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.
`italics()`	Kết quả giống như thẻ <I> trên chuỗi str.
`lastIndexOf(srchStr [,index])`	Trả lại vị trí trong chuỗi str vị trí xuất hiện cuối cùng của chuỗi srchStr. Chuỗi str được tìm từ phải sang trái. Tham số index có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.

link(url)	Được sử dụng để tạo ra một kết nối HTML động cho chhuỗi str. Tham số href là URL đích của liên kết.
strike()	Kết quả giống như thẻ <STRIKE> trên chuỗi str.
sub()	Tạo ra một subscript cho chuỗi str, giống thẻ <SUB>.
substr(a,b)	Trả lại chuỗi con của str là các ký tự từ vị trí thứ a tới vị trí thứ b. Các ký tự được đếm từ trái sang phải bắt đầu từ 0.
sup()	Tạo ra superscript cho chuỗi str, giống thẻ <SUP>.
toLowerCase()	Đổi chuỗi str thành chữ thường.
toUpperCase()	Đổi chuỗi str thành chữ hoa.

13. Đối tượng Image

Hình ảnh là một đối tượng không thể thiếu trên trang web. Javascript cung cấp đối tượng Image cho phép ta điều khiển các hình ảnh trên trang web.

Khởi tạo đối tượng Image mới :

```
var img = new Image();
```

Truy xuất đến 1 Image trên trang :

```
document.images[image_name]
document.images[index]
```

Các thuộc tính

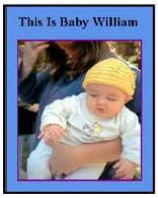
border	Độ lớn của border
height	Chiều cao của ảnh
hspace	Khoảng trống bên trên và dưới của ảnh
name	Tên của hình
prototype	Phương thức định nghĩa mở rộng
src	Đường dẫn ảnh
vspace	Khoảng trống bên trái và phải của ảnh
width	Độ rộng của ảnh

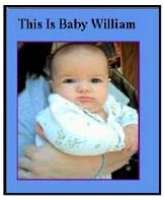
Ví dụ : tạo hiệu ứng chuyển ảnh khi đưa chuột vào

```
<html>
    <head><title>Preloading Images</title></head>
    <h2>  This Is Baby William</h2>
    <script language="JavaScript">
        if(document.images){
            var baby1=new Image();      // Preload an image
            baby1.src="baby1image.jpg";
        }
        if (document.images){
            var baby2=new Image();     // Preload an image

            baby2.src="baby2image.jpg";
        }
```

```
</script>
<body bgcolor="cornflowerblue">
<a href="#" onMouseOver="document.willy.src=baby2.src;"
            onMouseOut="document.willy.src=baby1.src;">
<img name="willy"
    src="baby1image.jpg"
    align="left"
    alt="baby" border=2  hspace="10"
    width="220" height="250">
</body>
</html>
```





14. Đối tượng Link

Các link (liên kết) là nền tảng của Web, cho phép người xem di chuyển giữa các trang với nhau.

Truy xuất đến 1 Image trên trang :

```
document.links[image_name]
document.links[index]
```

Các thuộc tính

hash	port
host	protocol
hostname	search
href	target
pathname	

III. Bảng tổng kết các từ khoá

Sau đây là các từ đựoc định nghĩa là một phần trong ngôn ngữ JavaScript và không được sử dụng là tên biến:

abstract	eval	int	static
boolean	extends	interface	super
break	false	long	switch
byte	final	native	synchrinized
case	finally	new	this
catch	float	null	throw
char	for	package	throws
class	function	parseFloat	transient
const	goto	parseInt	true
continue	if	private	try
default	implements	protected	var
do	import	public	void
double	in	return	while
else	instanceof	short	with