Chương 01

TỔNG QUAN VỀ JAVASCRIPT

- ✓ *Giới Thiệu*
- ✓ *Nhúng JavaScript vào trang Web*
- ✓ *Các lệnh cơ bản*

I. Giới thiệu

Với HTML sẽ cho ta 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, trình duyệt đọc JavaScript dưới dạng mã nguồn. Chính vì vậy ta có thể dễ dàng học JavaScript trên các trang Web có sử dụng JavaScript.

JavaScript là ngôn ngữ dựa trên đối tượng, 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.

II. Nhúng JavaScript vào File HTML

Sử dụng một trong các cách sau:

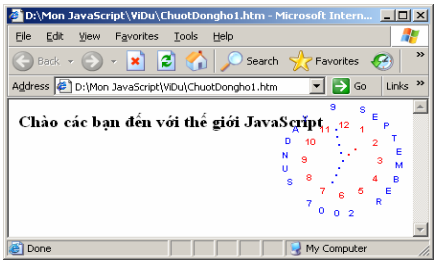
- Sử dụng các câu lệnh và các hàm trong cặp thẻ <SCRIPT>
- Sử dụng các File nguồn JavaScript
- Sử dụng một biểu thức JavaScript làm giá trị của một thuộc tính HTML
- Sử dụng thẻ sự kiện (event handlers) trong một thẻ HTML nào đó

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

1. Nhúng JavaScript vào trang HTML

JavaScript được đưa vào File HTML bằng cách sử dụng cặp thẻ <Script> và </Script>. Nếu đặt trong phần <Head>, nó sẽ được tải và sẵn sàng trước khi phần còn lại của văn bản được tải.Sử dụng cú pháp sau :

```
<Script >
        // Chèn các mã Javacript vào đây
</Script>
```



Ví dụ: Tạo trang web (Clock1.htm) sử dụng nhúng mã JavaScript trực tiếp vào trang

Ghi chú: Có thể sưu tầm các mã JavaScript từ Website http://www.javascriptbank.com.vn, www.echip.com.vn

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>
```

Ví dụ: <Script Src=" Clock2.js ">

Các File JavaScript bên ngoài chỉ chứa các câu lệnh JavaScript và định nghĩa hàm. Tên File của các hàm JavaScript bên ngoài cần có đuôi .js,

Ví dụ: Tạo trang web(Clock.htm) sử dụng nhúng mã JavaScript thông qua 1 tập tin Javascript .

III. Các lệnh cơ bản

1. Cú pháp cơ bản của lệnh :

JavaScript xây dựng 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;
```

VÍ Dụ: document.write("Chào các bạn!
");

2. Hiển thị một dòng văn bản

Đối tượng document trong JavaScript được thiết kế sẵn hai phương thức để xuất một dòng văn bản ra màn hình client: write() và writeln().

```
document.write(“Chuỗi văn bản”);
```

Ví dụ: document.write("Chào các bạn");

document.writeln(“Chúc các bạn vui vẽ!”);

Phương thức write(): Xuất ra màn hình dòng văn bản nhưng không xuống dòng

Phương thức writeln(): Sau khi viết xong dòng văn bản tự động xuống dòng.

Ghi chú: Có thể dùng “+” để ghép nhiều chuỗi ký tự.

Cho phép dùng các kí tự đặc biệt trong chuỗi:

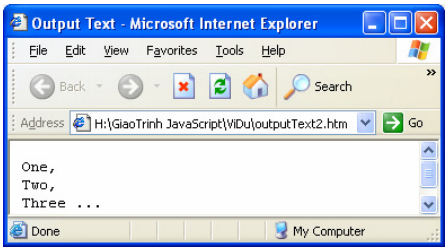
\n : Xuống dòng

\t : Tab

Khi có dùng các ký tự đặc biệt hoặc lệnh Writeln thì phải đặt khối JavaScript trong cặp thẻ <Pre> . . </Pre> (Thẻ quy định văn bản định dạng trước)

Ví dụ: Tạo trang (OutputText.htm) để 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>
```



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

Cú pháp:

```
alert("Câu thông báo");
```

Khi đó sẽ chờ cho đến khi người sử dụng nhấn vào nút OK . Thông thường, cách thức alert() được sử dụng trong các trường hợp:

- Thông tin đưa vào form không hợp lệ
- Kết quả sau khi tính toán không hợp lệ
- Khi dịch vụ chưa sẵn sàng để truy nhập dữ liệu

Ví dụ: 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()

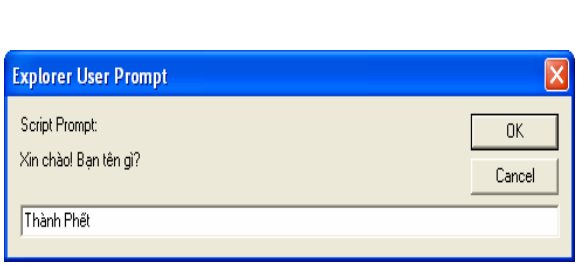
Một hộp thoại gồm 1 dòng thông báo, 1 trường nhập dữ liệu, 1 nút OK và 1 nút Cancel. Người sử dụng nhập vào trường đó rồi kích vào OK. Khi đó, ta có thể xử lý dữ liệu vừa đưa vào.

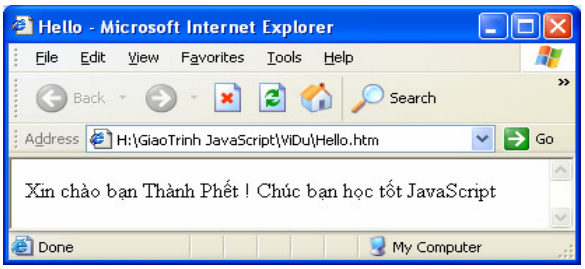
Cú pháp:

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

Ví dụ: Tạo trang (Hello.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=window.prompt("Xin chào!Bạn tên gì?","");
document.write("Xin chào bạn " + name + " ! Chúc bạn học tốt JavaScript ");
</Script>
</Body>
```





5. Hỏi đáp người sử dụng – Lệnh confirm()

Lệnh **confirm()** tạo ra 1 hộp thoại gồm 1 dòng thông báo, nút OK và nút Cancel. Người sử dụng có thể click vào OK. Khi đó sẽ 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

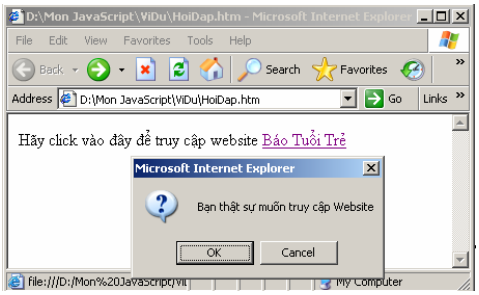
Cú pháp:

```
confirm("Câu thông báo hỏi ?");
```

Ví dụ: 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:Báo Tuổi Trẻ



Chương 02

NGÔN NGỮ KỊCH BẢN JAVASCRIPT

- ✓ *Biến và khai báo biến*
- ✓ *Kiểu dữ liệu*
- ✓ *Lệnh, khối lệnh*
- ✓ *Toán tử và biểu thức*
- ✓ *Cấu trúc lập trình*
- ✓ *Mảng*
- ✓ *Hàm*

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

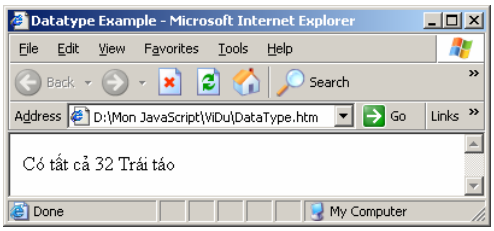
- Biến toàn cục: Có thể được truy cập từ bất kỳ đâu trong ứng dụng. Được khai báo: x = 0;
- Biến cục bộ: Chỉ được truy cập trong phạm vi chương trình mà nó khai báo. Biến cục bộ được khai báo trong một hàm với từ khoá var: var x = 0;

II. Kiểu dữ liệu

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

Ví dụ: Tạo trang (DataType.htm) như sau

```
<HTML>
<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>
```



Trong JavaScript, có bốn kiểu dữ liệu sau đây:

1. Kiểu nguyên (Interger)

Số nguyên có thể được biểu diễn theo ba cách: Hệ cơ số 10 (hệ thập phân), Hệ cơ số 8 (hệ bát phân) và Hệ cơ số 16 (hệ thập lục phân) -Với hai chữ số đầu tiên là 0x. (VÍ Dụ: 0x5F)

2. Kiểu dấu phẩy động (Floating 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.

Ví dụ: 9.87 hay -0.85E4

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 biểu diễn bởi không hay nhiều ký tự đặt trong cặp dấu " ... " hay '... '

Ví dụ:

"The dog ran up the tree" hay "100"

Ghi chú: Để biểu diễn dấu nháy kép ("), trong chuỗi sử dụng (\"),

Ví dụ:

document.write(" \"This text inside quotes.\" ");

III. Lệnh, khối lệnh trong JavaScript

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 đặt trong cặp ngoặc nhọn: { . . . }

Bên trong một khối lệnh có thể chứa một hay nhiều 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. Tóan tử & Biểu thức trong JavaScript

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). Về cơ bản có ba kiểu biểu thức:

- Số học: Nhằm để lượng giá giá trị số. VÍ Dụ: (3+4)+(84.5/3) bằng 197.1666666667.
- Chuỗi: Nhằm để đánh giá chuỗi. VÍ Dụ: "The dog"+"barked!" là "The dog barked!"
- Logic: Nhằm đánh giá giá trị logic. VÍ Dụ: 23>32 là False.

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

(condition) ? valTrue : valFalse

Nếu điều kiện condition là đúng, biểu thức nhận giá trị valTrue, ngược lại nhận giá trị là False.

Ví dụ:

ketqua = (diemtb>=5) ? "Đậu" : "Rớt"

Trong ví dụ này biến ketqua được gán giá trị "Đậu" nếu giá trị của biến tdiemtb lớn hơn hoặc bằng 5; ngược lại nó nhận giá trị "Rớt".

2. Các Toán tử.

Toán tử được sử dụng để thực hiện một phép toán. Được nhóm thành các loại sau đây: gán, so sánh, số học, chuỗi và logic.

=	Gán giá trị của toán hạng bên phải cho toán hạng bên trái.
==	(Bằng)Trả lại giá trị đúng nếu toán hạng bên trái bằng toán hạng bên phải
!=	(Khác)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
>=	(Lớn hơn hoặc bằng)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
<=	(Nhỏ hơn hoặc bằng)Trả lại giá trị đúng nếu toán hạng bên trái nhỏ hơn hoặc bằng toán hạng bên phải
var1 % var2	(Chia lấy phần dư) Trả lại phần dư khi chia var1 cho var2
-	(Phủ định) Cho 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)
+	Kết hợp hai chuỗi
expr1 && expr2	Toán tử AND trả về giá trị đúng nếu expr1 và expr2 cùng đúng.
expr1 || expr2	Toán tử OR trả về giá trị đúng nếu ít nhất 1 trong 2 expr1,expr2 đúng.

V. Cấu trúc lập trình

Có thể chia các cấu trúc lập trình của JavaScript thành 2 nhóm sau:Cấu trúc rẽ nhánh (Điều kiện) và Cấu trúc lặp.

1. Cấu trúc lập trình rẽ nhánh (Điều Kiện)

Cú pháp:

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

Ví dụ: Tạo trang (CauTrucDK.htm) Sử dụng phương pháp confirm() với phát biểu if

```
<HTML>
<Head> <Title>Cấu Trúc Điều Kiện</Title>
<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;
</Script>
</Head> <Body>
<Script Language="Javascript">
        document.write(output);
</Script>
</Body> </HTML>
```

2. Cấu trúc lặp

a. Vòng lặp For

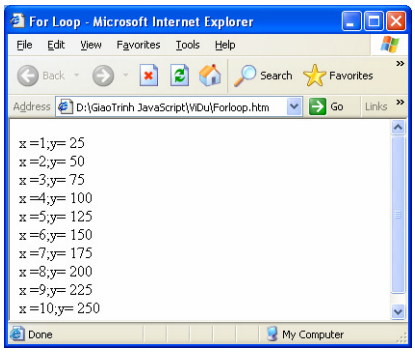
Vòng lặp for thiết lập 1 biểu thức khởi đầu - initExpr, sau đó lặp 1 đoạn mã cho đến khi biểu thức <điều kiện> được đánh giá là đúng. Sau khi kết thúc mỗi vòng lặp, biểu thức incrExpr được đánh giá lại.

Cú pháp:

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

Ví dụ: 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

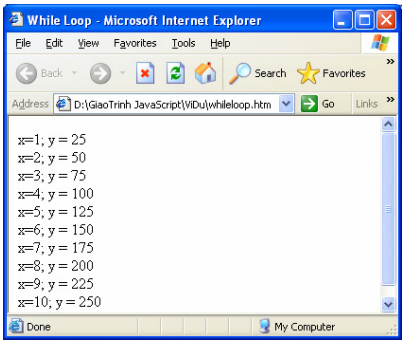
Vòng lặp while lặp khối lệnh chừng nào <điều kiện> còn được đánh giá là đúng

Cú pháp:

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

Ví dụ: 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 ví dụ này giống như ví dụ 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;

Ví dụ: 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

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;

Ví dụ: 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

Mặc dù JavaScript không hỗ trợ cấu trúc dữ liệu mảng nhưng tạo ra phương thức cho phép bạn tự tạo ra các hàm khởi tạo mảng như sau:

```
function taomang(n) {
        this.length = n;
        for (var i=1; i<=n; i++){
                this[i]=0
        }
        return this;
}
```

Tạo ra 1 mảng với kích thước xác định trước (n) và điền các giá trị 0.

Ví dụ: a = new taomang(10);

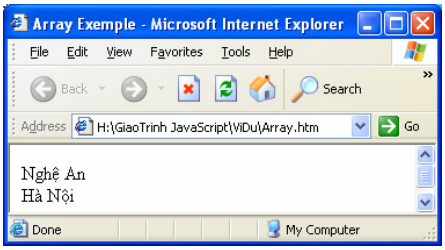
Tạo ra các thành phần từ a[1] đến a[10] với giá trị là 0. Gán giá trị cho các thành phần :

```
a[1] = "Nghệ An";
a[2] = "Hà Nội";
```

Ví dụ: Tạo trang (Array.htm)

```
<HTML> <Head>
<Title> Array Exemple </Title>
<Script Language= "JavaScript">
function taomang(n) {
        this.length = n;
        for (var i=1; i<=n; i++){
                this[i]=0
        }
```

```
            return this;
        }
        a = new taomang(10);
        a[1] = "Nghệ An";
        a[2] = "Hà Nội";
        document.write(a[1] + "<BR>");
        document.write(a[2] + "<BR>");
        </Script>
        </Head>
        <Body> </Body>
        </HTML>
```



VII. Hàm - Function

1. Giới thiệu

Trong lập trình sử dụng hàm là để thực hiện một đoạn chương trình nào đó. Trong Javascript có các hàm được xây dựng sẵn để giúp thực hiện một chức năng và ta cũng có thể định nghĩa ra các hàm khác để thực hiện một công việc nào đó.

Hàm có thể có 1 hay nhiều tham số truyền vào và 1 giá trị trả về. Hàm có thể là thuộc tính của 1 đối tượng, trong trường hợp này nó được xem như là phương thức của đối tượng đó.

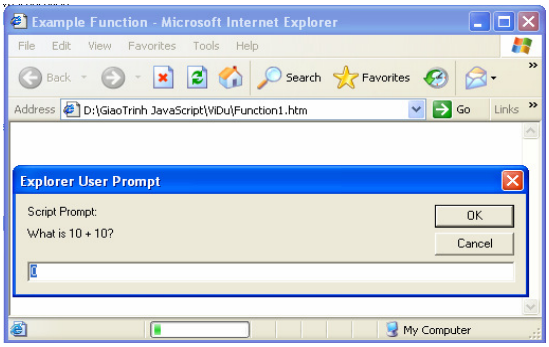
2. Định Nghĩa Hàm

Cú pháp:

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

Ví dụ: Tạo trang (Function.htm)

```
<HTML> <Head> <Title>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 thành giá trị số eval("10*10") trả về giá trị là 100

3. 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

a. Hàm eval

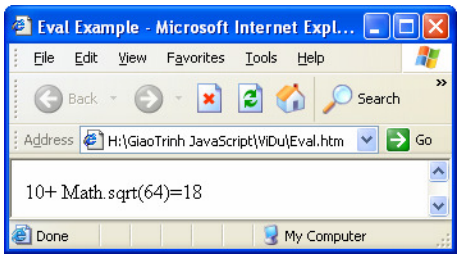
Chuyển đổi giá trị chuỗi thành giá trị số.

Cú pháp:

returnval=eval (biểu thức)

Ví dụ: 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>
```



b. Hàm parseInt

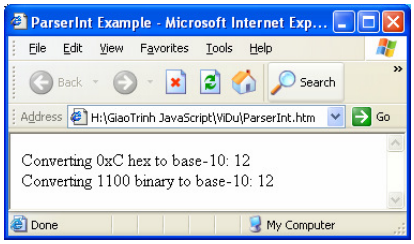
Hàm này chuyển một chuỗi số thành số nguyên với cơ số là tham số thứ hai.

Cú pháp:

parseInt (string, [, radix])

Ví dụ: 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>
```



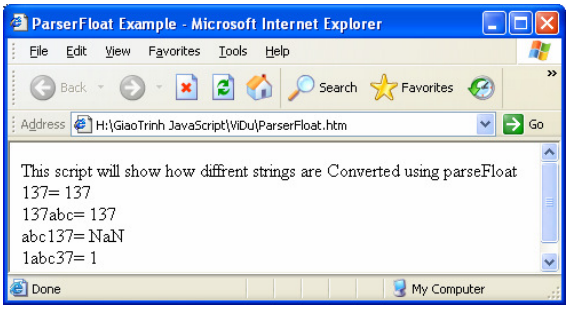
c. Hàm parseFloat

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

Cú pháp: parseFloat (string)

Ví dụ: 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>
```



Chương 03

ĐỐI TƯỢNG & SỰ KIỆN

- ✓ *Đối tượng và thao tác trên đối tượng*
- ✓ *Sự kiện và xử lý sự kiện*
- ✓ *Các đối tượng thường dùng*

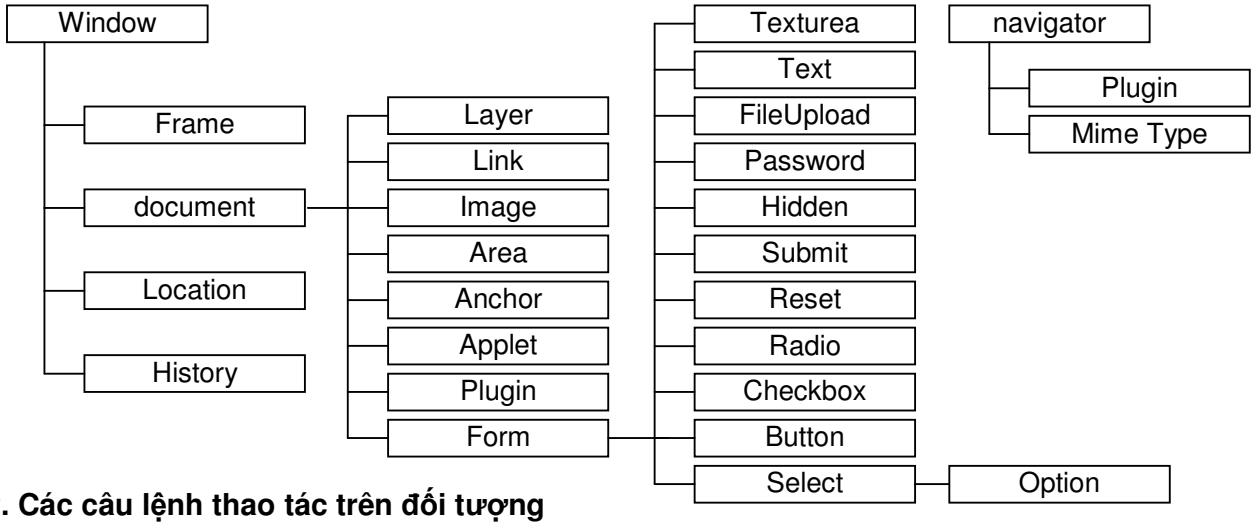
I. Khái Niệm Đối Tượng

1. Khái Niệm Về Đôi Tượng

JavaScript là ngôn ngữ lập trình dựa trên đối tượng, nhưng không hướng 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 cha.

Vi dụ chương trình xử lý sự kiện trên form tên frmDieutra là thuộc tính của đối tượng document và trường text txtAge là thuộc tính của form frmDieutra. Để tham chiếu đến giá trị của txtAge phải sử dụng: **document.frmDieucha.txtAge.value**

Sơ đồ sau sẽ minh hoạ sự phân cấp của các đối tượng



2. Các câu lệnh thao tác trên đối tượng

a. Lệnh For...in

Câu lệnh này được sử dụng biêt tất cả các thuộc tính (properties) của một đối tượng.

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

Ví dụ: Tạo trang (ForIn.htm) in tất cả các thuộc tính của đối tượng Window.

```
<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>
```

b. Biến new

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

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

c. Từ Khóa This

Từ khoá this được sử dụng để chỉ đối tượng hiện thời.

```
this [.property]
```

d. Lệnh With

Lệnh này được sử dụng để thiết lập đối tượng ngầm định cho một nhóm các lệnh.

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

Ví dụ: Tạo trang (Object.htm) minh hoạ cách tạo và sử dụng biến New, từ khoa this và lệnh with.

```
<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>
```

II. Sự Kiện & Xư Ly Sự Kiện

1. Khái niệm sự kiện và xư ly sự kiện

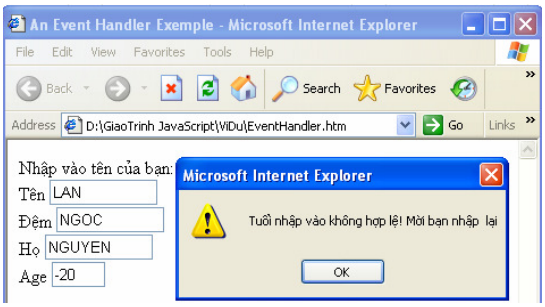
JavaScript là ngôn ngữ định hướng sự kiện, nghĩa là sẽ phản ứng trước các sự kiện như: Click chuột . . .

Chương trình xử lý sự kiện (Event handler) là 1 đoạn mã hay 1 hàm được thực hiện để phản ứng trước 1 sự kiện được xác định là một thuộc tính của một thẻ HTML:

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

Ví dụ: Trang EventHander.htm thẩm định giá trị đưa vào trong trường text Tuổi phai hợp lệ nếu sẽ xuất hiện thông báo yêu cầu nhập lại.

```
<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="frmDieutra">
        Nhập vào tên của bạn:<BR>
        Tên <Input Type=Text Name="TEN" ><BR>
        Đệm <Input Type=Text Name="DEM" ><BR>
        Họ  <Input Type=Text Name="HO" ><BR>
        Age <Input Type=Text Name="AGE" onChange="CheckAge(frmDieutra)"><BR>
        <Input Type=Submit Value=”Submit”>
        <Input Type=Reset Value=”Reset”>
</Form>
</Body></HTML>
```



2. Một số sự kiện trong JavaScript:

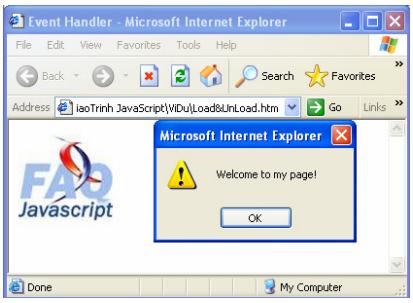
onBlur	Xảy ra khi input focus bị xoá từ thành phần form
onClick	Xảy ra khi người dùng kích vào các thành phần hay liên kết của form.
onChange	Xảy ra khi giá trị của thành phần được chọn thay đổi
onFocus	Xảy ra khi thành phần của form được focus(làm nổi lên).
onLoad	Xảy ra trang Web được tải.
onMouseOver	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

3. các sự kiện có sẵn của một số đối tượng.

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

Ví dụ: Trang LoadUnLoad.htm

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



III. CÁC ĐỐI TƯỢNG THƯỜNG DÙNG

1. Đối tượng window

Đối tượng window là đối tượng ở mức cao nhất. Các đối tượng document, frame, location đều là thuộc tính của đối tượng window.

Các thuộc tính:

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.
Length	Số lượng các frame trong cửa sổ cha.

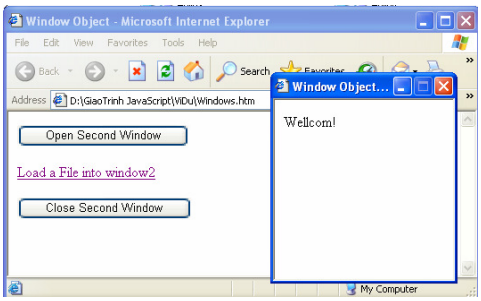
Name	Tên của cửa sổ hiện thời.
Parent	Đối tượng cửa sổ chA
Self	Cửa sổ hiện thời.
Status	Được sử dụng thông báo tạm thời hiển thị lên trên thanh trạng thái cửa sổ.
Top	Cửa sổ ở trên cùng.
Window	Cửa sổ hiện thời.

Các phương thức

alert ("message")	Hiển thị hộp thoại với chuỗi "message" và nút OK.
clearTimeout(timeoutID)	Xóa timeout do SetTimeout đặt. SetTimeout trả lại timeoutID
WindowReference.close	Đóng cửa sổ windowReference.
confirm("message")	Hiển thị hộp thoại với chuỗi "message", nút OK và nút Cancel. Trả lại trị True cho OK và False cho Cancel.
[windowVar =][window]. open("URL", "windowName", ["windowFeatures"])	Mở cửa sổ mới.
prompt ("message" [,"defaultInput"])	Mở hộp hội thoại để nhận dữ liệu vào trường text.
TimeoutID = setTimeout(expression,msec)	Đánh giá biểu thức expresion sau thời gian msec.

Ví dụ: TrangWindows.htm nút thứ nhất để mở 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')">
<BR><A HREF="doc.htm" TARGET="window2"> Load a File into window2 </A>
<Input Type="button" VALUE="Close Second Window" onClick="msgWindow.close()">
</Form>
</Body>
</HTML>
```



2. Đối tượng forms

Các form được tạo ra nhờ cặp thẻ <FORM> . . . </FORM>. Có một vài phần tử (elements) của đối tượng forms như: Button, checkbox, password, radio, reset, select, submit, text, textarea

Các thuộc tính

Action	thuộc tính ACTION của thẻ FORM.
Elements	Mảng chứa các thành phần trong form (như checkbox, textBOX . .
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

formName.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.

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

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 cho phép 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 được đặt tên để JavaScript truy cập đến chúng qua

Thuộc tính **Ttype** : Đó là một xâu chỉ định rõ kiểu của phần tử được đưa vào như nút bấm, một trường text hay một checkbox... 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 vì dữ liệu trong form phải được gửi tới một địa chỉ URL để xử lý và lưu trữ. Một phần tử button được chỉ định rõ khi sử dụng thẻ INPUT:

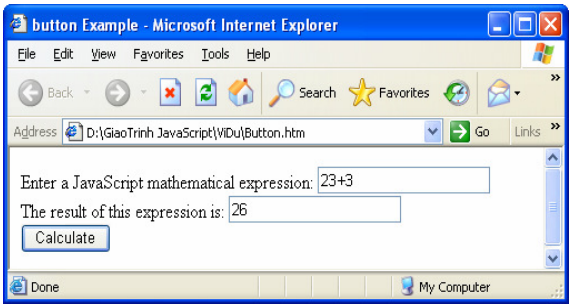
<INPUT TYPE="button" NAME="name" VALUE= "buttonName">

“name” là tên của button, “buttonname” là nhãn của button sẽ được hiển thị.

Chỉ có một sự kiện duy nhất là onClick.

Ví dụ: Trang Button.htm Định giá trị trong form sử dụng phần tử button.

```
<HTML>
<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></HTML>
```



b. Phần tử checkbox

Các phần tử checkbox có khả năng bật tắt dùng để chọn hoặc không chọn một thông tin.

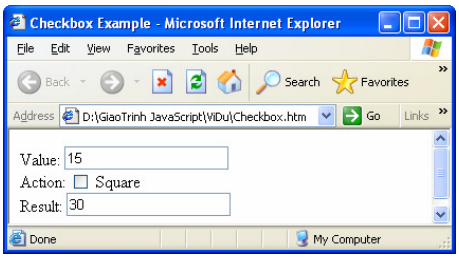
Bang danh sách các thuộc tính và các phương thức

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

Ví dụ: Trang Checkbox.htm Tạo hộp checkbox để nhập vào một số lựa chọn:

```
<HTML><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></HTML>
```



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.

d. Phần tử hidden

Phần tử hidden là phần tử 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 nhưng nó không được hiển thị trên trang.

e. Phần tử Password

Đối tượng Password là đối tượng mà khi gõ bất kỳ ký tự nào vào cũng đều hiển thị dấu sao(*). Dùng để nhập những thông tin bí mật như mật khẩu...

f. Phần tử radio

Đối tượng radio gần giống sự bật tắt checkbox. Khi nhiều radio được kết hợp thành một nhóm, chỉ có một nút được chọn trong bất kỳ một thời điểm nào.

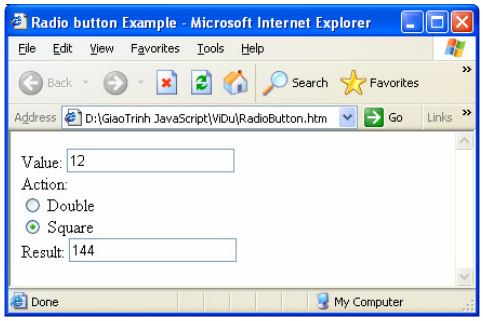
Nhóm các nút radio lại bằng cách đặt cho chúng có cùng một tên trong các thẻ INPUT. Bảng sau hiển thị các thuộc tính và cách thức của đối tượng radio.

Thuộc tính và cách thức	Mô tả
checked	Mô tả trạng thái hiện thời của phần tử radio
defaultChecked	Mô tả trạng thái mặc định của phần tử
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
value	Mô tả giá trị hiện thời của phần tử được định ra trong thẻ INPUT
click()	Mô phỏng một click trên nút radio (cách thức)

Ví dụ: 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>
```

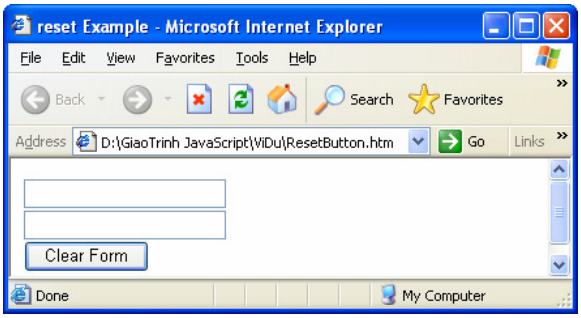


g. Phần tử reset

Sử dụng đối tượng reset, cũng giống đối tượng button, đối tượng reset có hai thuộc tính là name và value và một sự kiện onClick. Đối tượng reset dùng để xoá form.

Ví dụ: Trang ResetButton.htm minh hoạ cách sử dụng nút reset để xoá các giá trị của form.

```
<HTML>
<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></HTML>
```



h. Phần tử select

Danh sách lựa chọn trong các form 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>1
   <OPTION>2
   <OPTION>3
</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>1
   <OPTION>2
   <OPTION>3
</SELECT>
```

Trong cả hai VÍ DỤ: 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>
      <OPTION SELECTED>1
      <OPTION>2
      <OPTION>3
</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.

Ví dụ: Có danh sách lựa chọn sau:

```
<Select Name="example" onFocus="react();">
   <Option SELECTED VALUE="Number One">1
   <Option VALUE="The Second">2
   <Option VALUE="Three is It">3
</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
```

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.

j. Phần tử Text

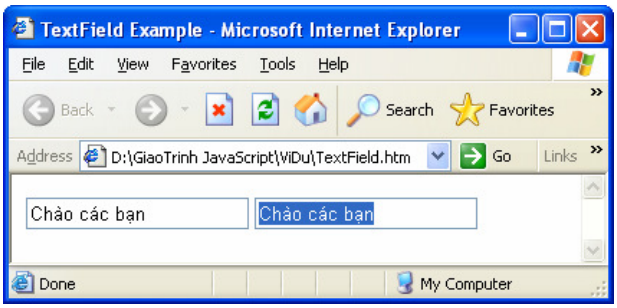
Phần tử này nằm trong những phần tử hay được sử dụng nhất trong các form HTML. Trường text cho phép nhập vào một dòng đơn.

Bảng sau mô tả các thuộc tính và phương thức.

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)

Ví dụ: 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,this.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. VÍ Dụ:

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

VÍ DỤ: 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[]. VÍ DỤ: 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.

3. 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.

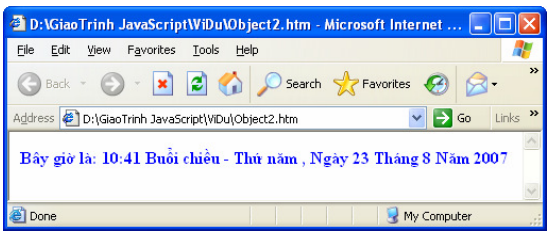
Các phương thức

dateVar.getDate()	Trả lại ngày trong tháng (1-31) cho dateVar.
dateVar.getDay()	Trả lại ngày trong tuần (0=chủ nhật,...6=thứ bảy) cho dateVar.
dateVar.getHours()	Trả lại giờ (0-23) cho dateVar.
dateVar.getMinutes()	Trả lại phút (0-59) cho dateVar.
dateVar.getSeconds()	Trả lại giây (0-59) cho dateVar.
dateVar.getTime()	Trả lại số lượng các mili giây từ 00:00:00 ngày 1/1/1970.
dateVar.getTimeZoneOffset()	Trả lại độ dịch chuyểnbằng phút của giờ địa phương hiện tại so với giờ quốc tế GMT.
dateVar.getYear()	Trả lại năm cho dateVar.
Date.parse (dateStr)	Phân tích chuỗi dateStr và trả lại số lượng các mili giây tính từ 00:00:00 ngày 01/01/1970.
dateVar.setDay(day)	Đặt ngày trong tháng là day cho dateVar.
dateVar.setHours(hours)	Đặt giờ là hours cho dateVar.
dateVar.setMinutes(minutes)	Đặt phút là minutes cho dateVar.
dateVar.setMonths(months)	Đặt tháng là months cho dateVar.
dateVar.setSeconds(seconds)	Đặt giây là seconds cho dateVar.
dateVar.setTime(value)	Đặt thời gian là value, trong đó value biểu diễn số lượng mili giây từ 00:00:00 ngày 01/01/1970.
dateVar.setYear(years)	Đặt năm là years cho dateVar.
dateVar.toGMTString()	Trả lại chuỗi biểu diễn dateVar dưới dạng GMT.
dateVar.toLocaleString()	Trả lại chuỗi biểu diễn dateVar theo khu vực thời gian hiện thời.

Date.UTC (year, month, day [,hours] [,minutes] [,seconds])	Trả lại số lượng mili giây từ 00:00:00 01/01/1970 GMT.

Ví dụ:Tạo trang (DatTime.htm)

```
<head><meta content="charset=unicode"></head>
<Script Language="JavaScript">
        mydate = new Date();
        myday = mydate.getDay() ;
        mymonth = mydate.getMonth()+1;
        myweekday= mydate.getDate();
        weekday= myweekday;
        myyear= 1900  + mydate.getYear();
        myhours = mydate.getHours();
        ampmhour = (myhours > 12) ? myhours - 12 : myhours;
        ampm = (myhours >= 12) ? 'Buổi chiều ' : ' Buổi sáng ';
        myminutes = 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>
```



4. 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

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.

Các phương thức

Math.abs (number)	Trả lại giá trị tuyệt đối của number.
Math.acos (number)	Trả lại giá trị arc cosine (theo radian) của number. Giá trị của number phải năng giữa 1 và 1.
Math.asin (number)	Trả lại giá trị arc sine (theo radian) của number. Giá trị của number phải năng giữa 1 và 1.
Math.atan (number)	Trả lại giá trị arc tan (theo radian) của number.
Math.ceil (number)	Trả lại số nguyên nhỏ nhất lớn hơn hoặc bằng number.
Math.cos (number)	Trả lại giá trị cosine của number.
Math.exp (number)	Trả lại giá trị e^ number, với e là hằng số Euler.
Math.floor (number)	Trả lại số nguyên lớn nhất nhỏ hơn hoặc bằng number.
Math.log (number)	Trả lại logarit tự nhiên của number.
Math.max (num1,num2)	Trả lại giá trị lớn nhất giữa num1 và num2
Math.min (num1,num2)	Trả lại giá trị nhỏ nhất giữa num1 và num2.
Math.pos (base,exponent)	Trả lại giá trị base luỹ thừa exponent.
Math.random (r)	Trả lại một số ngẫu nhiên giữa 0 và 1. Phwong thức này chỉ thực hiện được trên nền tảng UNIX.
Math.round (number)	Trả lại giá trị của number làm tròn tới số nguyên gần nhất.
Math.sin (number)	Trả lại sin của number.
Math.sqrt (number)	Trả lại căn bậc 2 của number.
Math.tan (number)	Trả lại tag của number.

5. 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.

Các phương thức

str.anchor (name)	Được sử dụng để tạo ra thẻ <A> (một cách động). Tham số name là thuộc tính NAME của thẻ <A>.
str.big()	Kết quả giống như thẻ <BIG> trên chuỗi str.
str.blink()	Kết quả giống như thẻ <BLINK> trên chuỗi str.
str.bold()	Kết quả giống như thẻ <BOLD> trên chuỗi str.
str.charAt(a)	Trả lại ký tự thứ a trong chuỗi str.
str.fixed()	Kết quả giống như thẻ <TT> trên chuỗi str.
str.fontcolor()	Kết quả giống như thẻ <FONTCOLOR = color>.
str.fontsize(size)	Kết quả giống như thẻ <FONTSIZE = size>.
str.index0f(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
str.italics()	Kết quả giống như thẻ <I> trên chuỗi str.
str.lastIndex0f(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.
str.link(href)	Được sử dụng để tạo ra một kết nối HTML động cho chhuỗi str. Tham số href là URL đích của liên kết.
str.small()	Kết quả giống như thẻ <SMALL> trên chuỗi str.
str.strike()	Kết quả giống như thẻ <STRIKE> trên chuỗi str.
str.sub()	Tạo ra một subscript cho chuỗi str, giống thẻ <SUB>.
str.substring(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.
str.sup()	Tạo ra superscript cho chuỗi str, giống thẻ <SUP>.
str.toLowerCase()	Đổi chuỗi str thành chữ thường.

str.toUpperCase()	Đổi chuỗi str thành chữ hoa.

6. 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.

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.

history.go (delta | "location") - Được sử dụng để chuyển lên hay chuyển xuống delta bậc hay di chuyển đến URL xác định bởi location trong danh sách. Dịch chuyển lên phía trên khi delta dương và xuống phía dưới khi delta âm.

7. 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.

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

Các thuộc tính

hash	Tên anchor của vị trí hiện thời (VÍ DỤ: topic3).
Host	Phần hostname:port của URL (VÍ DỤ: www.abc.com).
Hostname	Tên của host và domain (VÍ DỤ: ww.abc.com).
href	Toàn bộ URL cho document hiện tại.
Pathname	Phần đường dẫn của URL (VÍ DỤ: /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) (VÍ DỤ: http:).
Search	Câu truy vấn tìm kiếm có thể ở cuối URL cho các script CGI.
Target	Giống thuộc tính TARGET của <LINK>

8. Đối tượng location

Các thuộc tính của đối tượng location duy trì các thông tin về URL của document hiện thời.

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

Các thuộc tính

hash	Tên anchor của vị trí hiện thời (VÍ DỤ: topic3).
Host	Phần hostname:port của URL (VÍ DỤ: www.abc.com).
Hostname	Tên của host và domain (VÍ DỤ: www.abc.com).
href	Toàn bộ URL cho document hiện tại.
Pathname	Phần đường dẫn của URL (VÍ DỤ: /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) (VÍ DỤ: http:).
Search	Câu truy vấn tìm kiếm có thể ở cuối URL cho các script CGI.

9. Đối tượng Navigator

Đối tượng này được sử dụng được các thông tin về trình duyệt như số phiên bản. Đối tượng này không có phương thức hay 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.

10. Đối tượng document

Đối tượng này chứa các thông tin về document hiện thời. Đối tượng document được tạo ra bằng cặp thẻ <BODY> và </BODY>. Một số các thuộc tính gắn với thẻ <BODY>.

Các thuộc tính

alinkColor	Giống như thuộc tính ALINK.
anchor	Mảng tất cả các anchor trong document.
bgColor	Giống thuộc tính BGCOLOR.
cookie	Sử dụng để xác định cookie.
fgColor	Giống thuộc tính TEXT.
forms	Mảng tất cả các form trong document.
lastModified	Ngày cuối cùng văn bản được sửa.
linkColor	Giống thuộc tính LINK.
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>.
vlinkColor	Giống thuộc tính VLINK.

Các phương thức

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

LỜI KẾT

Bạn có thể tham khảo toàn diện JavaScript trong quyển Teach Yourself JavaScript in 14 Days, hoặc JavaScript Guide. Do JavaScript là ngôn ngữ còn mới và có sự thay đổi nhanh chóng, bạn nên đến với trang Web của hãng Netscape (http://www.netscape.com) để có các thông tin mới nhất về ngôn ngữ này.