

MỤC LỤC

MỤC LỤC	1
CHƯƠNG 1: GIỚI THIỆU HTML – CÁC SIÊU LIÊN KẾT	4
1.1 GIỚI THIỆU	4
1.2 GIỚI THIỆU INTERNET	4
1.3 GIỚI THIỆU HTML	5
1.3.1 HTML Development	6
1.3.2 Cấu trúc của một tài liệu HTML	7
1.3.3 Sử dụng thẻ <META>	10
1.3.4 Sử dụng ký tự đặc biệt trong tài liệu HTML	10
1.4 SỬ DỤNG CÁC SIÊU LIÊN KẾT	12
1.4.1 Giới thiệu siêu liên kết và URL	12
1.4.2 Sử dụng siêu liên kết	14
1.4.3 Điều hướng quanh Web site	19
CHƯƠNG 2: CÁC THẺ HTML CƠ BẢN	22
2.1 GIỚI THIỆU	22
2.2 HEADING (TIÊU ĐỀ)	22
2.3 THẺ KHỐI , <DIV>	23
2.4 DANH SÁCH	24
2.6.1 Danh sách không thứ tự	25
2.6.2 Danh sách có thứ tự	28
2.5 THẺ KÈ ĐƯỜNG NGANG: <HR>	32
2.6 SỬ DỤNG FONT	33
2.7 SỬ DỤNG MÀU SẮC	34
2.8 SỬ DỤNG HÌNH ẢNH TRONG TÀI LIỆU HTML	35
CHƯƠNG 3: SỬ DỤNG BẢNG	40
3.1 GIỚI THIỆU	40
3.2 CÁCH TẠO BẢNG	40
3.2.1 Thẻ dùng để tạo bảng	40
3.2.2 Chèn hàng và cột	43
3.2.3 Xóa hàng và cột	44
3.2.4 Trộn ô: kết hợp cột hay dòng	45
3.2.5 Định dạng cho ô	48
3.4 CHÈN MULTIMEDIA VÀO TÀI LIỆU HTML	50
3.4.1 Chèn ảnh động (.GIF) vào tài liệu HTML	50
3.4.2 Chèn âm thanh vào tài liệu HTML	51
3.4.3 Chèn video vào tài liệu HTML	53
CHƯƠNG 4: SỬ DỤNG BIỂU MẪU VÀ KHUNG	54
4.1 GIỚI THIỆU	54
4.2 GIỚI THIỆU BIỂU MẪU	54
4.2.1 Sử dụng biểu mẫu	54
4.2.2 Phần tử FORM	55
4.2.3 Các phần tử nhập của HTML	56
4.2.4 Tạo biểu mẫu	66
4.3 KHUNG (FRAME)	69

4.3.1 Tại sao sử dụng khung?	70
4.3.2 Sử dụng khung.....	70
4.3.3 Phần tử IFRAME – Khung trên dòng (inline frame).....	76
CHƯƠNG 5: SỬ DỤNG STYLE	78
5.1 GIỚI THIỆU	78
5.2 DHTML	78
5.2.1 Giới thiệu DHTML.....	78
5.2.2 Các đặc điểm của DHTML.....	79
5.3 STYLE SHEETS.....	80
5.3.1 Khái niệm, chức năng và lợi ích của Style Sheets	80
5.3.2 Quy tắc Style Sheets	83
5.3.3 Các Selector trong Style Sheets.....	85
5.3.4 Kết hợp, liên kết và chèn một Style Sheet vào tài liệu HTML	91
5.3.5 Thiết lập thuộc tính trong Style Sheet	93
CHƯƠNG 6: JAVASCRIPT, NỀN TẢNG VÀ CÚ PHÁP	95
6.1 GIỚI THIỆU	95
6.2 GIỚI THIỆU VỀ JAVASCRIPT	95
6.2.1 Javascript là gì?.....	95
6.2.2 Hiệu ứng và quy tắc Javascript.....	96
6.2.3 Các công cụ Javascript và IDE, môi trường thực thi.....	97
6.2.4 Nhúng Javascript vào trang Web	97
6.2.5 Ví dụ đơn giản sử dụng hộp thông báo và phương thức write	103
6.3 CÁC BIẾN.....	104
6.3.1 Khai báo biến.....	104
6.3.2 Phạm vi của biến.....	104
6.4 CÁC KIỂU DỮ LIỆU	105
Ý nghĩa	108
6.5 CÁC TOÁN TỬ	109
6.5.1 Toán tử số học.....	110
6.5.2 Toán tử so sánh	111
6.5.3 Toán tử logic	112
6.5.4 Toán tử chuỗi	113
6.5.4 Toán tử Evaluation.....	113
6.5.5 Mức ưu tiên của các toán tử.....	115
6.6 MẢNG	115
6.7 CÁC CÂU LỆNH ĐIỀU KIỆN	120
6.8 CÁC LỆNH VÒNG LẶP.....	121
6.9 HÀM (FUNCTION).....	125
CHƯƠNG 7: CÁC ĐỐI TƯỢNG CƠ BẢN TRONG JAVASCRIPT	129
7.1 GIỚI THIỆU	129
7.2 CÁC ĐỐI TƯỢNG JAVASCRIPT	129
7.2.1 Câu lệnh This	130
7.2.2 Đối tượng String.....	132
7.2.3 Đối tượng Math.....	134
7.2.4 Đối tượng Date.....	137

7.3 ĐỐI TƯỢNG EVENT – KHÁI NIỆM	141
7.4 CÁC SỰ KIỆN JAVASCRIPT	141
7.5 TRÌNH XỬ LÝ SỰ KIỆN.....	150
TÀI LIỆU THAM KHẢO.....	153

CHƯƠNG 1: GIỚI THIỆU HTML – CÁC SIÊU LIÊN KẾT

Kết thúc chương này, bạn có thể:

- Mô tả về Internet
- Mô tả về HTML
- Viết một tài liệu HTML đơn giản
- Sử dụng siêu liên kết trong tài liệu HTML
- Sử dụng thẻ <META>
- Sử dụng các ký tự đặc biệt trong tài liệu HTML

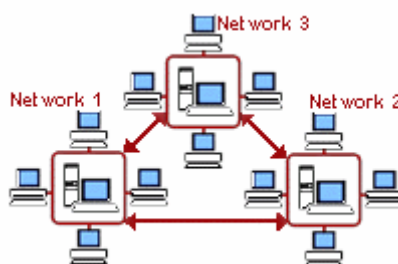
1.1 GIỚI THIỆU

“Internet”, “World Wide Web”, và “Web page” không chỉ còn là các thuật ngữ. Giờ đây, các thuật ngữ này đã trở thành hiện thực. Internet là mạng máy tính lớn nhất trên thế giới, được xem như là mạng của các mạng. World Wide Web là một tập con của Internet. World Wide Web gồm các Web Servers có mặt khắp mọi nơi trên thế giới. Các Web server chứa thông tin mà bất kỳ người dùng nào trên thế giới cũng có thể truy cập được. Các thông tin đó được lưu trữ dưới dạng các trang Web.

Trong phần này, chúng ta sẽ học về Ngôn ngữ đánh dấu siêu văn bản (HTML), đây là một phần quan trọng trong lĩnh vực thiết kế và phát triển thế giới Web.

1.2 GIỚI THIỆU INTERNET

Mạng là một nhóm các máy tính kết nối với nhau. Internet là mạng của các mạng. Giao thức TCP/IP (Transmission Control Protocol/Internet Protocol) cung cấp việc kết nối tất cả các máy tính trên thế giới.



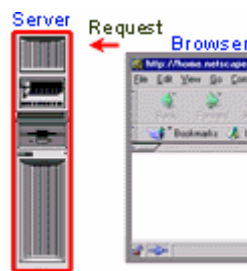
Hình 1.1: Internet

World Wide Web là một tập con của Internet. Nó bắt đầu như là đề án nghiên cứu cấp quốc gia tại phòng nghiên cứu CERN ở Thụy Sĩ. Ngày nay, nó cung cấp thông tin cho người dùng trên toàn thế giới.

WWW hoạt động dựa trên 3 cơ chế để đưa các tài nguyên có giá trị đến với người dùng. Đó là:

- Giao thức: Người dùng tuân theo các giao thức này để truy cập tài nguyên trên Web. HyperText Transfer Protocol(HTTP) là giao thức được WWW sử dụng.
- Địa chỉ: WWW tuân theo một cách thức đặt tên thống nhất để truy cập vào các tài nguyên trên Web. URL được sử dụng để nhận dạng các trang và các tài nguyên trên Web.
- HTML: Ngôn ngữ đánh dấu siêu văn bản (HTML) được sử dụng để tạo các tài liệu có thể truy cập trên Web. Tài liệu HTML được tạo ra bằng cách sử dụng các thẻ và các phần tử của HTML. File được lưu trên Web server với đuôi .htm hoặc .html.

Khi bạn sử dụng trình duyệt để yêu cầu một số thông tin nào đó, Web server sẽ đáp ứng các yêu cầu đó. Nó gửi thông tin được yêu cầu đến trình duyệt dưới dạng các trang web. Trình duyệt định dạng thông tin do máy chủ gửi về và hiển thị chúng.



Hình 1.2: Trình duyệt yêu cầu đến máy chủ

1.3 GIỚI THIỆU HTML

Ngôn ngữ đánh dấu siêu văn bản chỉ rõ một trang Web được hiển thị như thế nào trong một trình duyệt. Sử dụng các thẻ và các phần tử HTML, bạn có thể:

- Điều khiển hình thức và nội dung của trang
- Xuất bản các tài liệu trực tuyến và truy xuất thông tin trực tuyến bằng cách sử dụng các liên kết được chèn vào tài liệu HTML
- Tạo các biểu mẫu trực tuyến để thu thập thông tin về người dùng, quản lý các giao dịch
- Chèn các đối tượng như audio clip, video clip, các thành phần ActiveX và các Java Applet vào tài liệu HTML

Tài liệu HTML tạo thành mã nguồn của trang Web. Khi được xem trên trình soạn thảo, tài liệu này là một chuỗi các thẻ và các phần tử, mà chúng xác định trang web hiển thị như thế nào. Trình duyệt đọc các file có đuôi .htm hay .html và hiển thị trang web đó theo các lệnh có trong đó.

Ví dụ, theo cú pháp HTML dưới đây sẽ hiển thị thông điệp “My first HTML document”

Ví dụ 1:

```
<HTML>
```

```
    <HEAD>
```

```
<TITLE>Welcome to HTML</TITLE>

</HEAD>

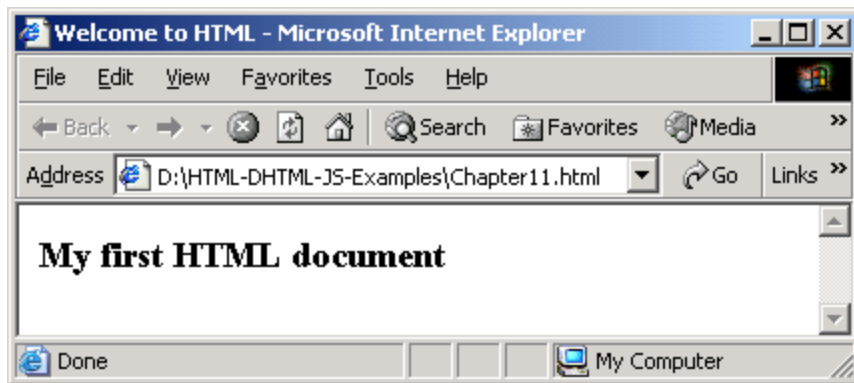
<BODY>

    <H3>My first HTML document</H3>

</BODY>

</HTML>
```

Trình duyệt thông dịch những lệnh này và hiển thị trang web như hình 1.3



Hình 1.3: Kết quả ví dụ 1

1.3.1 HTML Development

Tài liệu HTML được hiển thị trên trình duyệt. Vậy trình duyệt là gì ? Trình duyệt là một ứng dụng được cài đặt trên máy khách. Trình duyệt đọc mã nguồn HTML và hiển thị trang theo các lệnh trong đó.

Trình duyệt được sử dụng để xem các trang Web và điều hướng. Trình duyệt được biết đến sớm nhất là Mosaic, được phát triển bởi Trung tâm ứng dụng siêu máy tính quốc gia (NCSA). Ngày nay, có nhiều trình duyệt được sử dụng trên Internet. Netscape's Navigator và Microsoft's Internet Explorer là hai trình duyệt được sử dụng phổ biến. Đối với người dùng, trình duyệt dễ sử dụng bởi vì nó có giao diện đồ họa với việc trỏ và kích chuột.

Để tạo một tài liệu nguồn, bạn phải cần một trình soạn thảo HTML. Ngày nay, có nhiều trình soạn thảo đang được sử dụng: Microsoft FrontPage là một công cụ tổng hợp được dùng để tạo, thiết kế và hiệu chỉnh các trang Web. Chúng ta cũng có thể thêm văn bản, hình ảnh, bảng và những thành phần HTML khác vào trang. Thêm vào đó, một biểu mẫu cũng có thể được tạo ra bằng FrontPage. Một khi chúng ta tạo ra giao diện cho trang web, FrontPage tự động tạo mã HTML cần thiết. Chúng ta cũng có thể dùng Notepad để tạo tài liệu HTML. Để xem được tài liệu trên trình duyệt bạn phải lưu nó với đuôi là .htm hay .html.

Các lệnh HTML được gọi là các thẻ. Các thẻ này được dùng để điều khiển nội dung và hình thức trình bày của tài liệu HTML. Thẻ mở (" \langle ") và thẻ đóng (" \rangle "), chỉ ra sự bắt đầu và kết thúc của một lệnh HTML.

Ví dụ, thẻ HTML được sử dụng để đánh dấu sự bắt đầu và kết thúc của tài liệu HTML

```
<HTML>
. . .
```

</HTML>

Chú ý rằng các thẻ không phân biệt chữ hoa và chữ thường, vì thế bạn có thể sử dụng <html> thay cho <HTML>

Thẻ HTML bao gồm:

<ELEMENT ATTRIBUTE = value>

Phần tử: nhận dạng thẻ

Thuộc tính: Mô tả thẻ

Value: giá trị được thiết lập cho thuộc tính

Ví dụ, <BODY BGCOLOR = lavender>

Trong ví dụ trên, BODY là phần tử, BGCOLOR(nền) là thuộc tính và “lavender” là giá trị. Khi cú pháp HTML được thực hiện, màu nền cho cả trang được thiết lập là màu “lavender”.

1.3.2 Cấu trúc của một tài liệu HTML

Một tài liệu HTML gồm 3 phần cơ bản:

- Phần HTML: Mọi tài liệu HTML phải bắt đầu bằng thẻ mở HTML <HTML> và kết thúc bằng thẻ đóng HTML </HTML>

<HTML> ... </HTML>

Thẻ HTML báo cho trình duyệt biết nội dung giữa hai thẻ này là một tài liệu HTML

- Phần tiêu đề: Phần tiêu đề bắt đầu bằng thẻ <HEAD> và kết thúc bởi thẻ </HEAD>. Phần này chứa tiêu đề mà được hiển thị trên thanh điều hướng của trang Web. Tiêu đề nằm trong thẻ TITLE, bắt đầu bằng thẻ <TITLE> và kết thúc là thẻ </TITLE>.

Tiêu đề là phần khá quan trọng. Các mốc được dùng để đánh dấu một web site. Trình duyệt sử dụng tiêu đề để lưu trữ các mốc này. Do đó, khi người dùng tìm kiếm thông tin, tiêu đề của trang Web cung cấp từ khóa chính yếu cho việc tìm kiếm.

- Phần thân: phần này nằm sau phần tiêu đề. Phần thân bao gồm văn bản, hình ảnh và các liên kết mà bạn muốn hiển thị trên trang web của mình. Phần thân bắt đầu bằng thẻ <BODY> và kết thúc bằng thẻ </BODY>

Ví dụ 2:

<HTML>

<HEAD>

<TITLE>Welcome to the world of HTML</TITLE>

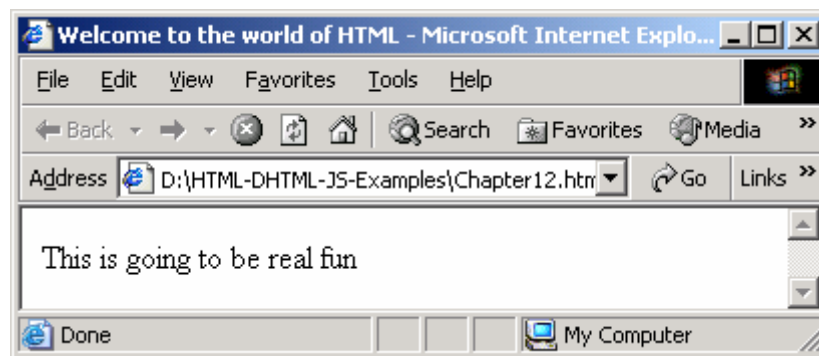
</HEAD>

<BODY>

<P>This is going to be real fun</P>

</BODY>

</HTML>



Hình 1.4: Kết quả của ví dụ 2

Đoạn

Bạn có chú ý đến thẻ <P> trong ví dụ 2 không? Thẻ <P> để trình bày một đoạn

Khi viết một bài báo hay một bài luận, bạn nhóm nội dung thành một loạt các đoạn. Mục đích là nhóm các ý tưởng logic lại với nhau và áp dụng một số định dạng cho nội dung. Trong một tài liệu HTML, nội dung có thể được nhóm thành các đoạn. Thẻ đoạn <P> được sử dụng để đánh dấu sự bắt đầu của một đoạn mới.

Ví dụ 3

<HTML>

<HEAD>

<TITLE>Welcome to HTML</TITLE>

</HEAD>

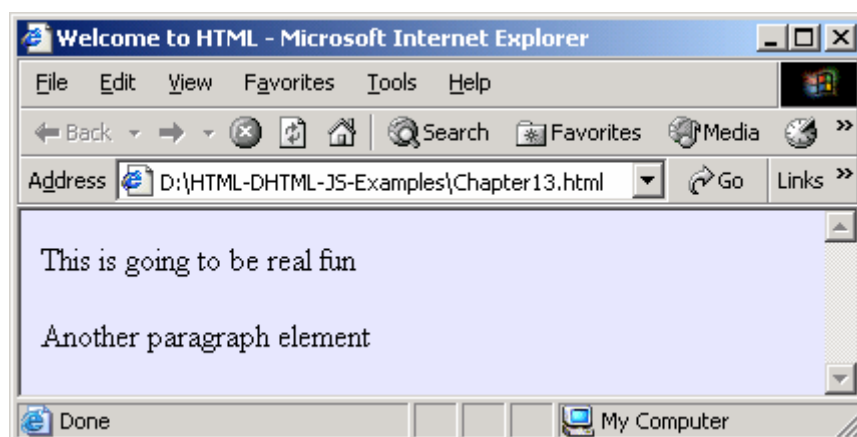
<BODY BGCOLOR = lavender>

<P>This is going to be real fun

<P> Another paragraph element

</BODY>

</HTML>



Hình 1.5: Kết quả của ví dụ 3

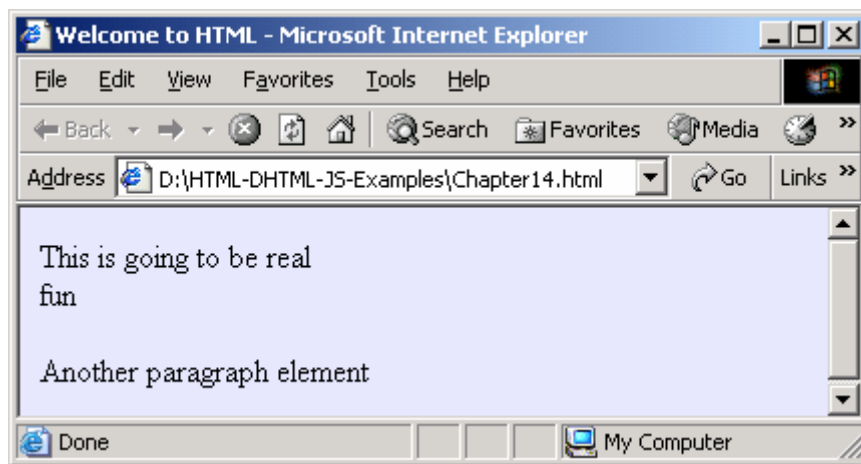
Thẻ đóng </P> là không bắt buộc. Thẻ <P> kế tiếp sẽ tự động bắt đầu một đoạn mới.

Các thẻ ngắt

Phần tử
 được sử dụng để ngắt dòng trong tài liệu HTML. Thẻ
 bổ sung một ký tự xuống dòng tại vị trí của thẻ.

Ví dụ 4:

```
<HTML>
  <HEAD>
    <TITLE>Welcome to HTML</TITLE>
  </HEAD>
  <BODY BGCOLOR = lavender>
    <P>This is going to be real <BR>fun
    <P> Another paragraph element
  </BODY>
</HTML>
```



Hình 1.6: Kết quả của ví dụ 4

Chọn canh lề

Thuộc tính align được sử dụng để canh lề cho các phần tử HTML trong trang Web. Chúng ta có thể canh lề văn bản, các đối tượng, hình ảnh, các đoạn, các phân đoạn,... Sau đây, bạn sẽ học cách canh lề văn bản:

Thuộc tính align gồm các giá trị sau:

Value	Description
Left	Văn bản được canh lề trái
Center	Văn bản được canh giữa
Right	Văn bản được canh phải
Justify	Văn bản được canh đều hai bên

Canh lề được mặc định dựa vào hướng của văn bản. Nếu hướng văn bản là từ trái sang phải thì mặc định là trái.

Ví dụ 5

```
<HTML>

  <HEAD>

    <TITLE>Learning HTML</TITLE>

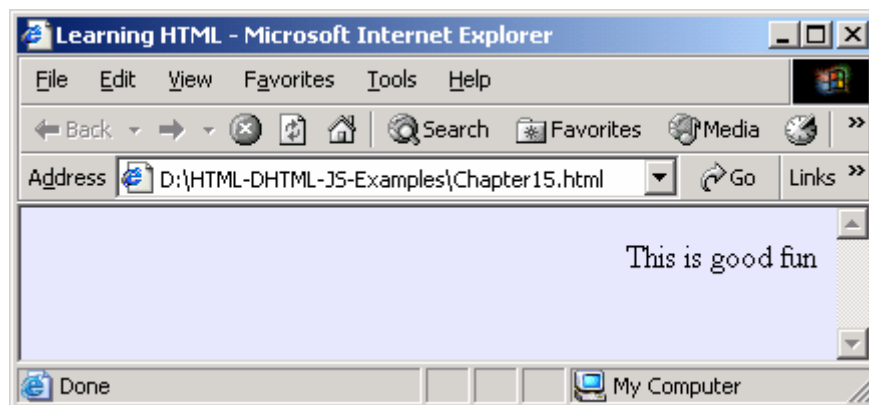
  </HEAD>

  <BODY BGCOLOR=lavender>

    <P align = right>This is good fun</P>

  </BODY>

</HTML>
```



Hình 1.7: Kết quả của ví dụ 5

1.3.3 Sử dụng thẻ <META>

Phần tiêu đề cũng chứa phần tử META. Phần tử này cung cấp thông tin về trang web của bạn. Nó gồm tên tác giả, tên phần mềm dùng để viết trang đó, tên công ty, thông tin liên lạc Phần tử META sử dụng sự kết hợp giữa thuộc tính và giá trị.

Ví dụ, để chỉ Graham Browne là tác giả, người ta sử dụng phần tử META như sau:

```
<META name="Author" content="Graham Browne">
```

Tác giả của tài liệu là "Graham Browne"

Thuộc tính http-equiv có thể được sử dụng để thay thế thuộc tính name. Máy chủ HTTP sử dụng thuộc tính này để tạo ra một đầu đáp ứng HTTP (HTTP response header).

Đầu đáp ứng được truyền đến trình duyệt để nhận dạng dữ liệu. Nếu trình duyệt hiểu được đầu đáp ứng này, nó sẽ tiến hành các hành động đặc biệt đối với đầu đáp ứng đó.

Ví dụ, <META http-equiv="Expires" content="Mon, 15 Sep 2003 14:25:27 GMT"> sẽ sinh ra một đầu đáp ứng HTTP như sau:

```
Expires: Mon, 15 Sep 2003 14:25:27 GMT
```

Do vậy, nếu tài liệu đã lưu lại, HTTP sẽ biết khi nào truy xuất một bản sao của tài liệu tương ứng.

1.3.4 Sử dụng ký tự đặc biệt trong tài liệu HTML

Bạn có thể chèn các ký tự đặc biệt vào văn bản của tài liệu HTML. Để đảm bảo trình duyệt không nhầm chúng với thẻ HTML, bạn phải gán mã định dạng cho các ký tự đặc biệt này.

Ký tự đặc biệt	Mã định dạng	Ví dụ
Lớn hơn (>)	>	<pre><HTML> <HEAD> <TITLE>Learning HTML</TITLE> </HEAD> <BODY> <CODE>If A &gt; B Then
 A = A + 1 </CODE> <P> The above statement used special characters </BODY> </HTML></pre>
Nhỏ hơn (<)	<	<pre><HTML> <HEAD> <TITLE>Learning HTML</TITLE> </HEAD> <BODY> <CODE>If A &lt; B Then
 A = A + 1 </CODE> <P> The above statement used special characters </BODY> </HTML></pre>
Trích dẫn(“”)	"	<pre><HTML> <HEAD> <TITLE>Learning HTML</TITLE> </HEAD> <BODY> &quot; To be or not to be ? &quot; That is the question </BODY> </HTML></pre>
Ký tự “&”	&	<pre><HTML> <HEAD></pre>

		<pre> <TITLE>Learning HTML</TITLE> </HEAD> <BODY> <P> William & Graham went to the fair </BODY> </HTML> </pre>
--	--	---

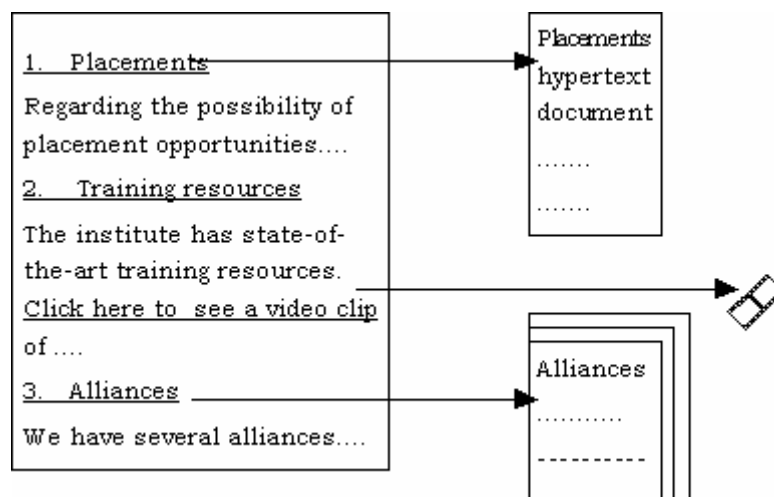
1.4 SỬ DỤNG CÁC SIÊU LIÊN KẾT

Siêu liên kết là một phần tử bên trong tài liệu mà liên kết đến một vị trí khác trong cùng tài liệu đó hoặc đến một tài liệu hoàn toàn khác. Chẳng hạn, khi ta kích vào siêu liên kết sẽ nhảy đến liên kết cần đến. Các siêu liên kết là thành phần quan trọng nhất của hệ thống siêu văn bản

1.4.1 Giới thiệu siêu liên kết và URL

Khả năng chính của HTML là hỗ trợ siêu liên kết. Một siêu liên kết, hay nói ngắn gọn là một liên kết, là sự kết nối đến tài liệu hay file khác (đồ họa, âm thanh, video) hoặc ngay cả đến một phần khác trong cùng tài liệu đó. Khi kích vào siêu liên kết, người dùng được đưa đến địa chỉ URL mà chúng ta chỉ rõ trong liên kết.

- ☐ Một phần khác trong cùng tài liệu
- ☐ Một tài liệu khác
- ☐ Một phần trong tài liệu khác
- ☐ Các file khác – hình ảnh, âm thanh, trích đoạn video
- ☐ Vị trí hoặc máy chủ khác

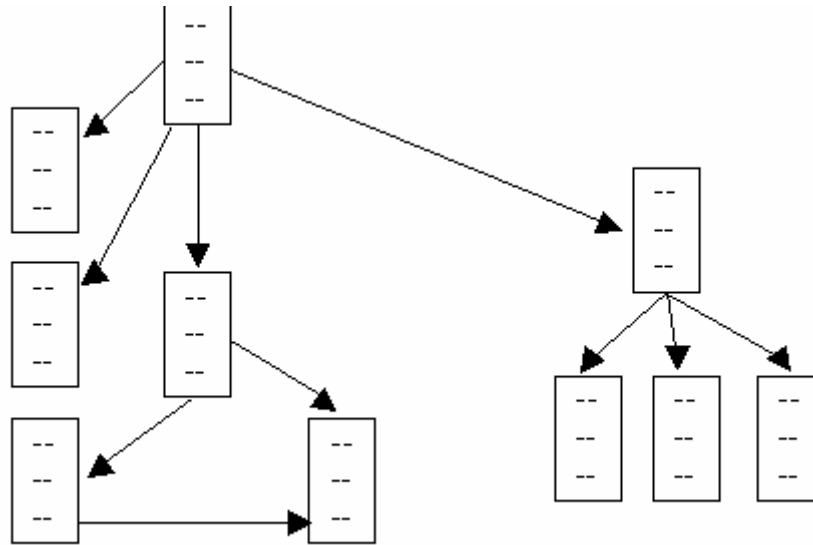


Hình 1.8: Sử dụng liên kết

Các liên kết có thể là liên kết trong hoặc liên kết ngoài

- ☐ **Liên kết trong** là liên kết kết nối đến các phần trong cùng tài liệu hoặc cùng một web site

- **Liên kết ngoài** là liên kết kết nối đến các trang trên các web site khác hoặc máy chủ khác.



Hình 1.9: Liên kết trong và liên kết ngoài

Để tạo siêu liên kết, chúng ta cần phải xác định hai thành phần:

1. Địa chỉ đầy đủ hoặc URL của file được kết nối
2. Điểm nóng cung cấp cho liên kết. Điểm nóng này có thể là một dòng văn bản, thậm chí là một ảnh.

Khi người dùng kích vào điểm nóng, trình duyệt đọc địa chỉ được chỉ ra trong URL và “nhảy” đến vị trí mới

Mỗi nguồn tài nguyên trên Web có một địa chỉ duy nhất. Ví dụ, 207.46.130.149 là địa chỉ web site của Micorsoft. Giờ đây, để nhớ các con số này rất khó và dễ nhầm lẫn. Vì vậy, người ta sử dụng các URL. URL là một chuỗi cung cấp địa chỉ Internet của web site hay tài nguyên trên World Wide Web.

Định dạng đặc trưng là www.nameofsite.typeofsite.contrycode

[trong đó

Nameofsite: tên của site

Typeofsite: kiểu của site

Contrycode: mã nước]

Ví dụ, 216.239.33.101 có thể được biểu diễn bằng URL là www.google.com

URL cũng nhận biết được giao thức mà site hay tài nguyên được truy nhập. Dạng URL thông thường nhất là “http”, nó cung cấp địa chỉ Internet của một trang Web. Một vài dạng URL khác là “gopher”, nó cung cấp địa chỉ Internet của một thư mục Gopher, và “ftp”, cung cấp vị trí của một tài nguyên FTP trên mạng.

URL cũng có thể tham chiếu đến một vị trí trong một tài nguyên. Ví dụ, bạn có thể tạo liên kết đến một chủ đề trong cùng một tài liệu. Trong trường hợp đó, định danh đoạn được sử dụng ở phần cuối của URL

Định dạng là, giao thức: tên của site / tài liệu chính #định danh đoạn

Có hai dạng URL:

- ☐ **URL tuyệt đối** – là địa chỉ Internet đầy đủ của trang hoặc file, bao gồm giao thức, vị trí mạng, đường dẫn tùy chọn và tên file. Ví dụ, <http://www.microsoft.com> là một địa chỉ URL tuyệt đối.
- ☐ **URL tương đối** – là một URL có một hoặc nhiều phần bị thiếu. Trình duyệt lấy thông tin bị thiếu từ trang chứa URL đó. Ví dụ, nếu giao thức bị thiếu, trình duyệt sử dụng giao thức của trang hiện thời.

1.4.2 Sử dụng siêu liên kết

Thẻ <A> được sử dụng để xác định văn bản hay ảnh nào sẽ dùng làm siêu liên kết trong tài liệu HTML. Thuộc tính HREF (tham chiếu siêu văn bản) được dùng để chỉ địa chỉ hay URL của tài liệu hoặc file được liên kết.

Cú pháp của HREF là:

```
<A HREF = protocol://host.domain:port/path/filename>Hypertext</A>
```

Trong đó,

Giao thức – Đây là loại giao thức. Một số giao thức thường dùng:

- ☐ http – giao thức truyền siêu văn bản
- ☐ telnet – mở một phiên telnet
- ☐ gopher – tìm kiếm file
- ☐ ftp – giao thức truyền file
- ☐ mailto – gửi thư điện tử

Host.domain – Đây là địa chỉ Internet của máy chủ

Port - Cổng phục vụ của máy chủ đích

HyperText – Đây là văn bản hay hình ảnh mà người dùng cần nhấp vào để kích hoạt liên kết

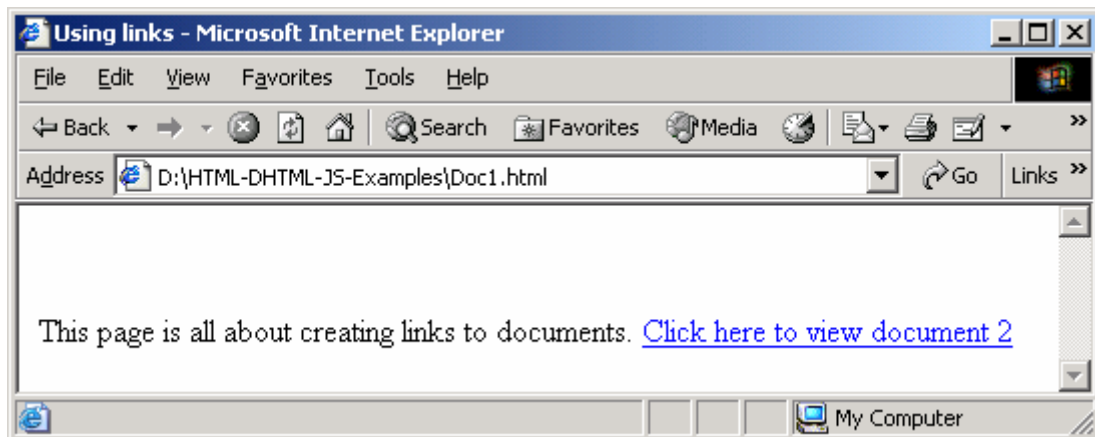
Liên kết đến những tài liệu khác

Giả sử có hai tài liệu HTML trên đĩa cứng cục bộ, Doc1.htm và Doc2.htm. Để tạo một liên kết từ Doc1.html đến Doc2.htm

Ví dụ 6:

```
<HTML>
  <HEAD>
    <TITLE> Using links</TITLE>
  </HEAD>
```

```
<BODY>
    <BR><BR>
    <P> This page is all about creating links to documents.
    <A HREF = Doc2.html>Click here to view document 2</A>
</BODY>
</HTML>
```

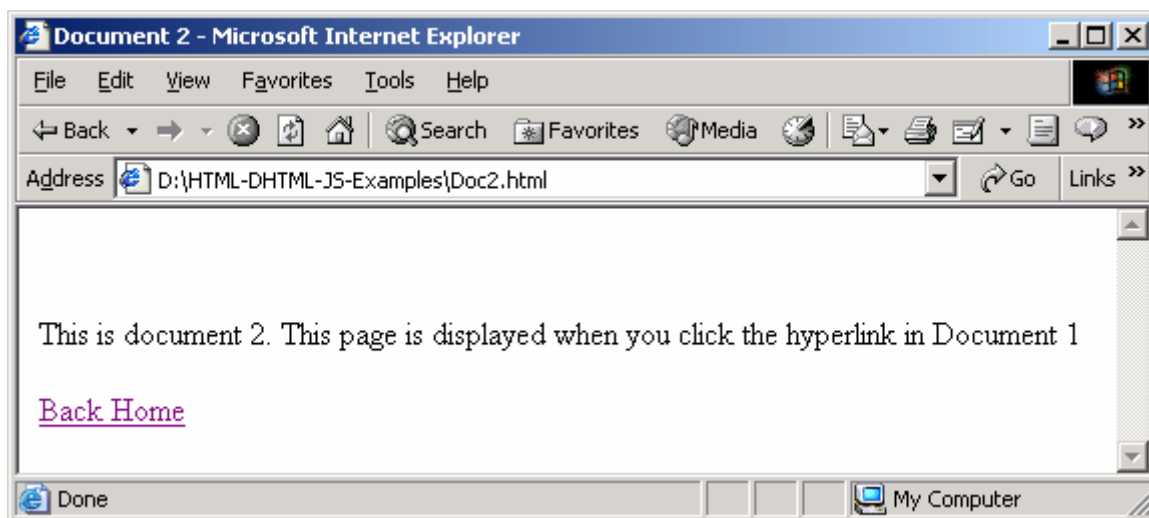


Hình 1.10: Kết quả của ví dụ 6

Khi người dùng “nhảy” đến một tài liệu khác, bạn nên cung cấp cách để quay trở lại trang chủ hoặc định hướng đến một file khác.

Ví dụ 7:

```
<HTML>
    <HEAD>
        <TITLE> Document 2</TITLE>
    </HEAD>
    <BODY>
        <BR><BR>
        <P> This is document 2. This page is displayed when you click
        the hyperlink in Document 1
        <BR><BR>
        <A HREF= Doc1.html>Back Home</A>
    </BODY>
</HTML>
```



Hình 1.11: Kết quả ví dụ 7

Chú ý là các liên kết được gạch chân. Trình duyệt tự động gạch chân các liên kết. Nó cũng thay đổi hình dáng con trỏ chuột khi người sử dụng di chuyển chuột vào liên kết.

Ở ví dụ trên, các file nằm trong cùng một thư mục, vì vậy chỉ cần chỉ ra tên file trong thông số HREF là đủ. Tuy nhiên, để liên kết đến các file ở thư mục khác, cần phải cung cấp đường dẫn tuyệt đối hay đường dẫn tương đối

Đường dẫn tuyệt đối chỉ ra đường dẫn đầy đủ từ thư mục gốc đến file. Ví dụ, C:\mydirectory\html examples\ Doc2.htm

Đường dẫn tương đối chỉ ra vị trí liên quan của file với vị trí file hiện tại. Ví dụ, nếu thư mục hiện hành là mydirectory thì đường dẫn sẽ là ,

```
<A HREF= "../Doc3.htm">Next page</A>
```

Vì vậy, nếu muốn liên kết các tài liệu không liên quan với nhau thì ta nên dùng đường dẫn tuyệt đối. Tuy nhiên, nếu ta có một nhóm tài liệu có liên quan với nhau, chẳng hạn phần trợ giúp trong HTML, thì ta nên sử dụng đường dẫn tương đối cho các tài liệu trong nhóm và đường dẫn tuyệt đối cho các tài liệu không liên quan trực tiếp đến chủ đề. Khi đó người dùng có thể cài đặt phần trợ giúp này trong thư mục mình chọn và nó vẫn hoạt động.

Liên kết đến các phần trong cùng một tài liệu

Thẻ neo <A> (anchor) được sử dụng để người dùng có thể “nhảy” đến những phần khác nhau của một tài liệu. Ví dụ, bạn có thể hiển thị nội dung của trang Web như một loạt các liên kết. Khi người dùng kích vào một đề tài nào đó thì các chi tiết nằm ở một phần khác của tài liệu được hiển thị.

Kiểu liên kết này gọi là “**named anchor**” bởi vì thuộc tính NAME được sử dụng để tạo các liên kết này

```
<A NAME = "marker">Topic name</A>
```

Bạn không phải sử dụng bất kỳ văn bản nào để đánh dấu điểm neo

Để dùng, ta sử dụng vạch dấu (marker) trong thông số HREF như sau

```
<A HREF= "#marker">Topic name</A>
```


Dấu # ở trước tên của siêu liên kết để báo cho trình duyệt biết rằng liên kết này liên kết đến một điểm được đặt tên trong tài liệu. Khi không có tài liệu nào được chỉ ra trước ký tự #, trình duyệt hiểu rằng liên kết này nằm trong cùng tài liệu.

Ví dụ 8:

```
<HTML>

  <HEAD>

    <TITLE>Using Links</TITLE>

  </HEAD>

  <BODY>

    <A HREF = #Internet>Internet</A>

    <BR><BR>

    <A HREF = #HTML>Introduction to HTML</A>

    <BR><BR>

    <A HREF = #Consistency>Unity and Variety</A>

    <BR><BR>

    <A name = Internet>Internet</A>

    <BR>

    <P> The Internet is a network of networks. That is, computer
    networks are linked to other networks, spanning countries and today
    the globe. The TCP/IP transfer protocol provides the bindings that
    connect all these computers the world over.

    </P>

    <A name = HTML>Introduction to HTML</A>

    <BR>

    <P> Hyper Text Markup Language is the standard language that
    the Web uses for creating and recognizing documents. Although not a
    subset of, it is loosely related to the Standard Generalized Markup
    Language (SGML). SGML is a method for representing document
    formatting languages.

    </P>

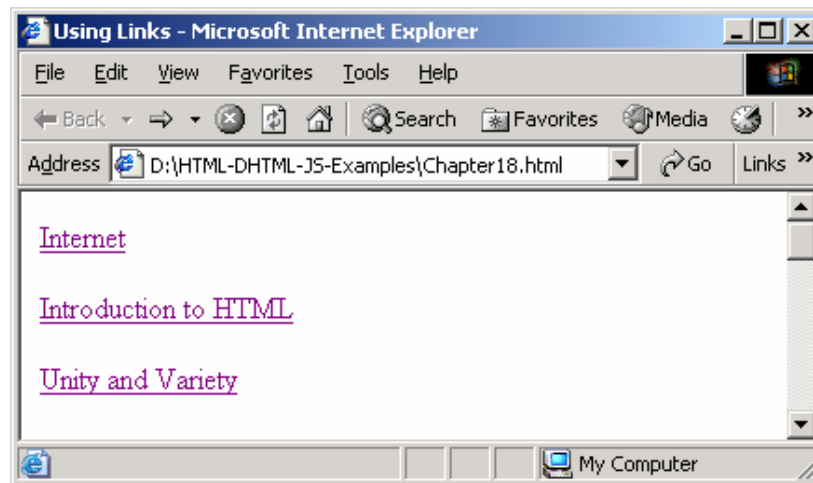
    <A name = Consistency>Unity and Variety</A>

    <BR>

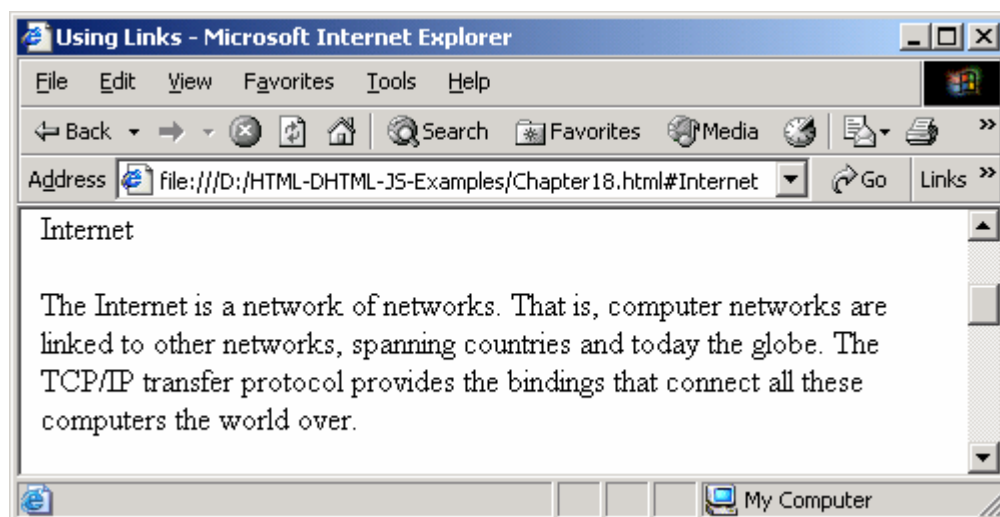
    <P> A basic rule is that of unity and variety. That is,
    everything should fit together as a unit, but at the same time there
    is enough variety to keep things interesting. Consistency creates and
    reinforces the unique identity of a site. Colors, fonts, column
    layout and other design elements should be consistent throughout
    every section of the site.

    </BODY>

</HTML>
```



Hình 1.12: Kết quả ví dụ 8



Hình 1.13: Kết quả ví dụ 8 sau khi kích vào Internet

Liên kết đến một điểm xác định ở một tài liệu khác

Bây giờ chúng ta đã biết cách sử dụng các vạch dấu trong cùng một tài liệu, hãy thử “nhảy” đến một vị trí trên một tài liệu khác.

Để “nhảy” đến một điểm trên tài liệu khác, chúng ta cần phải chỉ ra tên của tài liệu khi chúng ta tạo vạch dấu. Trước tiên trình duyệt sẽ đọc tên tài liệu và mở tài liệu đó. Sau đó nó sẽ đọc vạch dấu và di chuyển đến điểm được đánh dấu.

Ví dụ 9

```
<HTML>
```

```
<HEAD>
```

```
<TITLE>Main document</TITLE>
```

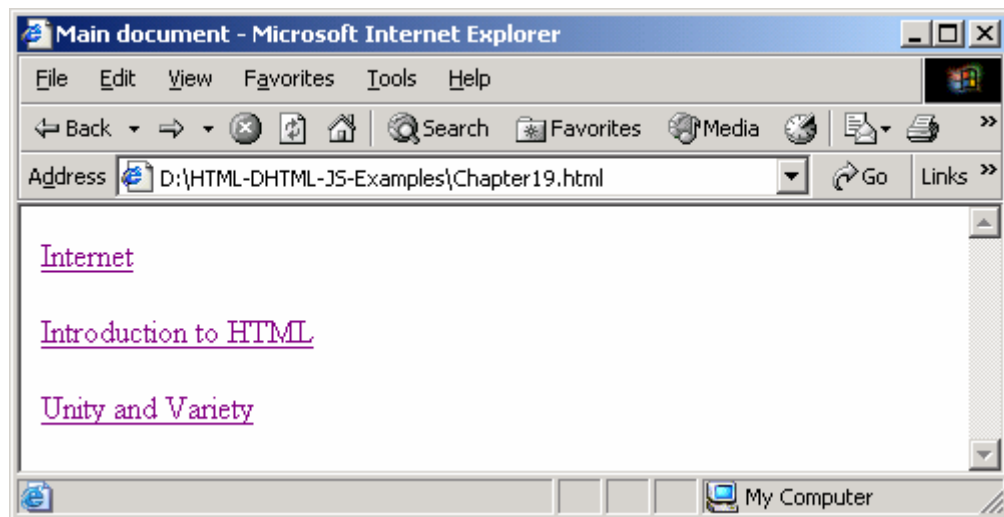
```
</HEAD>
```

```
<BODY>
```

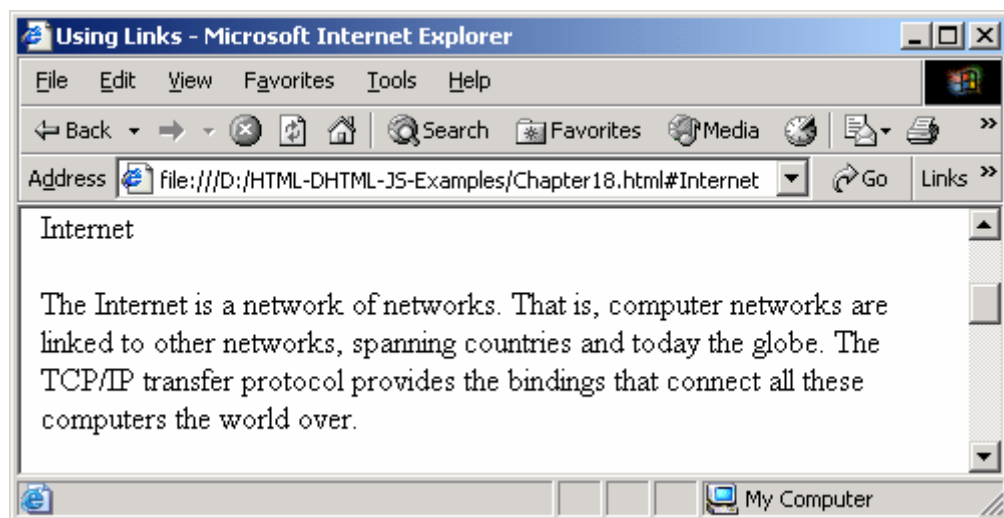
```
<A HREF=Chapter18.html#Internet>Internet</A>
```

```
<BR>
```

```
<BR>
<A HREF= Chapter18.html#HTML>Introduction to HTML</A>
<BR>
<BR>
<A HREF=Chapter18.html#Consistency>Unity and Variety</A>
</BODY>
</HTML>
```



Hình 1.14: Kết quả ví dụ 9



Hình 1.15: Kết quả ví dụ 9 sau khi kích vào Internet

Chú ý sự giống nhau giữa hình 1.13 và hình 1.15

Sử dụng e-mail

Nếu muốn người sử dụng gửi được e-mail, chúng ta có thể đưa một đặc tính vào trong trang Web và cho phép họ gửi e-mail từ trình duyệt. Tất cả những gì chúng ta cần làm là chèn giá trị *mailto* vào trong thẻ liên kết.

```
<A HREF="mailto:thisperson@mymail.com">
```

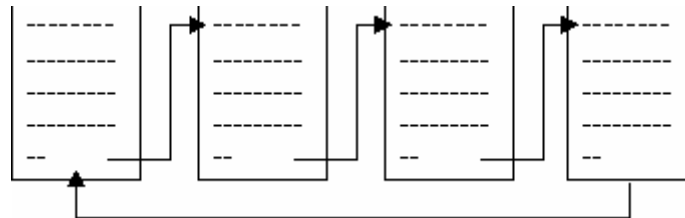
1.4.3 Điều hướng quanh Web site

Dù web site có lôi cuốn đến đâu, nếu nó không có một lược đồ điều hướng đơn giản thì người dùng cũng sẽ mất phương hướng ngay. Một số trong số họ có thể sẽ không bao giờ quay lại. Lược đồ điều hướng trong mỗi site là khác nhau. Tuy nhiên có một số nguyên tắc cơ bản mà bạn cần nhớ:

- ☐ Xác định nội dung của web site
- ☐ Tạo một lược đồ điều hướng để giúp người dùng đi đến phần cần đến một cách nhanh chóng
- ☐ Cung cấp các chức năng tìm kiếm thông tin

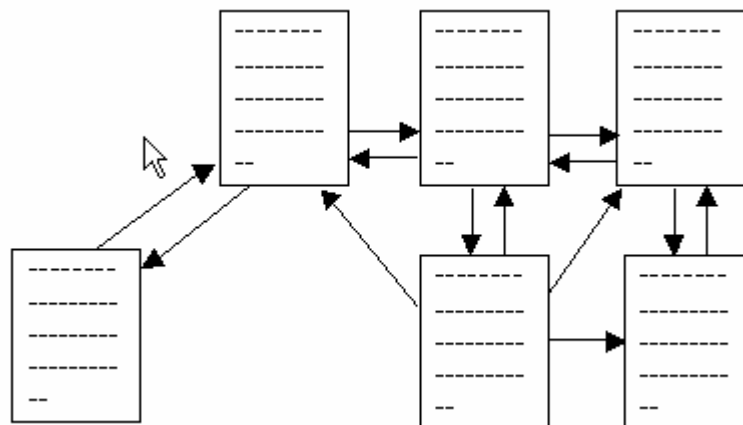
Có nhiều cách để tổ chức một web site

- ☐ **Trình bày tuyến tính** – Cách này theo cấu trúc tuyến tính. Có một chuỗi liên kết liên tục giữa các trang. Mỗi trang có liên kết đến trang trước và trang sau. Trang cuối có liên kết đến trang đầu. Định dạng này được dùng khi chúng ta muốn trình bày thông tin liên tục. Ví dụ, các chương trong cuốn sách hoặc các slide của một bài trình bày mà yêu cầu người dùng đọc thông tin theo trình tự liên tiếp nhau.



Hình 1.16: Trình bày tuyến tính

- ☐ **Trình bày theo phân cấp**: Đây là cách trình bày thông thường nhất được sử dụng trong thiết kế Web. Trang chủ liên kết với nhiều trang khác. Người dùng có thể chọn một liên kết và “nhảy” đến trang cần đến. Mỗi trang đều có liên kết về trang chủ.



Hình 1.17: Trình bày theo phân cấp

- ☐ **Bản đồ ảnh** – Một số người thấy cách trình bày trực quan thì dễ hiểu hơn. Vì vậy, bản đồ ảnh hay bản đồ site được đưa vào các trang chủ. Khi người dùng kích vào các điểm nóng thì tài liệu liên quan được hiển thị.

Khi đã quyết định một lược đồ điều hướng, đây là lúc để thiết kế tài liệu. Có một số nguyên tắc mà bạn cần ghi nhớ:

- ☐ **Siêu liên kết** nên rõ ràng. Từ nên được gạch chân và có màu xanh, bởi vì trên Web, màu xanh được quy ước là siêu liên kết. Siêu liên kết thường xuất hiện ở cuối trang. Nếu chúng xuất hiện ở giữa đoạn văn bản, có thể làm người đọc không tập trung vào đề tài.
- ☐ **Các trang con** – nên có một thanh menu hoặc thanh điều hướng, nếu không người đọc có thể không thấy các trang chính. Nếu người dùng kích vào một biểu tượng bất kỳ trên thanh điều hướng thì họ sẽ nhảy từ trang hiện thời đến trang mới
- ☐ Luôn có một **bảng mục lục** để người dùng nhảy sang một đề tài nào đó. Bảng mục lục (TOC) được hiển thị ở cuối trang. Người dùng có thể chọn một đề tài nào đó để nhảy đến trang cần đến
- ☐ Luôn có nút **“Back”** và **“Forward”** trong trang để người dùng có thể điều hướng đến trang cần đến. Trình duyệt có cung cấp các chọn lựa này, nhưng nó sử dụng các trang trong danh sách history mà có thể không phải là các trang trong Web site.

CHƯƠNG 2: CÁC THẺ HTML CƠ BẢN

Kết thúc chương này, bạn có thể:

- Sử dụng các thẻ cơ bản
- Chèn hình ảnh vào trong tài liệu HTML

2.1 GIỚI THIỆU

Chương này thảo luận về những thẻ cơ bản của HTML như các thẻ tiêu đề (Header), các thẻ đoạn và các thẻ khối. Phần này cũng nói về danh sách (Lists) và làm thế nào để sử dụng các ảnh trong tài liệu HTML

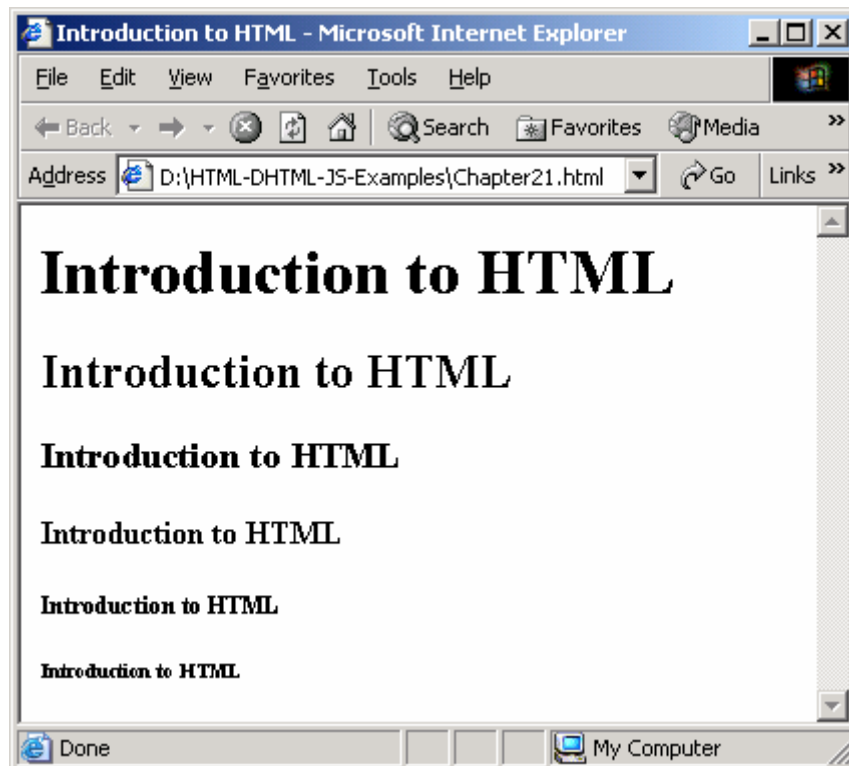
2.2 HEADING (TIÊU ĐỀ)

Phần tử này được sử dụng để cung cấp tiêu đề cho phần nội dung hiển thị trên trang web. Tất cả những phần tử tiêu đề phải có thẻ kết thúc. Nó bắt đầu bằng thẻ <H1> và kết thúc bằng </H1>.

Những phần tiêu đề được hiển thị to và in đậm hơn để phân biệt chúng với các phần còn lại của văn bản. Chúng ta cũng có thể hiển thị phần tiêu đề một trong sáu kích thước từ H1 đến H6. Tất cả những gì chúng ta làm là định rõ kích thước H1, H2... Trình duyệt chú trọng đến cách hiển thị

Ví dụ 1:

```
<HTML>
  <HEAD>
    <TITLE>Introduction to HTML</TITLE>
  </HEAD>
  <H1>Introduction to HTML</H1>
  <H2>Introduction to HTML</H2>
  <H3>Introduction to HTML</H3>
  <H4>Introduction to HTML</H4>
  <H5>Introduction to HTML</H5>
  <H6>Introduction to HTML</H6>
</HTML>
```



Hình 2.1: Kết quả ví dụ 1

2.3 THỂ KHỐI , <DIV>

Có những trường hợp chúng ta muốn chia văn bản trong một trang web thành những khối thông tin logic. Chúng ta cũng có thể áp dụng những thuộc tính thông thường cho toàn bộ khối. Phần tử DIV và SPAN được sử dụng để nhóm nội dung lại với nhau. Phần tử DIV dùng để chia tài liệu thành các thành phần có liên quan với nhau. Phần tử SPAN dùng để chỉ một khoảng các ký tự.

Phần tử SPAN dùng để định nghĩa nội dung trong dòng(in-line) còn phần tử DIV dùng để định nghĩa nội dung mức khối (block-level)

Ví dụ 5:

```
<HTML>

  <HEAD>

    <TITLE>Learning HTML</TITLE>

  </HEAD>

  <BODY>

    <DIV>

      Division 1

      <P> The DIV element is used to group elements

      <P>Typically, DIV is used for block level elements

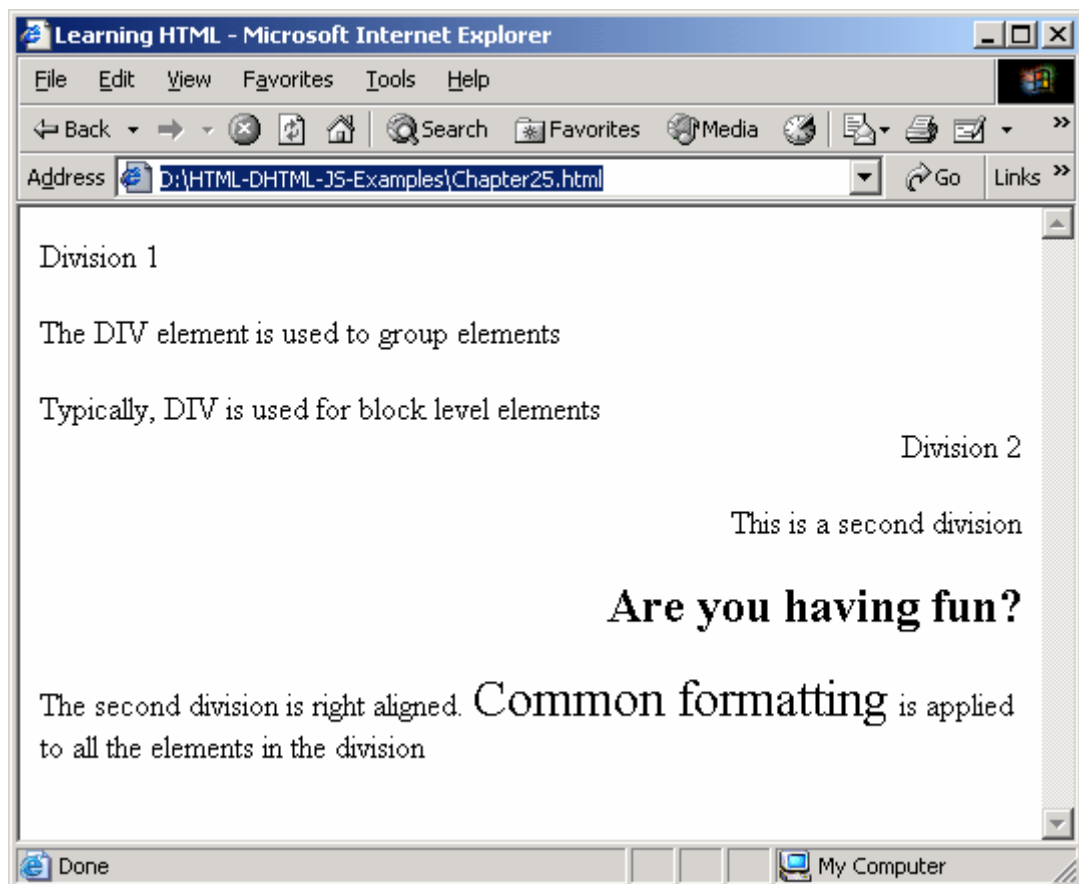
    </DIV>

    <DIV align = right>

      Division 2
```

```
<P>This is a second division  
<BR>  
<H2>Are you having fun?</H2>  
</DIV>  
<P>The second division is right aligned.  
<SPAN STYLE = font-size:25; Color:blue>Common formatting  
</SPAN>is applied to all the elements in the division  
</BODY>  
</HTML>
```

Chú ý: Phần tử SPAN không hiển thị trong Netscape



Hình 2.5: Kết quả ví dụ 5

2.4 DANH SÁCH

Danh sách dùng để nhóm dữ liệu một cách logic. Chúng ta có thể thêm các danh sách vào tài liệu HTML để nhóm các thông tin có liên quan lại với nhau.

Ví dụ,

- Roses
- Sunflowers
- Orchids

Apples
Oranges
Daffodils
Mangoes

Có thể được nhóm thành:

Fruits

Apples
Mangoes
Oranges

Flowers

Daffodils
Orchids
Roses
Sunflowers

Các loại danh sách mà bạn có thể chèn vào tài liệu HTML là:

- ☐ Danh sách không thứ tự
- ☐ Danh sách có thứ tự
- ☐ Danh sách định nghĩa

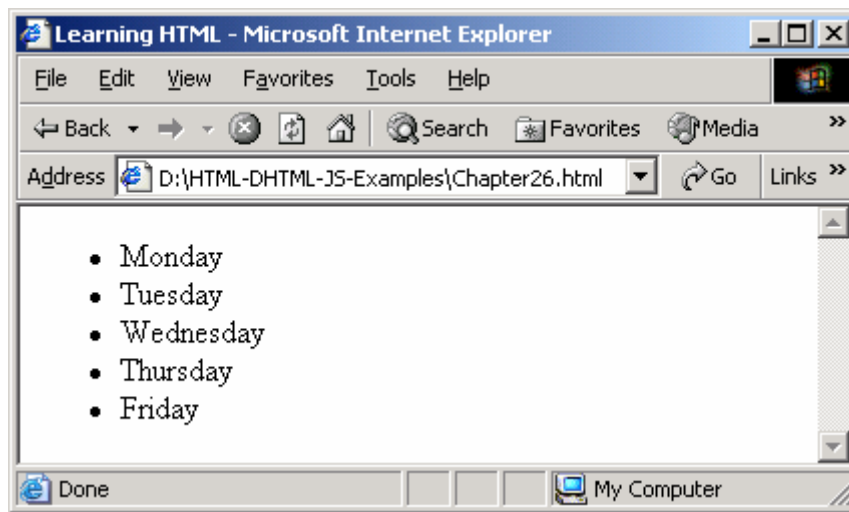
2.6.1 Danh sách không thứ tự

Đây là loại danh sách đơn giản nhất mà bạn có thể thêm vào tài liệu HTML. Danh sách không thứ tự là một “bulleted list”. Các mục được bắt đầu bằng dấu chấm tròn “bullet”. Danh sách không thứ tự được nằm trong cặp thẻ `...`. Mỗi mục trong danh sách được đánh dấu bằng thẻ ``. LI được viết tắt của từ List Item. Thẻ kết thúc `` là tùy chọn (không bắt buộc).

Ví dụ 6:

```
<HTML>
  <HEAD>
    <TITLE>Learning HTML</TITLE>
  </HEAD>
  <BODY>
    <UL>
      <LI>Monday
      <LI>Tuesday
      <LI>Wednesday
      <LI>Thursday
      <LI>Friday
```

```
</UL>
</BODY>
</HTML>
```



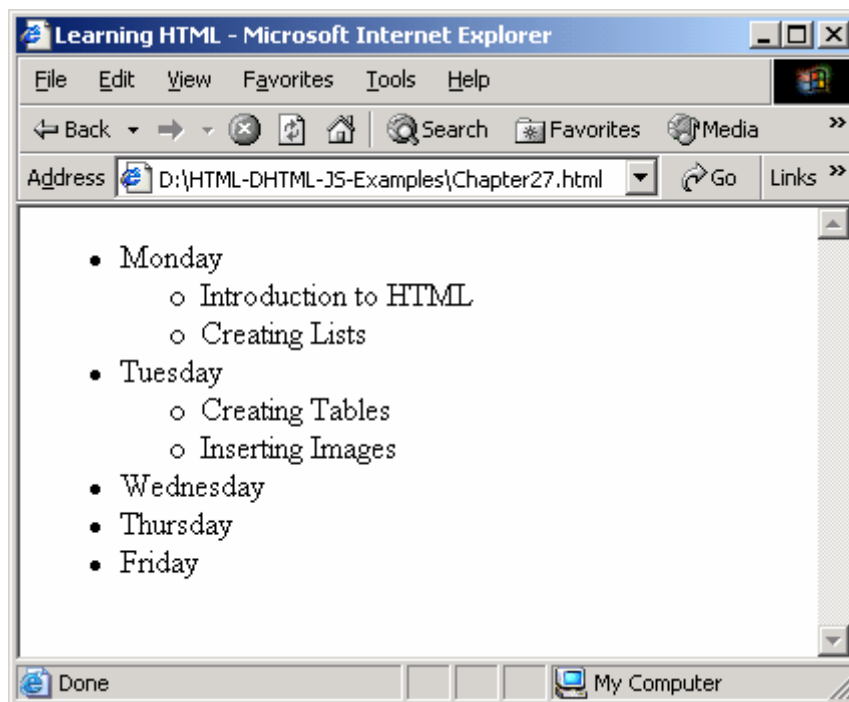
Hình 2.6: Kết quả ví dụ 6

Chúng ta có thể tạo ra các danh sách lồng nhau để mô tả nhóm con của thông tin.

Ví dụ 7:

```
<HTML>
  <HEAD>
    <TITLE>Learning HTML</TITLE>
  </HEAD>
  <BODY>
    <UL>
      <LI>Monday
        <UL>
          <LI>Introduction to HTML
          <LI>Creating Lists
        </UL>
      <LI>Tuesday
        <UL>
          <LI>Creating Tables
          <LI>Inserting Images
        </UL>
      <LI>Wednesday
      <LI>Thursday
      <LI>Friday
    </UL>
  </BODY>
```

</HTML>



Hình 2.7: Kết quả ví dụ 7

Thuộc tính TYPE có thể được dùng để định dạng các bullets để hiển thị cho các mục của danh sách.

Thuộc tính	Thẻ	Ví dụ
Bulleted	 và 	<pre><HTML> <HEAD> <TITLE>Learning HTML</TITLE> </HEAD> <BODY> Monday Introduction to HTML Creating Lists Tuesday Creating Tables Inserting Images </pre>

		<pre> Wednesday Thursday Friday </BODY> </HTML> </pre>
Customized bullets	<pre> <LI TYPE = SQUARE> Square bullets <LI TYPE = DISC> Sphere bullets <LI TYPE = CIRCLE> Round bullets </pre>	<pre> <HTML> <HEAD> <TITLE>Learning HTML</TITLE> </HEAD> <BODY> Monday <LI TYPE = DISC>Introduction to HTML <LI TYPE = DISC>Creating Lists Tuesday <LI TYPE = SQUARE>Creating Tables <LI TYPE = SQUARE>Inserting Images Wednesday <LI TYPE = CIRCLE>Creating Forms <LI TYPE = CIRCLE>Working with Frames Thursday Friday </BODY> </HTML> </pre>

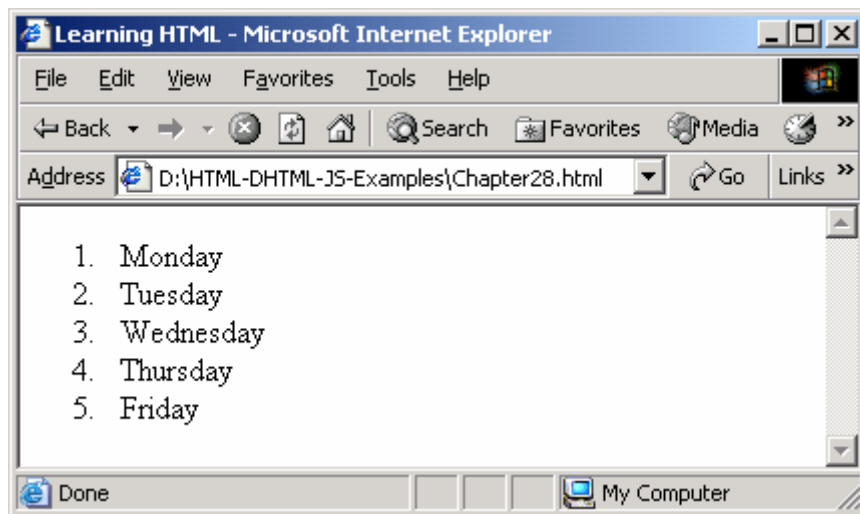
Chú ý: Thuộc tính TYPE không được hiển thị trong Internet Explorer

2.6.2 Danh sách có thứ tự

Danh sách có thứ tự nằm trong cặp thẻ `...`. Danh sách có thứ tự cũng hiển thị các mục danh sách. Sự khác nhau là các mục danh sách hiển thị theo thứ tự được tạo ra một cách tự động.

Ví dụ 8:

```
<HTML>
  <HEAD>
    <TITLE>Learning HTML</TITLE>
  </HEAD>
  <BODY>
    <OL>
      <LI>Monday
      <LI>Tuesday
      <LI>Wednesday
      <LI>Thursday
      <LI>Friday
    </OL>
  </BODY>
</HTML>
```



Hình 2.8: Kết quả ví dụ 8

Chúng ta có thể đặt các thuộc tính để định nghĩa hệ thống số mà được tạo ra cho các mục danh sách.

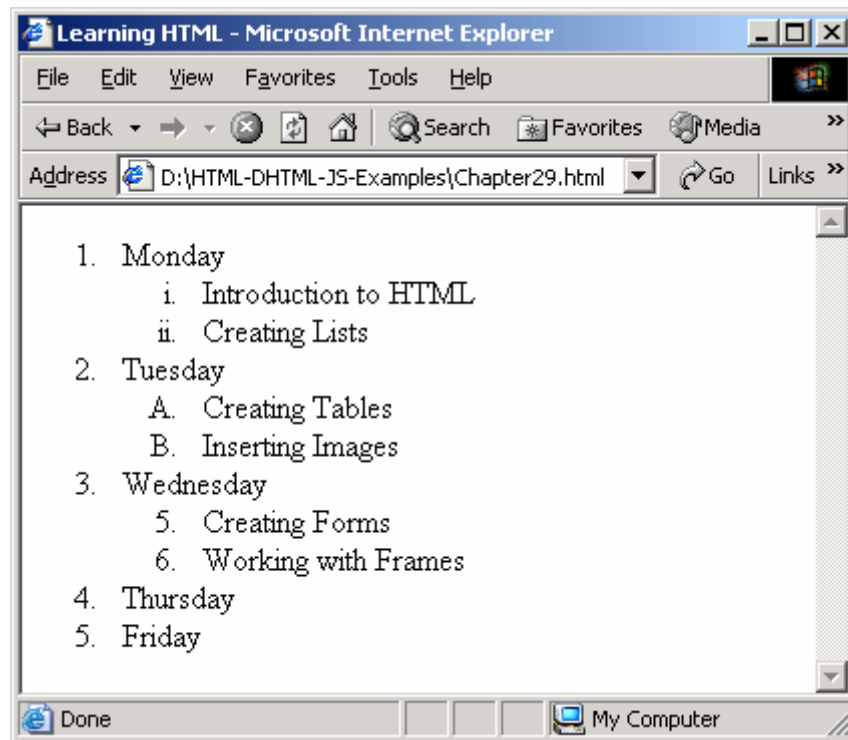
Thuộc tính	Thẻ
Upper Roman	<code><LI TYPE = I></code>
Lower Roman	<code><LI TYPE = i></code>
Uppercase	<code><LI TYPE = A></code>

Lowercase	<LI TYPE = a>
Bắt đầu với một số khác lớn hơn 1	<OL START = n>

Thuộc tính START xác định số khởi tạo ban đầu của danh sách.

Ví dụ 9:

```
<HTML>
  <HEAD>
    <TITLE>Learning HTML</TITLE>
  </HEAD>
  <BODY>
    <OL>
      <LI>Monday
        <OL>
          <LI TYPE = i>Introduction to HTML
          <LI TYPE = i>Creating Lists
        </OL>
      <LI>Tuesday
        <OL>
          <LI TYPE = A>Creating Tables
          <LI TYPE = A>Inserting Images
        </OL>
      <LI>Wednesday
        <OL START = 5>
          <LI >Creating Forms
          <LI >Working with Frames
        </OL>
      <LI>Thursday
      <LI>Friday
    </OL>
  </BODY>
</HTML>
```



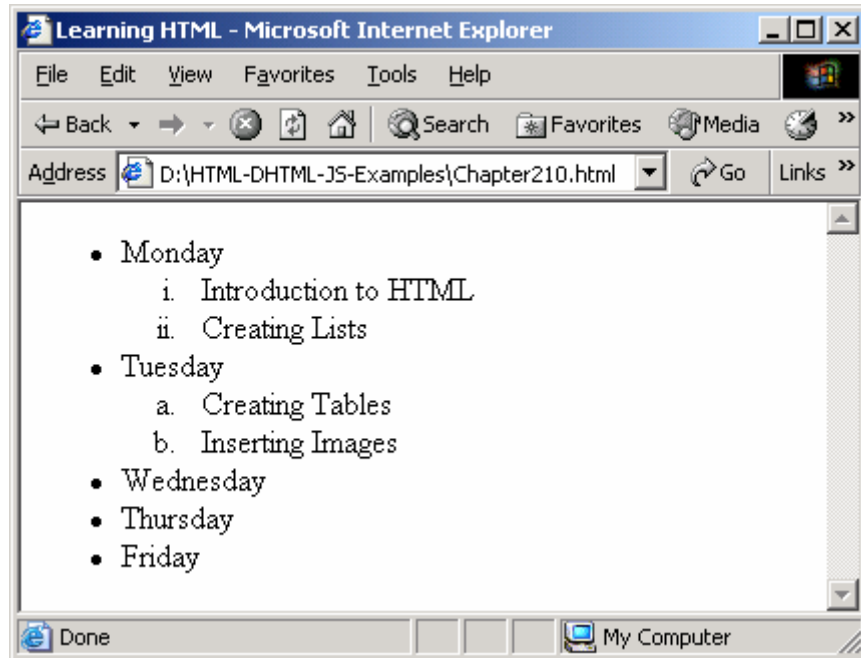
Hình 2.9: Kết quả ví dụ 9

Chúng ta có thể lồng các loại danh sách lại với nhau. Có thể lồng các danh sách có thứ tự vào trong các danh sách không thứ tự và ngược lại.

Ví dụ 10:

```
<HTML>
  <HEAD>
    <TITLE>Learning HTML</TITLE>
  </HEAD>
  <BODY>
    <UL>
      <LI>Monday
        <OL>
          <LI TYPE = i>Introduction to HTML
          <LI TYPE = i>Creating Lists
        </OL>
      <LI>Tuesday
        <OL>
          <LI TYPE = a>Creating Tables
          <LI TYPE = a>Inserting Images
        </OL>
      <LI>Wednesday
      <LI>Thursday
```

```
<LI>Friday
</UL>
</BODY>
</HTML>
```



Hình 2.10: Kết quả ví dụ 10

2.5 THẺ KẼ ĐƯỜNG NGANG: <HR>

Thẻ <HR>(horizontal rule) được dùng để kẻ một đường ngang trên trang. Những thuộc tính sau giúp điều khiển các đường nằm ngang. Nó chỉ có thẻ bắt đầu, không có thẻ kết thúc và không có nội dung.

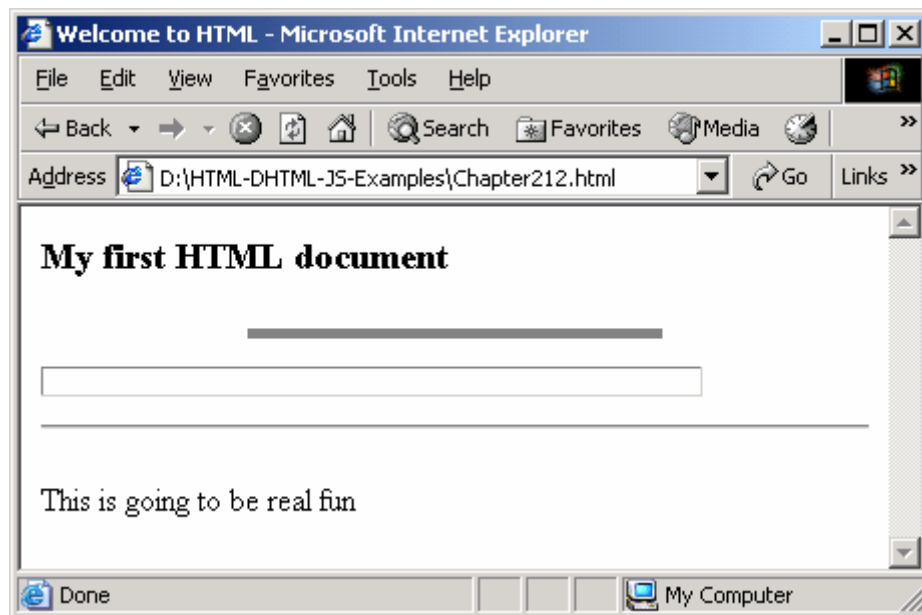
Thuộc tính	Mô tả
align	Chỉ định vị trí của đường nằm ngang. Chúng ta có thể canh lề center(giữa) right(phải) left(trái). Ví dụ align=center
width	Chỉ độ dài của đường thẳng. Nó có thể xác định bằng các pixel hoặc tính theo phần trăm. Mặc định là 100%, nghĩa là toàn bộ bề ngang của tài liệu.
size	Chỉ độ dày của đường thẳng và được xác định bằng các pixel.
noshade	Chỉ đường được hiển thị bằng màu đặc thay vì có bóng.

Ví dụ 12:

```
<HTML>
  <HEAD>
    <TITLE>Welcome to HTML</TITLE>
  </HEAD>
  <BODY>
```



```
<H3>My first HTML document</H3>
<HR noshade size = 5 align = center width = 50%>
<HR size = 15 align = left width = 80%>
<HR>
<P>This is going to be real fun
/BODY>
</HTML>
```



Hình 2.12: Kết quả ví dụ 12

2.6 SỬ DỤNG FONT

Thẻ dùng để điều khiển sự hiển thị văn bản trên trang web. Chúng ta cũng có thể chỉ định các thuộc tính như kích thước, màu sắc, kiểu...

Chúng ta có thể đặt các thuộc tính cho cả tài liệu bằng cách đặt phần tử vào bên trong thẻ <BODY>. Ngoài ra, thuộc tính FONT có thể đặt cho từng từ, từng phần và từng phần tử trong trang.

Thuộc tính	Mô tả
COLOR	Được dùng để chỉ màu của font. Chúng ta có thể dùng tên màu hoặc giá trị thập phân để xác định màu.
SIZE	Được dùng để chỉ kích thước của font. Chúng ta có thể xác định các kích thước FONT từ 1 cho đến 7. Kích thước lớn nhất là 7 và nhỏ nhất là 1. Chúng ta có thể dùng một kích thước chuẩn và chỉ ra những kích thước tiếp theo liên quan đến kích thước chuẩn. Ví dụ, nếu kích thước chính là 3, thì SIZE=+4 sẽ tăng lên 7 SIZE=-1 sẽ giảm xuống 2
FACE	Được dùng để chỉ định kiểu font (phông chữ)

Các thuộc tính của FONT có thể kết hợp trong cùng một thẻ.

Ví dụ 13:

```
<HTML>

  <HEAD>

    <TITLE>Welcome to HTML</TITLE>

  </HEAD>

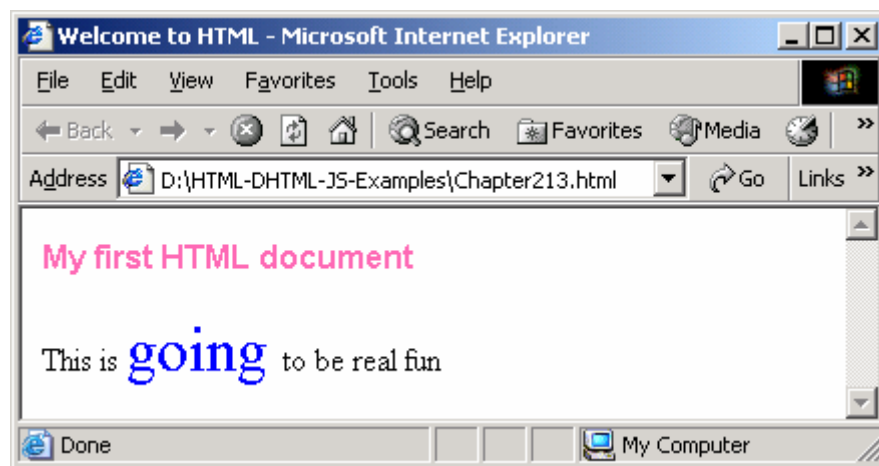
  <BODY>

    <H3><FONT SIZE = 3 COLOR = HOTPINK FACE = Arial>My first HTML
document</FONT></H3>

    <P>This is <FONT COLOR = BLUE SIZE = 6>going </FONT>to be real
fun

  </BODY>

</HTML>
```



Hình 2.13 Kết quả của ví dụ 13

2.7 SỬ DỤNG MÀU SẮC

Chúng ta có thể thêm màu vào trang và vào các phần tử trong trang. COLOR là thuộc tính có thể được sử dụng với nhiều phần tử như phần tử FONT và BODY.

Ví dụ 14:

```
<HTML>

  <HEAD>

    <TITLE>Learning HTML</TITLE>

  </HEAD>

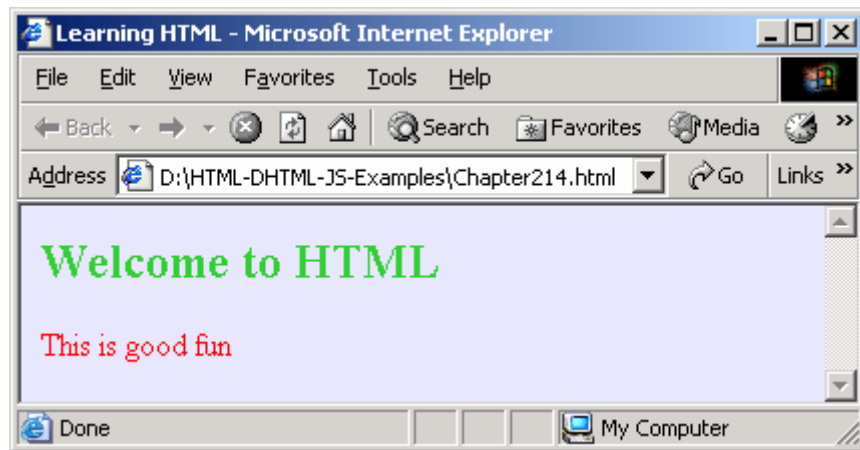
  <BODY BGCOLOR=lavender>

    <H2><FONT COLOR = LIMEGREEN>Welcome to HTML</FONT></H2>

    <P><FONT COLOR = RED>This is good fun</FONT></P>

  </BODY>

</HTML>
```



Hình 2.14: Kết quả của ví dụ 14

Có 3 kiểu màu chính: đỏ, xanh và xanh da trời. Mỗi màu chính được xem như một bộ hai số của hệ 16.

#RRGGBB

Số thập lục phân 00 chỉ 0% của màu trong khi đó số thập lục phân FF chỉ 100% của màu. Giá trị cuối cùng là một mã sáu chữ số chỉ màu.

Mã thập lục phân	Màu
#FF0000	Red
#00FF00	Green
#0000FF	Blue
#000000	Black
#FFFFFF	White

2.8 SỬ DỤNG HÌNH ẢNH TRONG TÀI LIỆU HTML

“Một bức tranh đáng giá ngàn từ”, tất nhiên điều này có thể áp dụng cho một trang web. Những hình ảnh được chèn vào trong trang web được gọi là những ảnh nội tuyến. Ảnh có thể là biểu tượng, bullet, ảnh, logo công ty và nhiều cái khác.

Ngày nay có nhiều định dạng đồ họa đang được sử dụng. Tuy nhiên, trên Web có khác đôi chút. Ba định dạng đồ họa thông thường được hiển thị trên hầu hết các trình duyệt là:

☐ **Ảnh GIF (Graphics Interchange Format) (.GIF)**

GIF là định dạng thông thường nhất được dùng trong những tài liệu HTML. Những file GIF được định dạng không phụ thuộc vào định dạng nền và hỗ trợ 256 màu. Ưu điểm của các file GIF là khá dễ để chuyển tải, ngay cả kết nối sử dụng MODEM tốc độ chậm

Có hai tiêu chuẩn cho các file .gif -87a và 89a (hỗ trợ trong suốt).

- **Định dạng GIF xen kẽ(Interlaced GIF format):** Một file ảnh thông thường hiển thị ảnh một lần một dòng. Mặc dù các ảnh xen kẽ được hiển thị một lần một dòng, nhưng thứ tự có thay đổi
- **Ảnh GIF trong suốt (Transparent GIF images):** Ảnh GIF trong suốt là ảnh trong đó nền của ảnh cùng màu với trang web. Ví dụ các biểu tượng và bullet có nền trong suốt vì vậy chúng hợp với màu nền của tài liệu.
- **Ảnh JPEG (Joint Photographic Expert Group) (.JPEG)**

Cách nén JPEG là một lược đồ nén *mất thông tin*. Điều này có nghĩa là ảnh sau khi bị nén không giống như ảnh gốc. Tuy nhiên trong quá trình phát lại thì ảnh tốt gần như ảnh gốc. Khi bạn lưu một file với định dạng JPG, bạn có thể định tỉ lệ nén. Tỉ lệ càng cao thì ảnh càng ít giống ảnh gốc.

JPEG hỗ trợ hơn 16 triệu màu và thường được sử dụng cho các ảnh có màu thực.

Cả hai file ảnh dạng JPEG (đuôi mở rộng là .jpg) và các dạng GIF đều nén ảnh để đảm bảo chế độ chuyển tải qua internet được nhanh hơn. Ảnh JPG có thể được nén nhiều hơn nhưng chậm hơn trong quá trình hiển thị so với ảnh .GIF. Có lẽ đó là lý do tại sao ảnh .GIF được phổ biến trong tài liệu HTML.

- **PNG (Portable Network Graphics)**

Định dạng cho một file PNG là “lossless” (không mất thông tin). Trong nén “lossless”, dữ liệu ảnh được nén mà không bỏ chi tiết. Các file PNG hỗ trợ ảnh màu thực và dải màu xám. Các file PNG cũng có thể được nén và chuyển qua mạng.

Khi quyết định định dạng đồ họa, chúng ta nên nhớ một vài yếu tố:

- **Chất lượng của ảnh** – Chất lượng của ảnh có quan trọng trong quá trình hiển thị không?
- **Độ lớn dữ liệu** – Kích cỡ file càng lớn thì thời gian truyền càng cao.
- **Các yêu cầu hiển thị** - Ảnh hỗ trợ trong suốt, hiển thị tuần tự hay xen kẽ.

Chèn ảnh

Thẻ IMG dùng để chèn những ảnh vào trong tài liệu HTML. Chúng ta cũng có thể đặt thẻ IMG tại vị trí mà ảnh được hiển thị. Thẻ IMG không có nội dung, nó hiển thị nội dung bằng cách xác định thuộc tính SRC. Cú pháp là:

```
<IMG SRC="URL">
```

Trong đó SRC (source) là thuộc tính và giá trị là một URL, chỉ vị trí chính xác của file ảnh

Đôi khi, chỉ hình ảnh không thể nói lên tất cả. Chúng ta cần phải cung cấp cho người dùng một vài giới thiệu về mục đích của hình ảnh. Bạn có thể canh lề ảnh cùng với văn bản.

Thuộc tính ALIGN của thẻ IMG có thể được sử dụng để điều chỉnh canh lề của ảnh với văn bản xung quanh.

Trong đó, vị trí của ảnh có thể là trên(TOP), dưới(BOTTOM), ở giữa(MIDDLE), trái(LEFT) hoặc phải(RIGHT).

Ví dụ 15:

```
<HTML>

  <HEAD>

    <TITLE>Inserting an Image</TITLE>

  </HEAD>

  <H1><FONT SIZE=3 COLOR=FORESTGREEN>
  <B>Inserting an Image</B></FONT></H1>

  <HR>

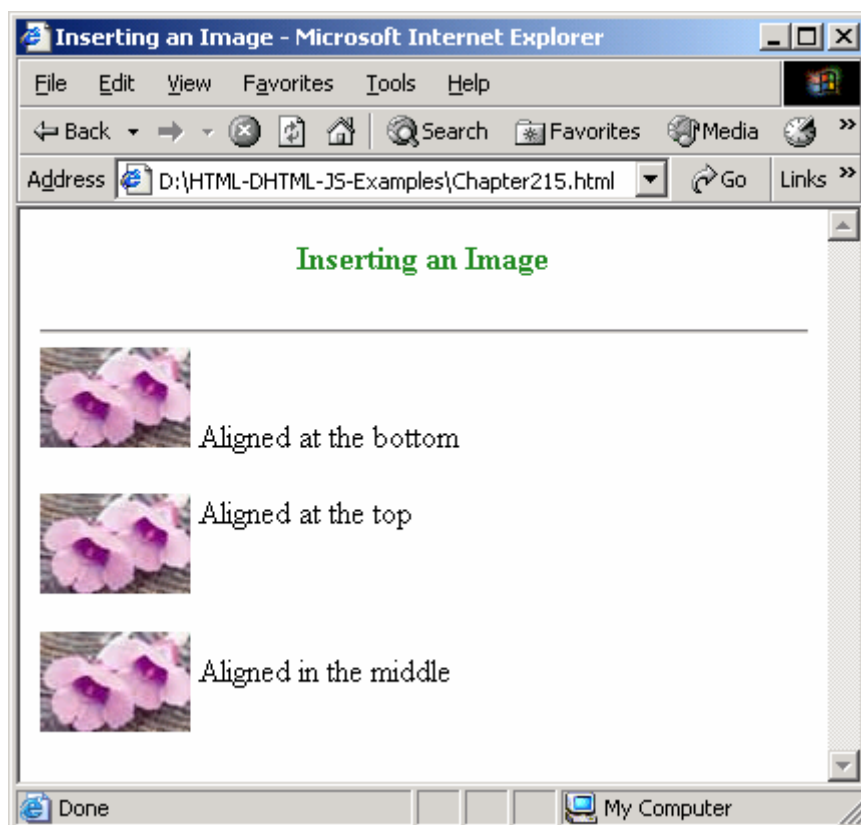
  <BODY>

    <IMG ALIGN=BOTTOM SRC=Flowers.jpg> Aligned at the bottom</P>
    <P>
    <IMG ALIGN=TOP SRC=Flowers.jpg> Aligned at the top</P>
    <P>
    <IMG ALIGN=MIDDLE SRC=Flowers.jpg> Aligned in the middle</P>

  </BODY>

</HTML>
```

Chú ý: file Flowers.jpg nằm trong cùng thư mục với file HTML nguồn.



Hình 2.15: Kết quả ví dụ 15

Chú ý rằng một vài trình duyệt không hiển thị những ảnh đồ họa. Trong trường hợp này, chúng ta cần xác định một dòng chú thích thay thế trong tài liệu HTML. Thuộc tính ALT được sử dụng để chỉ nội dung ảnh của bạn.

Ví dụ 16:

```
<HTML>

  <HEAD>

    <TITLE>Inserting an Image</TITLE>

  </HEAD>

  <H1><FONT SIZE=3 COLOR=FORESTGREEN>
  <B>Inserting an Image</B>
  </FONT></H1>

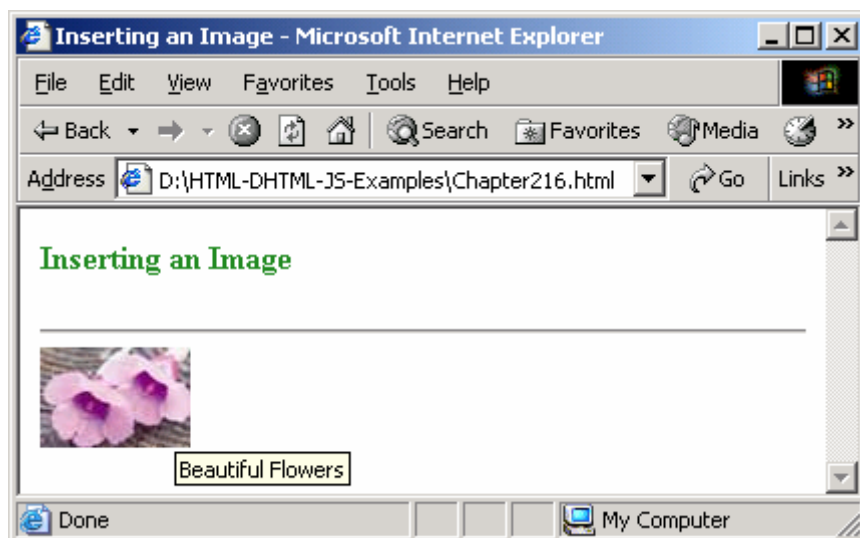
  <HR>

  <BODY>

    <IMG SRC=Flowers.jpg ALT="Beautiful Flowers">

  </BODY>

</HTML>
```



Hình 2.16: Kết quả ví dụ 16 (dòng văn bản giải thích hiển thị khi di chuyển chuột vào ảnh)

Nền của trang cũng quan trọng như văn bản. Người ta thường sử dụng màu cho trang web, mục đích là làm cho nội dung được nổi bật. Chúng ta cũng có thể sử dụng ảnh để làm nền. Để làm điều đó, ta cần phải dùng thuộc tính BACKGROUND trong thẻ BODY.

```
<BODY BACKGROUND=bgimage.gif> (Chỉ ra URL hoàn chỉnh của ảnh)
```

Nếu ảnh nhỏ hơn phạm vi hiển thị của tài liệu thì ảnh được xếp kề nhau để lấp đầy toàn bộ vùng hiển thị.

Ảnh thường cuộn theo văn bản khi người dùng kéo thanh cuộn trong trình duyệt. Nếu không muốn như vậy và thay vào đó ta muốn tạo ra hiệu ứng hình mờ, nghĩa là văn bản thì cuộn còn ảnh thì đứng yên, ta thiết lập thuộc tính BGPROPERTIES trong thẻ BODY có giá trị là FIXED

```
<BODY BACKGROUND=bgimage.gif BGPROPERTIES=FIXED>
```

Các ảnh được chèn vào tài liệu HTML cũng có thể sử dụng như siêu liên kết. Những ảnh như thế gọi là siêu ảnh. Khi người dùng kích vào ảnh, sẽ hiển thị tài liệu hoặc file được chỉ ra trong URL. Để làm điều này, ta cần lồng ảnh vào trong thẻ neo (anchor)

CHƯƠNG 3: SỬ DỤNG BẢNG

Kết thúc chương này, bạn có thể:

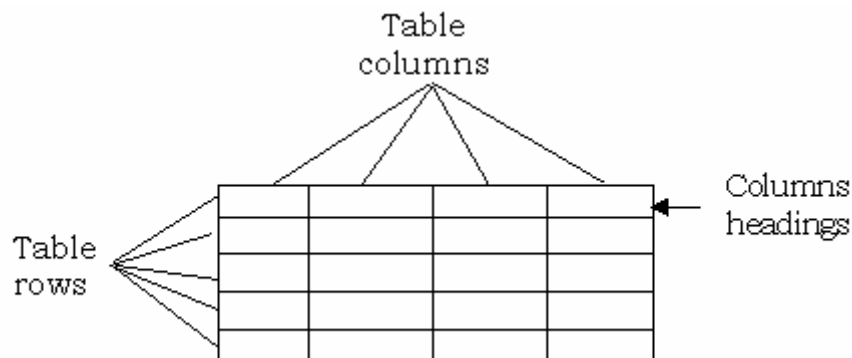
- Biết cách sử dụng bảng (table).
- Chèn các đối tượng đa phương tiện “Multimedia” vào tài liệu HTML

3.1 GIỚI THIỆU

Chúng ta đã học cách nhóm các dữ liệu có liên quan vào các danh sách, nhưng còn có một cách khác để điều khiển việc hiển thị văn bản trên trang web, bằng cách dùng các bảng. Trong chương này chúng ta sẽ học cách tạo bảng. Phần cuối của chương này sẽ nói về làm thế nào để chèn các điều khiển đa phương tiện “multimedia” vào bên trong tài liệu HTML.

3.2 CÁCH TẠO BẢNG

Chúng ta có thể sử dụng bảng để hiển thị dữ liệu dưới dạng các hàng và các cột. Bảng giúp cho chúng ta điều khiển, xác định và sắp xếp vị trí của văn bản và hình ảnh trên trang web, thay vì giao tất cả các việc đó cho trình duyệt.



Hình 3.1 Mô hình của bảng

3.2.1 Thẻ dùng để tạo bảng

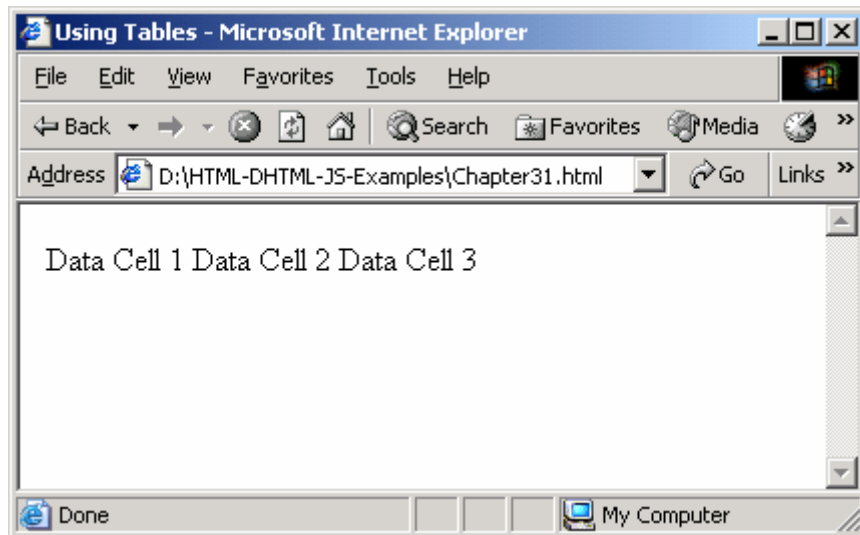
Thẻ <TABLE> được dùng để tạo bảng trong tài liệu HTML. Các thuộc tính của phần tử <TABLE> được áp dụng cho bảng, nhưng không cho dữ liệu hiển thị trên bảng. Đơn vị cơ bản của bảng là một ô và được định nghĩa bằng thẻ <TD>.

Ví dụ 1:

```
<HTML>
  <HEAD>
    <TITLE>Using Tables</TITLE>
  </HEAD>
  <BODY>
    <TABLE>
      <TD>Data Cell 1</TD>
      <TD>Data Cell 2</TD>
```



```
<TD>Data Cell 3</TD>
</TABLE>
</BODY>
</HTML>
```

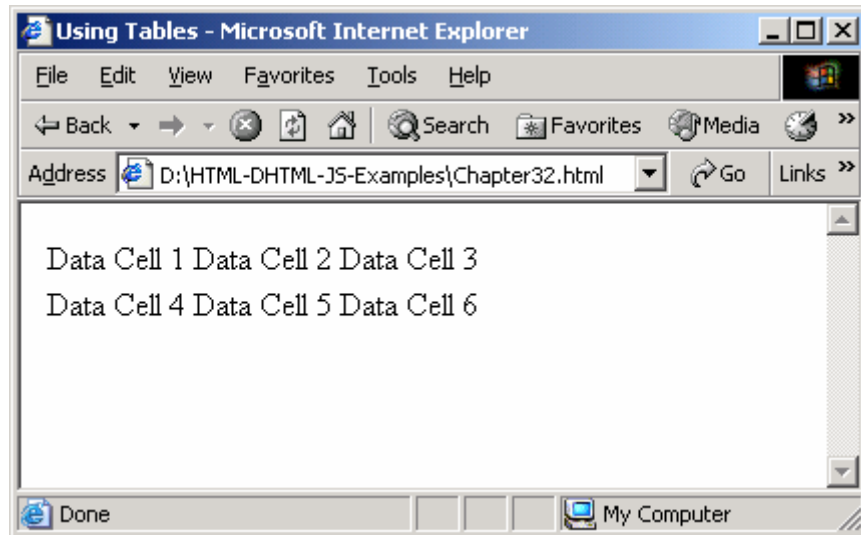


Hình 3.2: Kết quả của ví dụ 1

Một hàng của bảng được định nghĩa bằng thẻ <TR>

Ví dụ2:

```
<HTML>
  <HEAD>
    <TITLE>Using Tables</TITLE>
  </HEAD>
  <BODY>
    <TABLE>
      <TR>
        <TD>Data Cell 1</TD>
        <TD>Data Cell 2</TD>
        <TD>Data Cell 3</TD>
      <TR>
        <TD>Data Cell 4</TD>
        <TD>Data Cell 5</TD>
        <TD>Data Cell 6</TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```



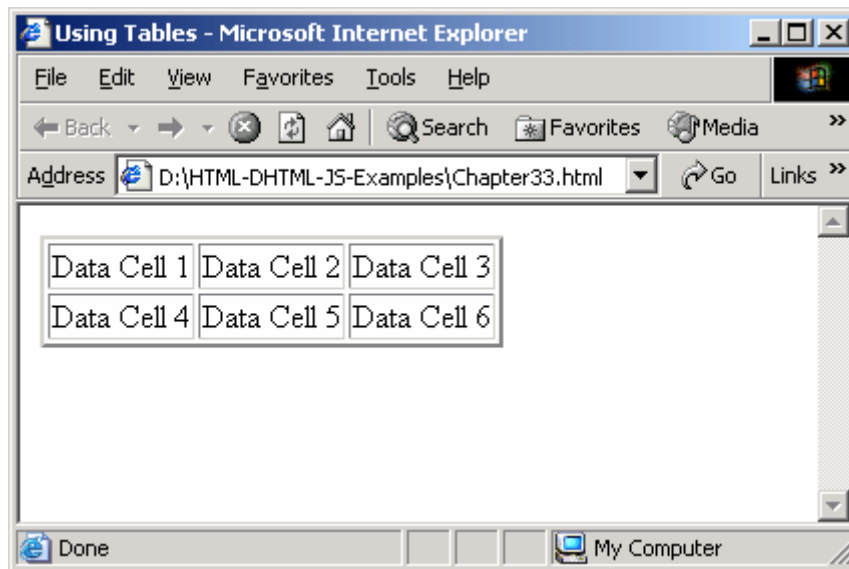
Hình 3.3: Kết quả của ví dụ 2

Các ô tạo thành một hàng. Các hàng tạo thành bảng. Điều này được nói đến trong cú pháp của HTML được sử dụng để tạo bảng. Thẻ TD được lồng trong thẻ TR. Thẻ TR được nằm trong cặp thẻ đóng và mở TABLE.

Thuộc tính BORDER có thể được sử dụng để định nghĩa các ô và cấu trúc của bảng. thuộc tính này chỉ độ rộng của đường viền. Nếu giá trị được đặt là 0 thì đường viền sẽ không hiển thị.

Ví dụ 3:

```
<HTML>
  <HEAD>
    <TITLE>Using Tables</TITLE>
  </HEAD>
  <BODY>
    <TABLE BORDER = 2>
      <TR>
        <TD>Data Cell 1</TD>
        <TD>Data Cell 2</TD>
        <TD>Data Cell 3</TD>
      <TR>
        <TD>Data Cell 4</TD>
        <TD>Data Cell 5</TD>
        <TD>Data Cell 6</TD>
      </TR>
    </TABLE>
  </BODY>
</HTML>
```



Hình 3.4: Kết quả của ví dụ 3

Mỗi cột trong bảng, phần tiêu đề có thể định nghĩa. Thẻ <TH> định nghĩa tiêu đề cho cột. Ví dụ:

```
<TH> Employee Name </TH>
```

Phần tử CAPTION được dùng để thêm vào một chú thích cho bảng. ở đây bạn sử dụng nó để mô tả bảng. Ví dụ,

```
<CAPTION> Creating a Table </CAPTION>
```

<CAPTION> nằm ngay sau thẻ mở <TABLE>

3.2.2 Chèn hàng và cột

Những thẻ <TD> và <TR> có thể được dùng để chèn theo thứ tự các hàng và các cột vào trong bảng. Ví dụ được mô tả bên dưới.

Ví dụ 4:

```
<HTML>

  <HEAD>

    <TITLE>Using Tables</TITLE>

  </HEAD>

  <BODY>

    <TABLE BORDER = 2>

      <TR>

        <TD>Data Cell 1</TD>

        <TD>Data Cell 2</TD>

        <TD>Data Cell 3</TD>

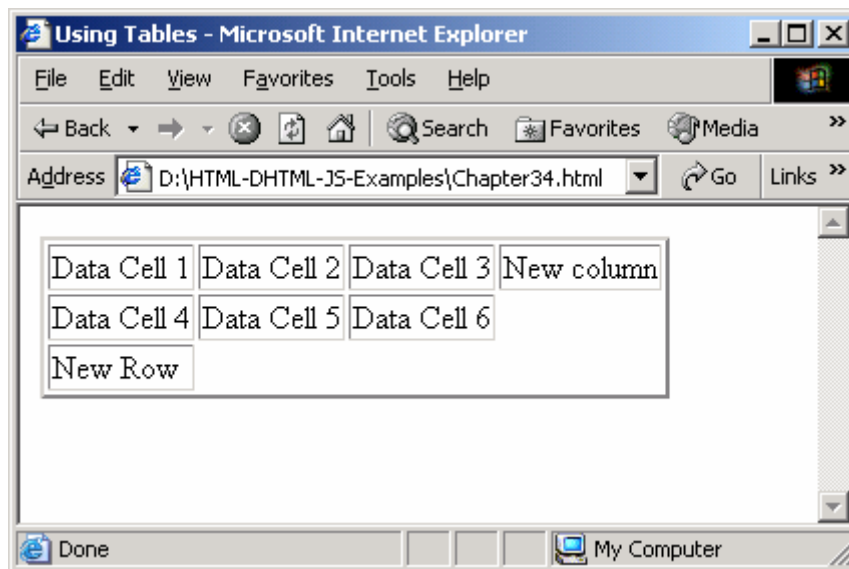
        <TD>New column</TD>

      <TR>

        <TD>Data Cell 4</TD>

        <TD>Data Cell 5</TD>
```

```
<TD>Data Cell 6</TD>
<TR>
<TD>New Row</TD>
</TABLE>
</BODY>
</HTML>
```



Hình 3.5: Kết quả của ví dụ 4

3.2.3 Xóa hàng và cột

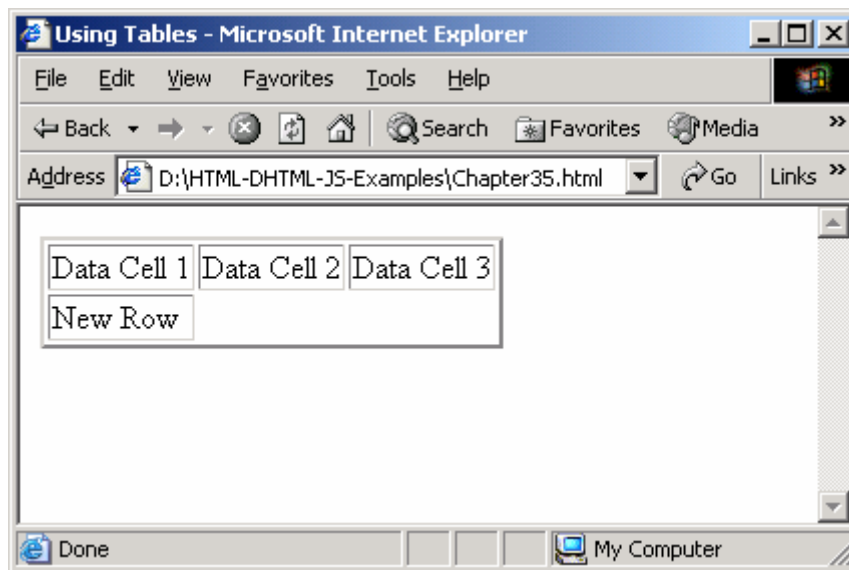
Để xóa một hàng, xóa thẻ `<TR>` tương ứng cùng với tất cả các thẻ `<TD>` bên trong nó từ một file mã nguồn HTML. Tương tự, để xóa các cột

Hãy xem Hình 3.4. Để xóa dòng thứ hai, chúng ta cần xóa thẻ `<TD>` cho ô dữ liệu 4, 5 và 6 cùng với thẻ `<TR>` mà nó kèm theo những thẻ `<TD>` này từ file HTML nguồn. Để xóa cột cuối cùng, đó là *New column*, chúng ta cần xóa thẻ `<TD>` lập nên ô này

Ví dụ 5:

```
<HTML>
<HEAD>
<TITLE>Using Tables</TITLE>
</HEAD>
<BODY>
<TABLE BORDER = 2>
<TR>
<TD>Data Cell 1</TD>
<TD>Data Cell 2</TD>
<TD>Data Cell 3</TD>
<TR>
<TD>New Row</TD>
```

```
</TABLE>
</BODY>
</HTML>
```



Hình 3.6: Kết quả của ví dụ 5

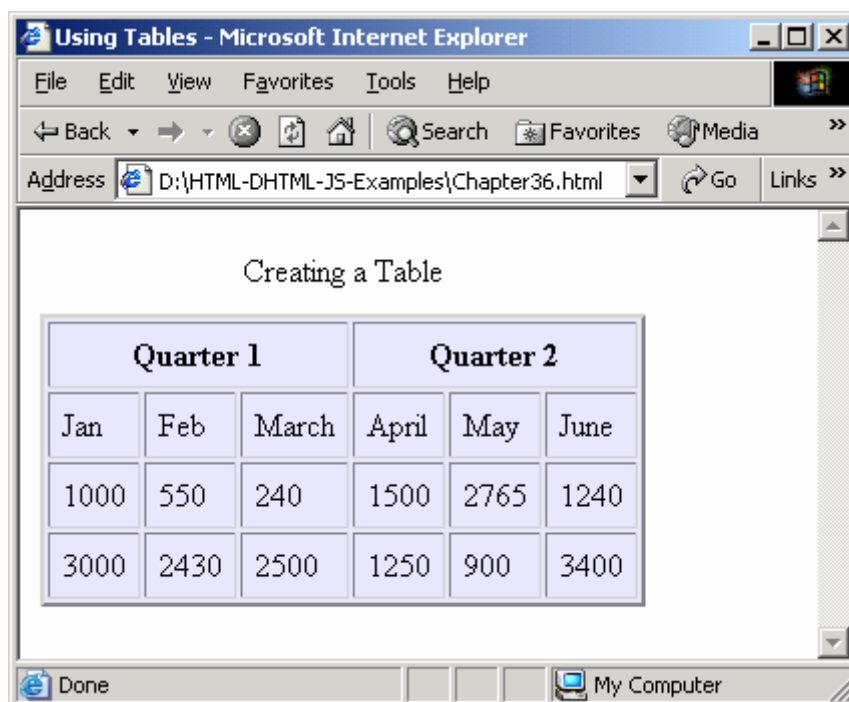
3.2.4 Trộn ô: kết hợp cột hay dòng

Đôi khi chúng ta muốn nối các dòng và các cột vào trong một ô. Như vậy, chúng ta tạo một cột để kéo rộng ra cho hơn một dòng, hay tạo ra một dòng để kéo rộng ra cho hơn một cột. Thuộc tính COLSPAN và ROWSPAN được sử dụng để tạo ra những ô mà chúng có thể kéo rộng ra cho hơn một dòng hay cột. Thuộc tính COLSPAN được sử dụng với thẻ <TH>, trong khi đó thuộc tính ROWSPAN được sử dụng với thẻ <TD>.

Ví dụ 6:

```
<HTML>
<HEAD>
  <TITLE>Using Tables</TITLE>
</HEAD>
<BODY>
  <TABLE BORDER = 2 BGCOLOR = lavender CELLSPACING = 2
    CELLPADDING = 6>
    <CAPTION>Creating a Table</CAPTION>
    <TH ALIGN = CENTER COLSPAN = 3>Quarter 1</TH>
    <TH ALIGN = CENTER COLSPAN = 3>Quarter 2</TH>
    <TR>
      <TD>Jan</TD>
      <TD>Feb</TD>
      <TD>March</TD>
      <TD>April</TD>
      <TD>May</TD>
```

```
        <TD>June</TD>
    <TR>
        <TD>1000</TD>
        <TD>550</TD>
        <TD>240</TD>
        <TD>1500</TD>
        <TD>2765</TD>
        <TD>1240</TD>
    <TR>
        <TD>3000</TD>
        <TD>2430</TD>
        <TD>2500</TD>
        <TD>1250</TD>
        <TD>900</TD>
        <TD>3400</TD>
    </TABLE>
</BODY>
</HTML>
```



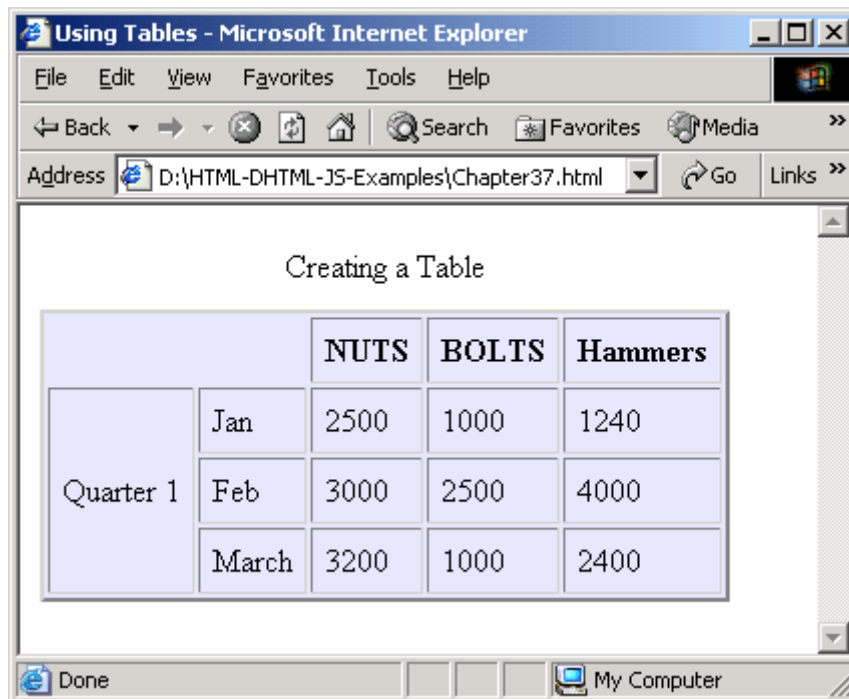
Hình 3.7: Kết quả của ví dụ 6

Ví dụ 7:

```
<HTML>
    <HEAD>
        <TITLE>Using Tables</TITLE>
```

```
</HEAD>

<BODY>
    <TABLE BORDER = 2 BGCOLOR = lavender CELLSPACING = 2
    CELLPADDING = 6>
        <CAPTION>Creating a Table</CAPTION>
        <TR>
            <TH></TH>
            <TH></TH>
            <TH>NUTS</TH>
            <TH>BOLTS</TH>
            <TH>Hammers</TH>
        <TR>
            <TD ROWSPAN = 3>Quarter 1</TD>
            <TD>Jan</TD>
            <TD>2500</TD>
            <TD>1000</TD>
            <TD>1240</TD>
        <TR>
            <TD>Feb</TD>
            <TD>3000</TD>
            <TD>2500</TD>
            <TD>4000</TD>
        <TR>
            <TD>March</TD>
            <TD>3200</TD>
            <TD>1000</TD>
            <TD>2400</TD>
        </TABLE>
    </BODY>
</HTML>
```



Hình 3.8: Kết quả của ví dụ 7

3.2.5 Định dạng cho ô

Có thể chỉ định được vị trí của văn bản bên trong mỗi ô của bảng. Những thuộc tính canh lề ngang (ALIGN) và canh lề dọc (VALIGN) được sử dụng để điều khiển việc canh lề trong các ô của bảng. Thuộc tính ALIGN có thể có các giá trị: trái, phải, giữa, đều hai bên. Thuộc tính VALIGN có thể có các giá trị: trên, giữa và dưới

Ngoài việc xác định vị trí nội dung của các ô trong bảng, chúng ta cũng có thể thay đổi kích thước, màu sắc của bảng. Chúng ta có thể xác định chiều rộng của bảng. Đó là vùng trên màn hình mà bảng sẽ mở rộng ra. Thuộc tính WIDTH của thẻ TABLE sẽ thật sự cần thiết. Giá trị của thuộc tính WIDTH có thể là một định dạng phần trăm hay thậm chí là các điểm. Chẳng hạn,

```
<TABLE WIDTH = 50%>
```

Điều này sẽ kéo rộng bảng đến 50% của màn hình

Nếu chúng ta muốn thêm một vài màu sắc cho bảng, chúng ta sử dụng thuộc tính BGCOLOR. Giá trị này có thể là tên của một màu hay là giá trị ở hệ thập lục phân.

Ví dụ 8:

```
<HTML>
  <HEAD>
    <TITLE>Using Tables</TITLE>
  </HEAD>
  <BODY>
    <TABLE BORDER = 2 WIDTH = 50% BGCOLOR = lavender>
```



```
<TR>

    <TD ALIGN = right VALIGN = bottom>Data Cell
1</TD>

    <TD>Data Cell 2</TD>
    <TD>Data Cell 3</TD>

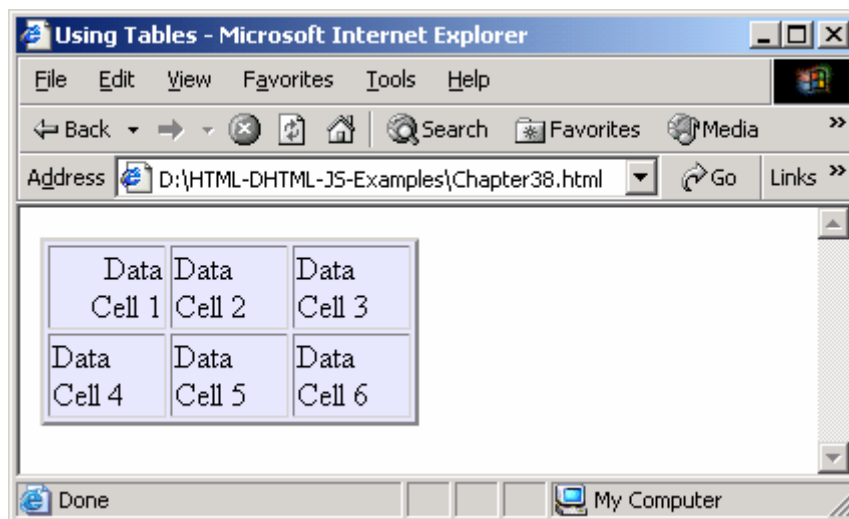
<TR>

    <TD>Data Cell 4</TD>
    <TD>Data Cell 5</TD>
    <TD>Data Cell 6</TD>

</TABLE>

</BODY>

</HTML>
```



Hình 3.9: Kết quả của ví dụ 8

Nếu nội dung của văn bản trong ô nhỏ hơn không gian trong ô đó thì sẽ có không gian trống là khoảng cách giữa văn bản và đường viền của ô. **CELLSPACING** là không gian giữa các ô và **CELLPADDING** là không gian giữa đường viền ô của bảng với nội dung văn bản được đặt trong ô. Như đã mô tả ở trên, thuộc tính **CELLSPACING** định nghĩa không gian giữa các ô, tính theo điểm (pixel). Thuộc tính **CELLPADDING** định nghĩa những không gian bên trong ô của bảng

Ví dụ 9:

```
<HTML>

    <HEAD>

        <TITLE>Using Tables</TITLE>

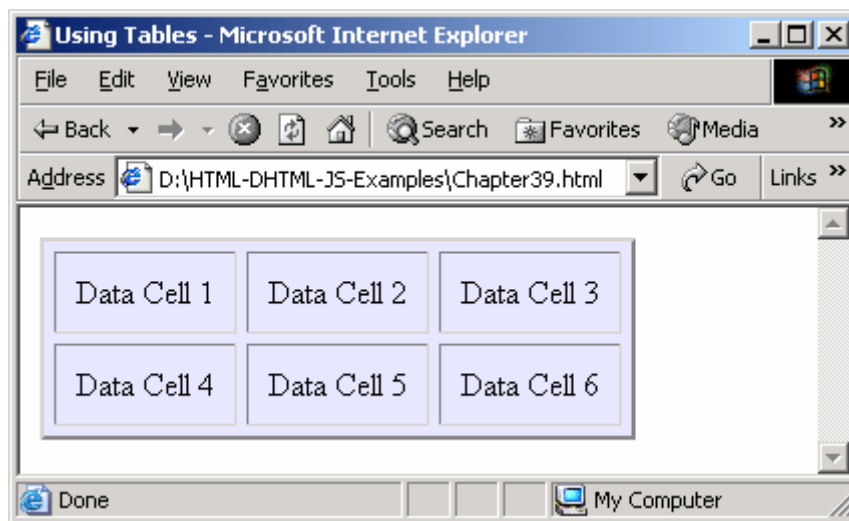
    </HEAD>

    <BODY>

        <TABLE BORDER = 2 BGCOLOR = lavender CELLSPACING = 5
        CELLPADDING = 10>

            <TR>
```

```
<TD>Data Cell 1</TD>
<TD>Data Cell 2</TD>
<TD>Data Cell 3</TD>
<TR>
<TD>Data Cell 4</TD>
<TD>Data Cell 5</TD>
<TD>Data Cell 6</TD>
</TABLE>
</BODY>
</HTML>
```



Hình 3.10: Kết quả của ví dụ 9

3.4 CHÈN MULTIMEDIA VÀO TÀI LIỆU HTML

Một trong những nhân tố lớn nhất trong sự phát triển của web đó là sự tích hợp của các đa phương tiện(multimedia) bên trong những tài liệu HTML. Khi những phiên bản đầu tiên của HTML ra đời, nó cũng đã bao gồm các đối tượng liên quan đến hình ảnh vào trong một tài liệu để cho phép nhúng hình ảnh nội tuyến vào nội dung tài liệu. Sau đó, HTML được mở rộng để cho phép nhúng không chỉ là những ảnh tĩnh mà còn là âm thanh và video. Hiện nay, tất cả những tính năng đó được phép sử dụng để làm phong phú thêm cho các trang web.

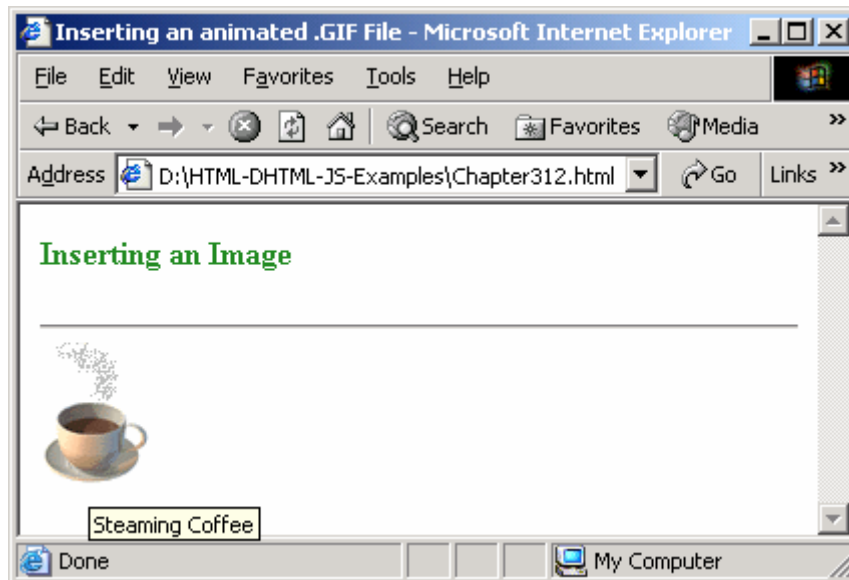
3.4.1 Chèn ảnh động (.GIF) vào tài liệu HTML

Trong chương 3, chúng ta vừa mới thảo luận thế nào là file GIF và chúng được đưa vào một tài liệu HTML bằng cách nào. Nhớ rằng thẻ được dùng để chèn một ảnh vào trong trang web.

Ví dụ 12:

```
<HTML>
<HEAD>
<TITLE>Inserting an animated .GIF File</TITLE>
```

```
</HEAD>
<H1><FONT          SIZE=3          COLOR=FORESTGREEN><B>Inserting          an
Image</B></FONT></H1>
<HR>
<BODY>
    <IMG SRC=coffeecup.gif ALT="Steaming Coffee">
</BODY>
</HTML>
```



Hình 3.13: Kết quả của ví dụ 12

Chú ý: trong ví dụ 12, coffeecup.gif là một ảnh động là file .gif, khi hiển thị trong trình duyệt thì nó hiển thị một luồng khói phía trên cốc cà phê. File này được đặt vào cùng thư mục với file HTML nguồn. Tuy nhiên, nó có thể được thay thế bởi các hình ảnh động khác

3.4.2 Chèn âm thanh vào tài liệu HTML

Tính hấp dẫn của một trang Web tương tác đó là thường sử dụng tiếng “bíp” khi người dùng kích hoạt đến một trang web khác. Một cách tùy chọn, một trang web sẽ yêu cầu kết hợp tiếng nhạc bên trong nó để làm cho trang web đó có tính hấp dẫn hơn. MIDI là một định dạng chuẩn của các file nhạc mà chúng được dùng trong tài liệu HTML. Các file MIDI chứa những nốt nhạc và các loại nhạc cụ cho các bản nhạc. Nhạc cụ điện tử trong card âm thanh mô phỏng tiếng nhạc. Nói cách khác, các file .wav và .au dùng để lưu âm thanh

Để thêm vào âm thanh cho trang web, chúng ta phải sử dụng các file âm thanh (.wav hay .midi) trên hệ thống của chúng ta. Chẳng hạn như,

```
<BGSOUND SRC="path\sound filename" loop="positive number/infinite">
```

Nếu chúng ta không xác định được đường dẫn thì trình duyệt sẽ tìm file mà ở đó trang web đang được định vị. Thuộc tính loop xác định số lần mà âm thanh sẽ được bật lên. Chú ý rằng phần tử BGSOUND không được hỗ trợ bởi Netscape.

Nhạc MIDI như đã đề cập ở phần trước, chỉ là một tiếng nhạc tổng hợp. Tuy nhiên, nếu muốn thêm vào file nhạc của mình, chẳng hạn như giọng nói hay một bài hát đặc biệt khi trang web được chuyển đến, thì chúng ta cần phải sử dụng những file âm thanh đã được số hóa.

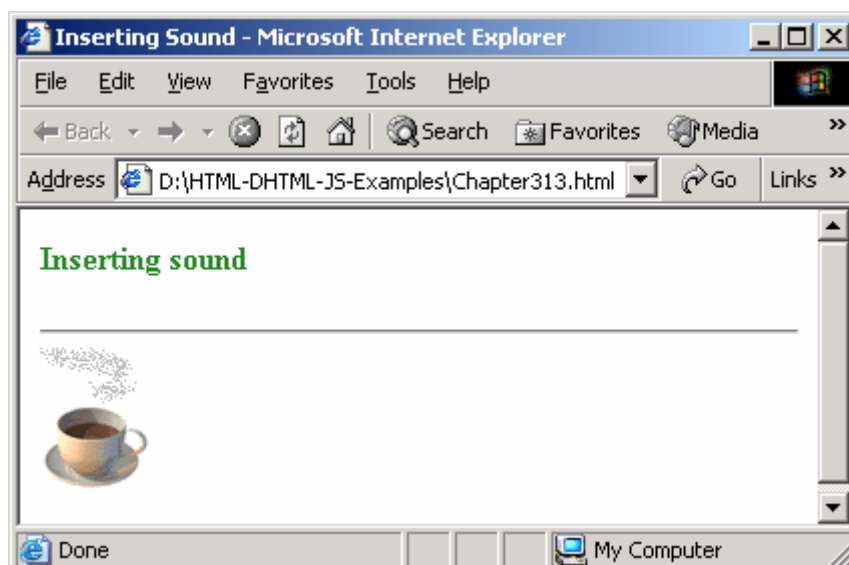
Một file âm thanh được số hóa chứa những thông tin để sao chép lại một bản sao âm thanh đúng như file gốc của nó. Tần số tại những âm thanh được đưa ra làm ví dụ chuẩn xác định được chất lượng của file âm thanh đó, tần suất cao hơn, chất lượng âm thanh tốt hơn. Điểm hạn chế đó là nó cũng sẽ làm gia tăng kích thước của file. Những file âm thanh được số hóa có thể được lưu trong hai định dạng, đó là file .au hay .wav

Định dạng file .wav bắt đầu với hệ điều hành Windows. Nó có tỉ lệ nén thấp và kích thước file lại

rất lớn. Còn định dạng file .au được tổ chức với những máy cài hệ điều hành Unix. Có vẻ sẽ tốt hơn cho việc lưu trữ các file âm thanh được và nó có một tỉ lệ nén cao hơn so với những file .wav

Ví dụ 3:

```
<HTML>
  <HEAD>
    <TITLE>Inserting Sound</TITLE>
  </HEAD>
  <H1><FONT SIZE=3 COLOR=FORESTGREEN><B>Inserting sound</B></FONT></H1>
  <HR>
  <BODY>
    <BGSOUND src="WindowsLogonSound.wav" loop="infinite">
    <IMG SRC=coffeecup.gif ALT="Steaming Coffee">
  </BODY>
</HTML>
```



Hình 3.14: Kết quả của ví dụ 13

Chú ý: Trong ví dụ 13, khi hiển thị trang web trong trình duyệt file nhạc đăng nhập Windows sẽ chạy. File windowLogonSound.wav được đặt trong cùng thư mục với file HTML nguồn. Tuy nhiên, nó có thể được thay thế bởi các file nhạc khác

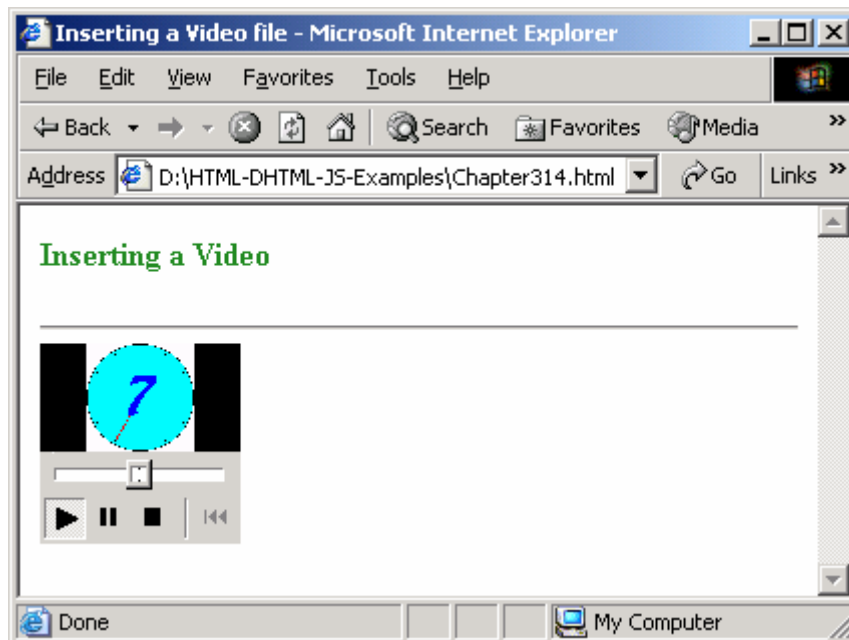
3.4.3 Chèn video vào tài liệu HTML

Một file video có thể có phần mở rộng là: .avi, .asf, .ram hay là .ra. Để chèn một file video vào tài liệu HTML, thẻ <EMBED> có thể được sử dụng. Chẳng hạn như,

```
<EMBED SRC="path\video file name" WIDTH="width in pixels or percentage"
HEIGHT="height in pixels or percentage">
```

Ví dụ 14:

```
<HTML>
  <HEAD>
    <TITLE>Inserting a Video file</TITLE>
  </HEAD>
  <H1><FONT SIZE=3 COLOR=FORESTGREEN><B>Inserting Video</B></FONT></H1>
  <HR>
  <BODY>
    <EMBED SRC=clock.avi HEIGHT=100 WIDTH=100>
  </BODY>
</HTML>
```



Hình 3.15: Kết quả của ví dụ

Chú ý: Để xem kết quả 14, trình Media Player phải được cài đặt. File clock.avi có thể được thay thế bởi các file video khác

CHƯƠNG 4: SỬ DỤNG BIỂU MẪU VÀ KHUNG

Kết thúc chương này, bạn có thể:

- Sử dụng biểu mẫu (form) và các phần tử nhập thông thường trong HTML
- Sử dụng khung (frame)

4.1 GIỚI THIỆU

Một người hay dùng Web thường xuyên gặp các loại biểu mẫu khác nhau và nhiều khi phải điền thông tin vào các loại biểu mẫu đó. Trong chương này, ta sẽ học cách thêm các phần tử nhập vào một tài liệu HTML và tạo ra một biểu mẫu. Ta cũng sẽ học thêm về khung để phân chia tài liệu.

4.2 GIỚI THIỆU BIỂU MẪU

Form HTML là một phần của tài liệu, nó chứa các phần tử đặc biệt gọi là các điều khiển. Các điều khiển được sử dụng để nhập thông tin từ người dùng và cung cấp một số tương tác. Dữ liệu do người dùng nhập vào có thể được xác nhận hợp lệ nhờ các kịch bản phía máy khách (client-side scripts) và được chuyển đến máy chủ để xử lý thêm.

4.2.1 Sử dụng biểu mẫu

Việc sử dụng biểu mẫu trên World Wide Web là khá nhiều và liên tục tăng lên. Sau đây là một số cách sử dụng thông thường:

- ☐ Thu thập tên, địa chỉ, số điện thoại, địa chỉ e-mail và các thông tin khác để người dùng đăng ký cho một dịch vụ hay một sự kiện nào đó
- ☐ Thu thập thông tin dùng để đăng ký mua một mặt hàng nào đó, ví dụ, khi muốn mua một cuốn sách trên Internet, ta phải điền tên, địa chỉ gửi thư, phương thức thanh toán và các thông tin liên quan khác.
- ☐ Thu thập thông tin phản hồi về một web site. Hầu hết các site cung cấp một dịch vụ nào đấy đều khuyến khích khách hàng gửi thông tin phản hồi. Ngoài việc xây dựng mối quan hệ với khách hàng, đây còn là một nguồn thông tin để trao đổi hoặc cải tiến dịch vụ.
- ☐ Cung cấp công cụ tìm kiếm cho web site. Các site cung cấp nhiều thông tin khác nhau thường cung cấp cho người dùng hộp tìm kiếm để cho phép họ tìm kiếm thông tin nhanh hơn.

Một biểu mẫu điển hình trên trang web như sau:

Sample Form - Microsoft Internet Explorer

File Edit View Favorites Tools Help

Back Forward Stop Home Search Favorites Media Links

Address Go

Sample Stock Survey

Describe your investment experience

☐ beginner ☒ intermediate ☐ expert

Types of Investments you make

☒ Individual Stocks ☐ Options ☒ Mutual Funds

How do you buy your stocks?

What is your hot stock pick for?

Any Investment Advice for others?

Done My Computer

Hình 4.1: Một dạng biểu mẫu

4.2.2 Phần tử FORM

Phần tử <FORM> được sử dụng để tạo một vùng trên trang như một biểu mẫu. Nó chỉ ra cách bố trí của biểu mẫu. Các thuộc tính bao gồm:

Thuộc tính	Mô tả
ACCEPT	Thuộc tính này xác định danh sách các kiểu MIME được máy chủ nhận ra, trong đó có chứa kịch bản (script) để xử lý biểu mẫu. Cú pháp là ACCEPT = "Internet media type"
ACTION	Thuộc tính này xác định vị trí của script sẽ xử lý biểu mẫu hoàn chỉnh và đã được gửi đi Cú pháp là: ACTION = "URL"
METHOD	Thuộc tính này xác định phương thức dữ liệu được gửi đến máy chủ. Nó cũng xác định giao thức được sử dụng khi máy khách gửi dữ liệu lên cho máy chủ. Nếu giá trị là GET thì trình duyệt sẽ tạo một câu hỏi

	có chứa địa chỉ URL của trang , một dấu chấm hỏi và các giá trị do biểu mẫu tạo ra. Trình duyệt sẽ trả lại câu hỏi cho kịch bản được xác định trong URL để xử lý. Nếu giá trị là POST, thì dữ liệu trên biểu mẫu được gửi đến kịch bản xử lý như một khối dữ liệu. Người ta không sử dụng chuỗi câu hỏi. Cú pháp là METHOD = (GET POST).
--	--

Ví dụ, để đưa một biểu mẫu đến chương trình “xử lý biểu mẫu” sử dụng theo phương thức POST

```
<FORM action="http://mysite.com/processform" METHOD="post">
...form contents...
</FORM>
```

4.2.3 Các phần tử nhập của HTML

Khi tạo một biểu mẫu, ta có thể đặt các điều khiển lên biểu mẫu để nhận dữ liệu nhập vào từ người dùng. Các điều khiển này được sử dụng với phần tử <FORM>. Tuy nhiên, ta cũng có thể sử dụng chúng ở bên ngoài biểu mẫu để tạo các giao diện người dùng.

☐ Phần tử INPUT

Phần tử <INPUT> xác định loại và sự xuất hiện của điều khiển trên biểu mẫu. Các thuộc tính của phần tử này là:

Attributes	Description
TYPE	Thuộc tính này xác định loại phần tử. Ta có thể chọn một trong các lựa chọn: TEXT, PASSWORD, CHECKBOX, RADIO, SUBMIT, RESET, FILE, HIDDEN và BUTTON. Mặc định là TEXT
NAME	Thuộc tính này chỉ tên của điều khiển. Ví dụ nếu có nhiều hộp văn bản (text box) trên một biểu mẫu thì bạn nên sử dụng tên để xác định chúng – TEXT1, TEXT2 hoặc bất kỳ tên nào mình chọn. Phạm vi hoạt động của thuộc tính NAME nằm trong phần tử FORM
VALUE	Đây là một thuộc tính tùy chọn, nó xác định giá trị khởi tạo của điều khiển. Tuy nhiên, đối với kiểu (TYPE) là RADIO thì ta phải xác định cho nó một giá trị.
SIZE	Thuộc tính này xác định độ rộng ban đầu của điều khiển. Đối với kiểu là TEXT hay PASSWORD thì kích thước được xác định theo ký tự. Đối với các loại phần tử nhập khác, độ rộng được xác định bằng điểm (pixels)
MAXLENGTH	Thuộc tính này được sử dụng để xác định số ký tự lớn nhất có thể nhập vào phần tử TEXT hoặc PASSWORD. Mặc định là không giới hạn.
CHECKED	Đây là thuộc tính logic để xác định nút có được chọn hay không. Thuộc tính này được sử dụng khi kiểu nhập là RADIO hay CHECKBOX
SRC	SRC = “URL”. Thuộc tính này được dùng khi ta muốn sử dụng một ảnh trong kiểu INPUT. Nó xác định vị trí của ảnh

Phần này ta sẽ thảo luận về các loại phần tử nhập cùng với một số đặc tính và sự kiện thường dùng.

☐ **Nút bấm**

Loại này tạo ra một điều khiển nút (button)

Tên	Mô tả
NAME	Thiết lập hoặc truy xuất tên của điều khiển
SIZE	Thiết lập hoặc truy xuất kích thước của điều khiển
TYPE	Truy xuất loại điều khiển bên trong được biểu diễn bởi <INPUT type = button>
VALUE	Thiết lập hoặc truy xuất giá trị của nút <INPUT type=button>.

Ví dụ,

```
<INPUT TYPE=button VALUE="click" NAME="b1">
```

☐ **Văn bản**

Loại này tạo một điều khiển nhập văn bản có một dòng. Thuộc tính SIZE xác định số ký tự có thể hiển thị trong phần tử. Thuộc tính MAXLENGTH xác định số ký tự tối đa có thể nhập vào phần tử này.

Ví dụ,

```
<INPUT TYPE=text VALUE="" NAME="textbox" SIZE=20>
```

Giá trị Value ở đây để hiển thị nội dung ban đầu của văn bản và để truy xuất văn bản khi biểu mẫu được gửi đi

☐ **Checkbox (Hộp kiểm)**

Loại này tạo ra một điều khiển checkbox. Người dùng có thể chọn một hoặc nhiều checkbox. Khi một phần tử checkbox được chọn, thì cặp tên/giá trị được nhận cùng với biểu mẫu. Giá trị mặc định của checkbox là bật (on) . Phần tử checkbox là một phần tử trên dòng và không cần thẻ đóng.

Tên	Mô tả
CHECKED	Thiết lập hoặc truy xuất trạng thái của checkbox
NAME	Thiết lập hoặc truy xuất tên của điều khiển
SIZE	Thiết lập hoặc truy xuất kích thước của điều khiển
STATUS	Thiết lập hoặc truy xuất trạng thái xem checkbox có được chọn hay không
TYPE	Truy xuất loại điều khiển, sử dụng <INPUT type = checkbox>.
VALUE	Thiết lập hoặc truy xuất giá trị của checkbox <INPUT

	type=checkbox>
--	----------------

☐ **Radio**

Loại này tạo ra điều khiển kiểu nút radio. Một điều khiển kiểu nút radio (radio button control) được sử dụng đối với các tập giá trị loại trừ lẫn nhau. Các điều khiển radio trong một nhóm phải có *cùng tên*. Vào một thời điểm, người dùng chỉ có thể chọn một lựa chọn. Chỉ có nút radio được chọn trong nhóm mới tạo tên cặp giá trị tên/value trong dữ liệu được nhận. Các nút radio nên đặt thuộc tính **giá trị**.

Tên	Mô tả
CHECKED	Thiết lập hoặc truy xuất trạng thái của nút radio
NAME	Thiết lập hoặc truy xuất tên của điều khiển
SIZE	Thiết lập hoặc truy xuất kích thước của điều khiển
STATUS	Thiết lập hoặc truy xuất trạng thái xem nút radio có được chọn hay không
TYPE	Truy xuất loại điều khiển, sử dụng <INPUT type = radio>.
VALUE	Thiết lập hoặc truy xuất giá trị của radio <INPUT type=radio>

Ví dụ,

```
<INPUT TYPE=radio NAME="sex" VALUE="male">Male
```

☐ **Submit**

Loại này tạo ra một nút submit. Khi người dùng nhấp vào nút Submit, biểu mẫu được chuyển đến vị trí được xác định trong thuộc tính ACTION. Cặp tên/giá trị của nút submit được nhận cùng với biểu mẫu.

Ví dụ,

```
<INPUT TYPE=submit VALUE="click" NAME="b1">
```

☐ **Ảnh**

Điều khiển này tạo ra một nút submit dạng ảnh. Giá trị của thuộc tính SRC xác định URL của ảnh được đặt trong nút ấy. Khi người dùng nhấp vào điều khiển ảnh này, biểu mẫu được chuyển đi để xử lý. Tọa độ x và y (được đo bằng điểm) tại vị trí nhấp chuột được chuyển đến máy chủ với định dạng sau

```
Name.x = valueofx
```

```
Name.y = valueofy
```

Trong đó, 'name' là tên của điều khiển

Ta có thể sử dụng nhiều nút submit với các hình ảnh khác nhau thay vì một nút submit chỉ có một hình. Nếu cần trình bày nhiều ảnh ta có thể sử dụng bản đồ ảnh.

Ví dụ,

```
<INPUT TYPE =image SRC="usamap.gif" NAME="name">
```

☐ **Reset**

Điều khiển này tạo ra nút reset. Khi người dùng nhấp vào nút này, các giá trị của tất cả các điều khiển được tái thiết lập trở về giá trị ban đầu, được xác định trong các giá trị thuộc tính của chúng

Ví dụ,

```
<INPUT TYPE=reset VALUE="Reset" NAME="B2">
```

Ví dụ sau thể hiện việc sử dụng nhiều kiểu nhập khác nhau.

Ví dụ 1:

```
<HTML>
  <HEAD>
    <TITLE> Sample Form </TITLE>
  </HEAD>
  <BODY bgcolor="#ffffcc" Text="#000099">
    <FORM ACTION="http://www.mysite.com/FormSite" METHOD="POST">
      <B><H2 align="left">Sample Stock Survey</H2></B>
      <p><B>Describe your investment experience</B></p>
      <p>
        <INPUT TYPE="RADIO" NAME="RESULT_RadioButton-3"
          VALUE="Radio-0">beginner
        <INPUT TYPE="RADIO" NAME="RESULT_RadioButton-3"
          VALUE="Radio-1">intermediate
        <INPUT TYPE="RADIO" NAME="RESULT_RadioButton-3"
          VALUE="Radio-2">expert
      </p>
      <p><B>Types of Investments you make</B></P>
      <P>
        <INPUT TYPE="CHECKBOX" NAME="RESULT_CheckBox-4"
          VALUE="CheckBox-0">Individual Stocks
        <INPUT TYPE="CHECKBOX" NAME="RESULT_CheckBox-4"
          VALUE="CheckBox-1">Options
        <INPUT TYPE="CHECKBOX" NAME="RESULT_CheckBox-4"
          VALUE="CheckBox-2">Mutual Funds<BR></p>
      <p><B>What is your stock pick for this year?</B></P>
      <P>
        <INPUT TYPE="TEXT" NAME="RESULT_TextField-6"
          SIZE="30" MAXLENGTH="30">
      </p>
    </p>
```

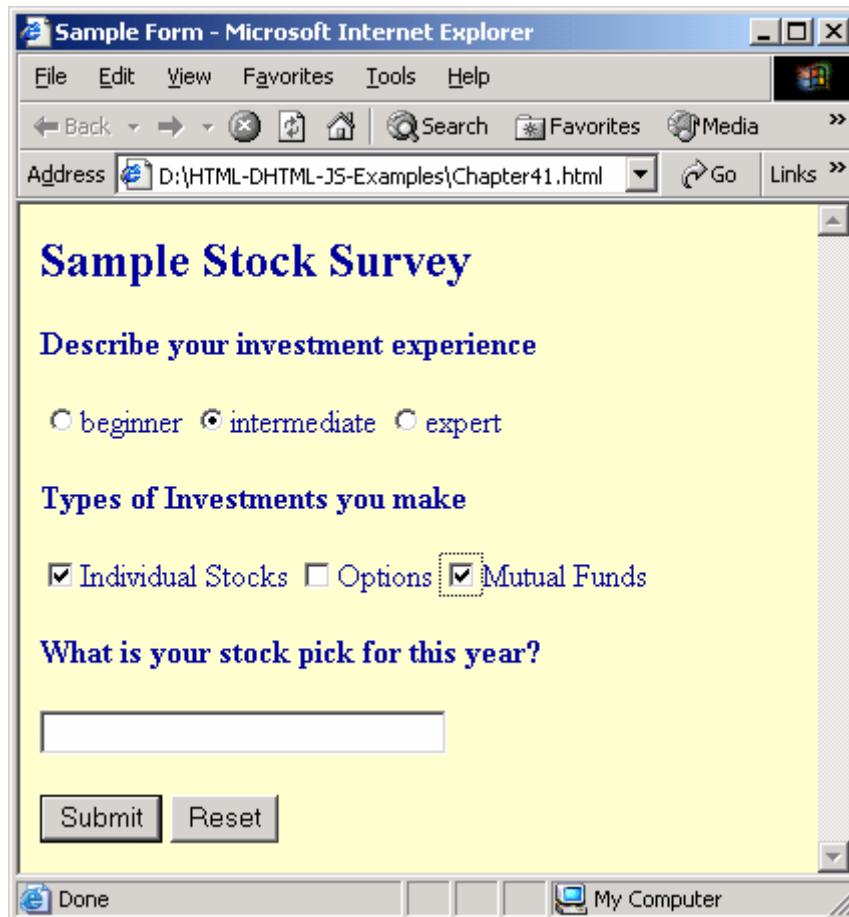
```
<INPUT TYPE="SUBMIT" NAME="Submit"
      VALUE="Submit">

<INPUT TYPE="RESET" NAME="Reset" VALUE="Reset"></p>

</FORM>

</BODY>

</HTML>
```



Hình 4.2: Kết quả ví dụ 1

□ Phần tử TextArea (vùng văn bản)

Loại này tạo ra một điều khiển nhập văn bản trên nhiều dòng so với hộp văn bản nhập một dòng. Ta phải xác định kích thước của textarea. Ta cũng phải xác định số dòng, số cột trong textarea. Tuy nhiên, ta phải kết thúc phần tử với thẻ đóng `</TEXTAREA>`

Tên	Mô tả
COLS	Truy xuất độ rộng của textarea
ROWS	Thiết lập hoặc truy xuất số hàng ngang trong <code><TEXTAREA></code>
SIZE	Thiết lập hoặc truy xuất kích thước của điều khiển
TYPE	Truy xuất loại điều khiển, sử dụng <code><TEXTAREA></code>

VALUE	Thiết lập hoặc truy xuất giá trị của <TEXTAREA>
-------	---

Ví dụ,

```
<TEXTAREA NAME="text1" COLS=20 ROWS=5> </TEXTAREA>
```

□ Phần tử **BUTTON** (Nút bấm)

Điều khiển này tạo ra điều khiển button. Khi người dùng nhấp vào nút Submit, biểu mẫu được nhận để xử lý. Cặp tên/giá trị của nút submit được nhận cùng với biểu mẫu.

Thuộc tính	Mô tả
NAME	Gán tên cho nút
VALUE	Gán giá trị cho nút
TYPE	Xác định loại nút. Các giá trị có thể là: Submit – tạo nút nhận biểu mẫu khi được nhấp vào Button – Tạo nút kích hoạt một script khi được kích vào Reset – tạo nút thiết lập lại (reset) biểu mẫu và các giá trị của các điều khiển trong biểu mẫu

Một nút (BUTTON) có loại là submit (type=submit) giống như một phần tử INPUT của loại nút. Sự khác nhau là ở chỗ khi phần tử BUTTON được nhấp vào thì cặp tên/giá trị được nhận cùng biểu mẫu. Một nút (BUTTON) có loại là submit cũng chứa một ảnh và giống một phần tử INPUT có loại là ảnh. Sự khác nhau là ở chỗ phần tử INPUT có dạng một ảnh “phẳng” trong khi phần tử BUTTON thì hiển thị như một nút có hiệu ứng lên / xuống khi nhấp vào.

Ví dụ sau minh họa cho việc sử dụng phần tử TEXTAREA và BUTTON

Ví dụ 2:

```
<HTML>
```

```
<HEAD>
```

```
<TITLE> Sample Form </TITLE>
```

```
</HEAD>
```

```
<BODY bgColor="#ffffcc" Text="#000099">
```

```
<FORM ACTION="http://www.mysite.com/FormSite" METHOD="POST">
```

```
<B><CENTER><H2>Sample Stock Survey</H2></CENTER></B>
```

```
<P><B>Any Investment Advice for others?</B></P>
```

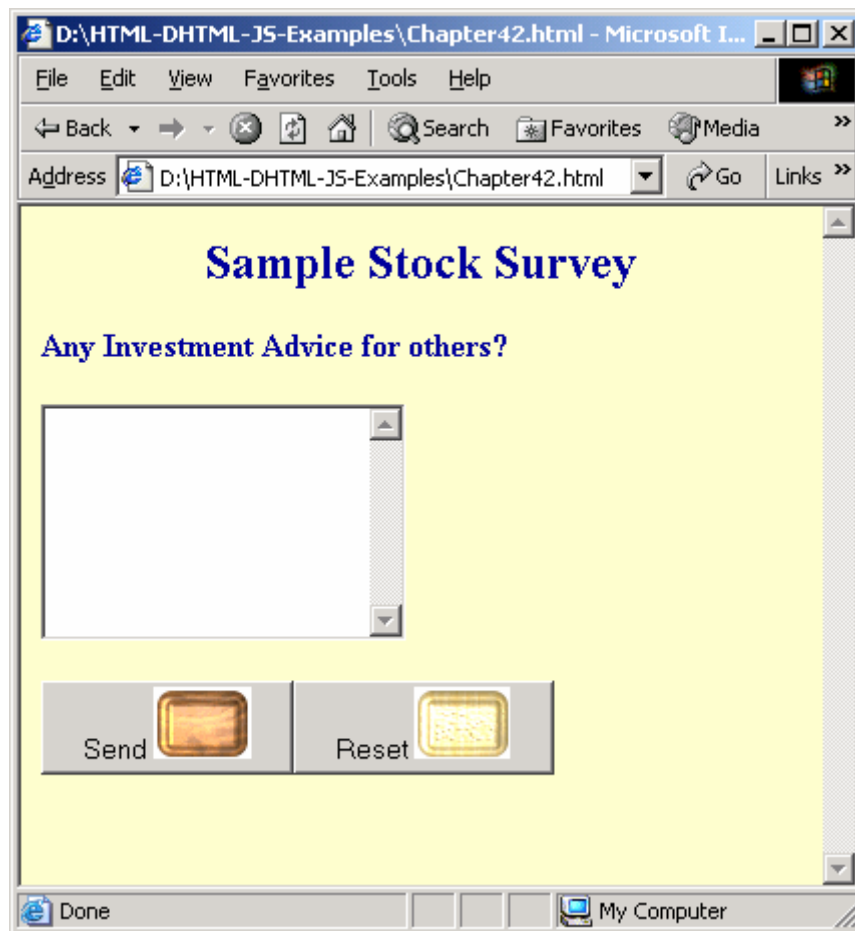
```
<TEXTAREA NAME="RESULT_TextArea-8" ROWS="7" COLS="20">
```

```
</TEXTAREA>
```

```
<P>
```

```
<BUTTON type="submit" name="submit" value="submit">
```

```
Send <IMG src="submit.gif" alt="Submit"> </BUTTON>
<BUTTON type="reset" name="reset" >
Reset <IMG src="reset.gif" alt="Reset"> </BUTTON>
</P>
</FORM>
</BODY>
</HTML>
```



Hình 4.3: Kết quả ví dụ 2

□ Phần tử lựa chọn (Select)

Phần tử SELECT được sử dụng để hiển thị một danh sách các lựa chọn cho người dùng. Mỗi lựa chọn được biểu diễn bởi phần tử OPTION. Một phần tử SELECT phải chứa ít nhất một phần tử OPTION. Thành phần được chọn lựa sẽ hiển thị với màu khác so với các thành phần còn lại.

Thuộc tính	Mô tả
NAME	Gán tên cho phần tử. Khi biểu mẫu được gửi đi, thuộc tính tên được gán với giá trị chọn lựa
SIZE	Nếu có nhiều sự chọn lựa, người dùng sử dụng cuộn, thuộc tính này xác định số dòng trong danh sách được hiển thị.

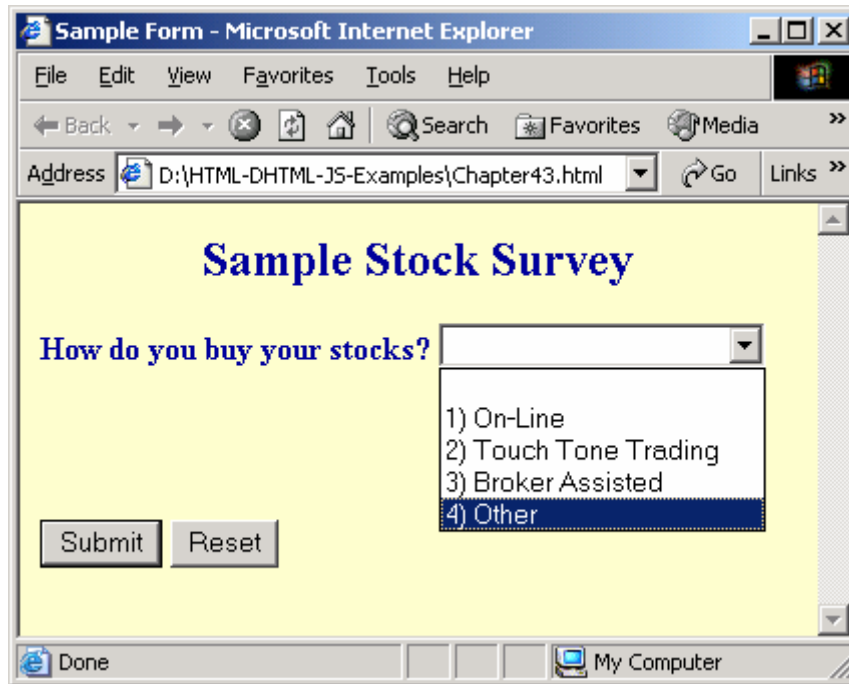
MULTIPLE	Là thuộc tính logic cho phép người dùng chọn một hoặc nhiều chọn lựa.
----------	---

Mỗi lựa chọn trong hộp chọn được lấy giá trị thông qua mô tả văn bản của nó, xem nó có được chọn hay không.

Mảng Option được tạo ra theo danh sách lựa chọn trong phần tử SELECT. Mỗi lựa chọn có thuộc tính Text và Selected cho phép chúng ta kiểm tra tùy chọn đó có được chọn hay không và truy xuất văn bản của lựa chọn theo thứ tự. Bây giờ ta có thể kiểm tra mỗi phần tử trong mảng và xác nhận nó.

Ví dụ 3:

```
<HTML>
  <HEAD>
    <TITLE> Sample Form </TITLE>
  </HEAD>
  <BODY bgcolor="#ffffcc" Text="#000099">
    <FORM ACTION="http://www.mysite.com/FormSite" METHOD="POST">
      <B><CENTER><H2>Sample Stock Survey </H2></CENTER></B>
      <P><B>How do you buy your stocks?</B>
      <SELECT NAME="RESULT_RadioButton-5">
        <OPTION></OPTION>
        <OPTION>1) On-Line</OPTION>
        <OPTION>2) Touch Tone Trading</OPTION>
        <OPTION>3) Broker Assisted</OPTION>
        <OPTION>4) Other</OPTION>
      </SELECT></P>
      <BR><BR><BR>
      <P><INPUT TYPE="SUBMIT" NAME="Submit" VALUE="Submit">
        <INPUT TYPE="RESET" NAME="Reset" VALUE="Reset">
    </P>
  </FORM>
</BODY>
</HTML>
```



Hình 4.4: Kết quả ví dụ 3

Phần tử OPTGROUP được sử dụng để nhóm các lựa chọn vào một cây phân cấp. Ví dụ, bảng nội dung có thể có tên các chương. Các chủ đề và chủ đề con trong một chương có thể được nhóm vào chương đó.

Thuộc tính	Mô tả
SELECTED	Đây là thuộc tính logic sử dụng để chọn trước một tùy chọn.
VALUE	Xác định giá trị được nhận vào cho tùy chọn được chọn. Giá trị này được gán với tên của phần tử SELECT. Nội dung của phần tử OPTION là giá trị mặc định
LABEL	Xác định văn bản hiển thị cho một tùy chọn.

Ví dụ 4:

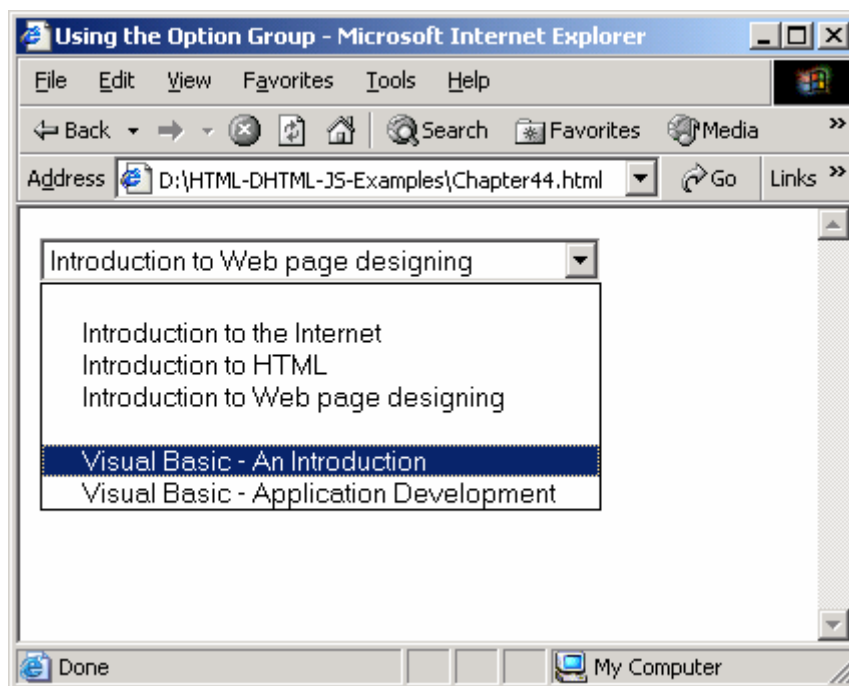
```
<HTML>

<Head>
<Title>Using the Option Group</Title>
</head>
<BODY>

<FORM action = "http://www.mysite.com/FormSite"
method = "post">
<P><SELECT name="course">
<OPTGROUP>
<OPTION value = "Internetintro">Introduction to the
Internet
<OPTION value="Introhtml">Introduction to HTML
```



```
<OPTION value="Introweb">Introduction to Web page  
designing  
</OPTGROUP>  
<OPTGROUP>  
    <OPTION value="vbintro">Visual Basic-An Introduction  
    <OPTION value="vbdev">Visual Basic - Application  
    Development  
</OPTGROUP>  
</SELECT>  
</FORM>  
</BODY>  
</html>
```



Hình 4.5: Kết quả ví dụ 4

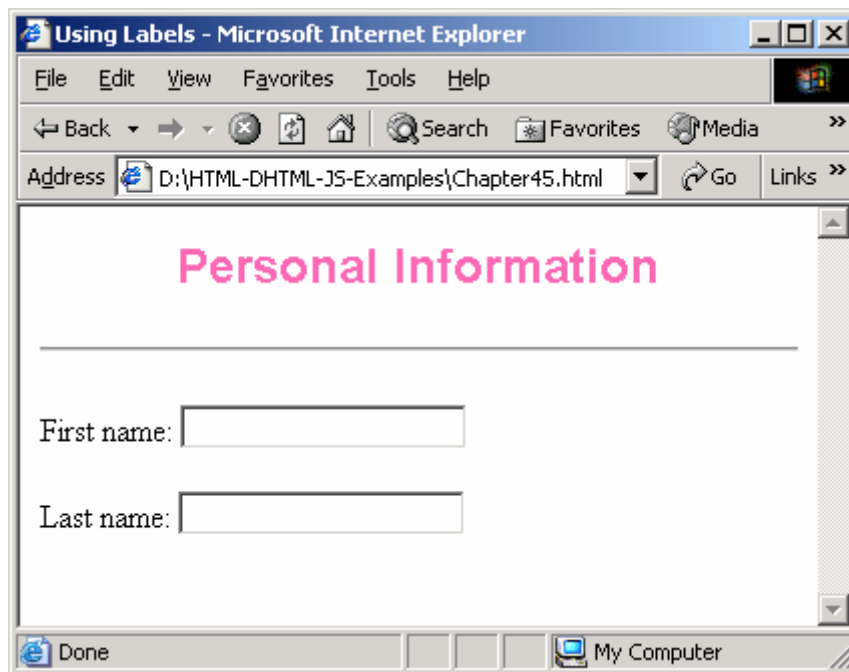
□ Phần tử LABEL (Nhãn)

Phần tử LABEL được sử dụng để gắn thông tin vào các phần tử điều khiển. Ví dụ, phần tử TEXT không có nhãn để xác định rõ nó. Ta có thể gắn nhãn vào phần tử TEXT khi trang hiển thị. Ta phải xác định thuộc tính ID của điều khiển mà nó được gắn vào.

Ví dụ 5:

```
<HTML>  
    <HEAD>  
        <TITLE>Using Labels</TITLE>  
    </HEAD>  
    <BODY>
```

```
<H2><CENTER><FONT size = 5 color = hotpink face = arial>Personal
Information</FONT></CENTER></H2>
<HR align = center>
<FORM action="http://somesite.com" method="post">
<P>
    <LABEL for="firstname">First name: </LABEL>
    <INPUT type="text" id="firstname"><BR><BR>
    <LABEL for="lastname">Last name: </LABEL>
    <INPUT type="text" id="lastname"><BR><BR>
</P>
</FORM>
</BODY>
</HTML>
```



Hình 4.6: Kết quả ví dụ 5

4.2.4 Tạo biểu mẫu

Chúng ta đã thảo luận về phần tử `<FORM>` và các điều khiển có thể thêm vào biểu mẫu để nhận thông tin người dùng. Trong phần này, chúng ta sẽ tạo một biểu mẫu để nhận thông tin về một người xin việc. Các nút RESET và SUBMIT thực hiện các công việc cần thiết.

Ví dụ 6:

```
<HTML>
  <HEAD>
    <TITLE>Job application</TITLE>
  </HEAD>
```

```
<BODY>

  <H1><CENTER><FONT SIZE = 4 COLOR = Forestgreen> Application
  Form</CENTER></FONT></H1>

  <HR><BR>

  <FORM action="http://somesite.com/processform" method="post">

    <P>

      <LABEL for="firstname">Name: </LABEL>

      <INPUT type="text" id="firstname"><BR>

      <P>Area of Interest

      <BR><BR>

      <INPUT TYPE=RADIO NAME="CONTROL1" VALUE="0"
CHECKED>Web Designer

      <INPUT TYPE=RADIO NAME="CONTROL1" VALUE="1" >Web
      Administrator

      <INPUT TYPE=RADIO NAME="CONTROL1" VALUE="2" >Web
      Developer

      <P>Experience

      <SELECT NAME="CONTROL2">

        <OPTION>None</OPTION>

        <OPTION>1 Year</OPTION>

        <OPTION>3 Years</OPTION>

        <OPTION>5 Years</OPTION>

      </SELECT>

      <BR>

      <P>Comments

      <BR>

      <TEXTAREA NAME="CONTROL3" COLS="30" ROWS="5">Type
      your comments here.</TEXTAREA>

      <BR>

      <P><INPUT NAME="CONTROL4" TYPE=CHECKBOX CHECKED>Send
      acknowledgement

      <BR>

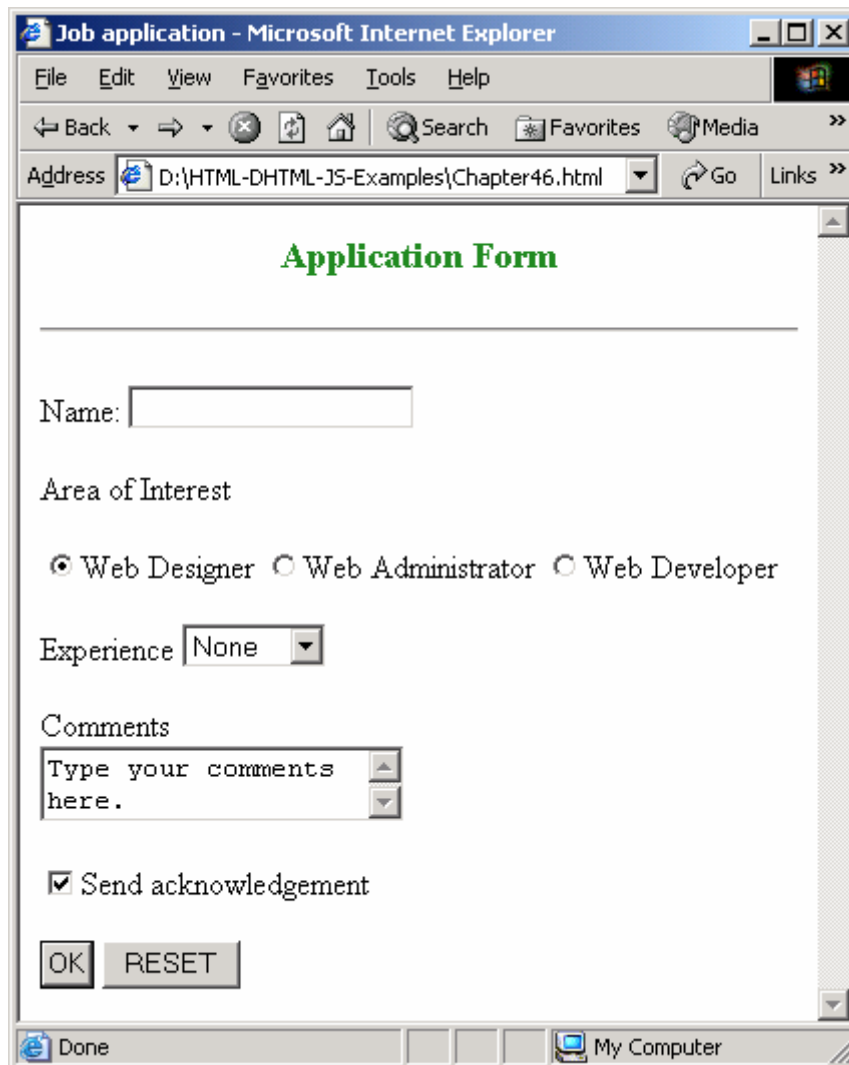
      <P><INPUT TYPE=SUBMIT VALUE=OK>

      <INPUT TYPE=RESET VALUE=RESET>

    </FORM>

  </BODY>

</HTML>
```



The screenshot shows a Microsoft Internet Explorer window titled "Job application - Microsoft Internet Explorer". The address bar displays "D:\HTML-DHTML-JS-Examples\Chapter46.html". The main content area contains an "Application Form" with the following elements:

- Name:** A text input field.
- Area of Interest:** Three radio buttons labeled "Web Designer", "Web Administrator", and "Web Developer".
- Experience:** A dropdown menu currently showing "None".
- Comments:** A text area with the placeholder text "Type your comments here." and vertical scrollbars.
- Send acknowledgement:** A checked checkbox.
- Buttons:** "OK" and "RESET" buttons.

The status bar at the bottom shows "Done" and "My Computer".

Hình 4.7: Kết quả ví dụ 6

Khi có nhiều phần tử trong một form, chúng ta cần phải điều khiển chúng. Sau đây là các thuộc tính để điều khiển các phần tử

□ **Thiết lập tiêu điểm (Focus)**

Một phần tử trở thành hoạt động khi nó nhận tiêu điểm. Ví dụ, để nhập văn bản vào phần tử TEXT, tiêu điểm phải nằm trên phần tử đó. Khi phần tử mất tiêu điểm, nó sẽ không hoạt động nữa. Cách đơn giản nhất để đặt tiêu điểm cho phần tử là ta kích vào nó bằng cách sử dụng chuột, joystick ... hoặc dùng bàn phím để đặt.

□ **Thứ tự tab**

Thuộc tính tabindex của một phần tử xác định trình tự phần tử nhận tiêu điểm thông qua bàn phím. Ở đây bao gồm các phần tử được lồng vào các phần tử khác. Giá trị có thể là bất cứ số nào giữa 0 và 32767. Tiêu điểm bắt đầu từ phần tử có giá trị tabindex thấp nhất. Nếu ta gán cùng một giá trị tabindex cho hơn một phần tử, thì các phần tử nhận tiêu điểm theo thứ tự nó xuất hiện trong tài liệu.

Nếu phần tử nào không hỗ trợ thuộc tính `tabindex`, nó sẽ là phần tử nhận tiêu điểm cuối cùng. Nếu ta vô hiệu hóa một phần tử, nó sẽ không được liệt kê vào thứ tự tab – và nó sẽ không nhận được tiêu điểm.

Ví dụ,

```
<INPUT tabindex=2 TYPE=RADIO NAME="CONTROL1" VALUE="0" CHECKED>Web Designer  
<INPUT tabindex = 6 TYPE=SUBMIT VALUE=OK>
```

☐ **Phím truy cập (Access Keys)**

Thuộc tính này được sử dụng để gán phím truy cập cho phần tử. Phím truy cập là một ký tự và thường được sử dụng cùng với phím ALT. Khi người dùng nhấn phím truy cập, phần tử được xác định sẽ nhận tiêu điểm và bắt đầu hoạt động.

Ví dụ,

```
<LABEL for="firstname">Name: </LABEL>  
<INPUT accesskey="N" tabindex=1 type="text" id="firstname">  
<TEXTAREA accesskey="C" tabindex=4 NAME="CONTROL3" COLS="30" ROWS="5">Type your comments here</TEXTAREA>
```

☐ **Phần tử vô hiệu hóa (Disabling Elements)**

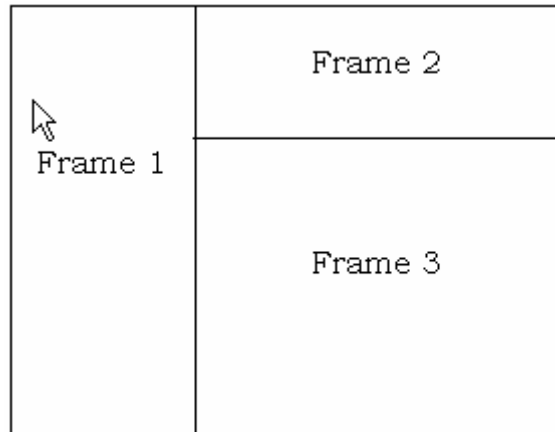
Nếu làm việc với trình soạn thảo văn bản, ta sẽ thấy rằng nếu không mở tài liệu nào thì các tùy chọn lưu và định dạng sẽ bị vô hiệu hóa. Đối với trang Web, ta có thể vô hiệu hóa các phần tử hoặc để ở trạng thái chỉ đọc (read-only) nếu không muốn người dùng truy cập chúng. Ví dụ, khi hiển thị một biểu mẫu, ta có thể vô hiệu hóa nút “Submit” cho đến khi người dùng nhập dữ liệu vào. Thuộc tính vô hiệu hóa được sử dụng để điều khiển truy cập một phần tử. Khi một phần tử bị vô hiệu hóa, nó không được liệt kê trong thứ tự tab. Do vậy, điều khiển không nhận được tiêu điểm và cuối cùng là các giá trị của điều khiển bị vô hiệu hóa không được chuyển đi cùng với biểu mẫu. Một điều khiển bị vô hiệu hóa có thể được kích hoạt nhờ các script lúc thực hiện.

Ví dụ,

```
<INPUT TYPE=SUBMIT VALUE=OK DISABLED=True>
```

4.3 KHUNG (FRAME)

Khung chia một cửa sổ trình duyệt thành nhiều vùng riêng biệt, mà mỗi vùng có thể hiển thị một trang riêng biệt có thể cuộn được. Mỗi khung là một cửa sổ trong cửa sổ chính. Ví dụ, ta có thể sử dụng ba khung trong trang Web, một làm biểu ngữ (banner), một làm menu điều hướng và một để hiển thị dữ liệu. Mỗi khung có thể được tạo, thay đổi và cuộn độc lập nhau.



Hình 4.8: Khung trong trang Web

4.3.1 Tại sao sử dụng khung?

Một trang có thể có một hoặc nhiều khung. Một số lý do sử dụng khung:

- ☐ Hiển thị một biểu tượng (logo) hoặc thông tin tĩnh trên một vị trí cố định trên trang Web
- ☐ Đối với bảng nội dung trong trang mà ở đó người dùng có thể kích vào và di chuyển quanh web site mà không cần phải liên tục quay lại trang nội dung.
- ☐ Nhiều cách hiển thị cho phép người thiết kế giữ một số thông tin tĩnh nào đó trong khi cuộn hay thao tác đối với những nội dung khác trên trang Web

Tuy nhiên, mặt hạn chế của việc sử dụng khung trong trang Web là: không phải tất cả các trình duyệt đều hỗ trợ khung, ví dụ, IE phiên bản 2.0 hoặc trước đó và Netscape 1.2 hoặc trước đó. Để nội dung vẫn hợp lệ mà người sử dụng không cần quan tâm đến trình duyệt có hỗ trợ khung hay không, người thiết kế cần phải cung cấp một cách khác để truy cập vào nội dung.

4.3.2 Sử dụng khung

Một tài liệu HTML chuẩn có phần HEAD và BODY. Một tài liệu HTML sử dụng khung thì có phần HEAD và phần FRAMESET. Phần FRAMESET xác định cách trình bày trong cửa sổ người dùng. Ta không thể sử dụng phần tử BODY và FRAMESET cùng với nhau. Trình duyệt chỉ nhận phần tử đầu tiên xuất hiện trong tài liệu và bỏ qua phần tử sau. Nghĩa là, nếu bạn sử dụng phần tử BODY, thì phần tử FRAMESET sau đó sẽ bị bỏ qua và ngược lại.

Khung được tạo ra bằng cách sử dụng phần tử FRAMESET. Các thuộc tính như sau:

Thuộc tính	Mô tả
ROWS	Xác định độ rộng của khung, được tính theo điểm(pixels), phần trăm hoặc độ rộng tương đối. Giá trị mặc định là 100%, nghĩa là một dòng
COLS	Xác định độ cao của khung, được tính theo điểm(pixels), phần trăm hoặc độ cao tương đối. Giá trị mặc định là 100%. Nó xác định chỉ có một cột

Phần tử FRAME xác định hình thức và nội dung của một khung trong FRAMESET. Ví dụ sau đây tạo hai khung bằng nhau, chia đôi cửa sổ.

Ví dụ 7:

```
<HTML>

  <HEAD>

    <TITLE>Two Equal Frames</TITLE>

  </HEAD>

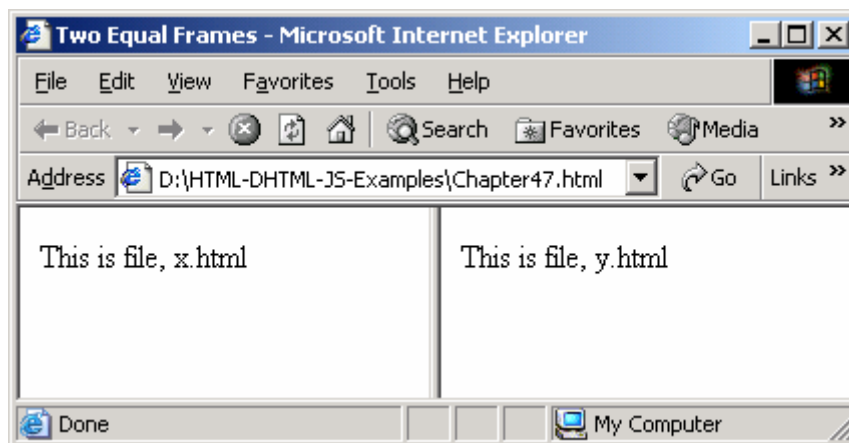
  <FRAMESET COLS="50%,*">

    <FRAME SRC=x.htm>

    <FRAME SRC=y.htm>

  </FRAMESET>

</HTML>
```



Hình 4.9: Kết quả ví dụ 7

Chú ý: File x.html và y.html được lưu cùng thư mục với file .html chính ý

Các thuộc tính của FRAME bao gồm:

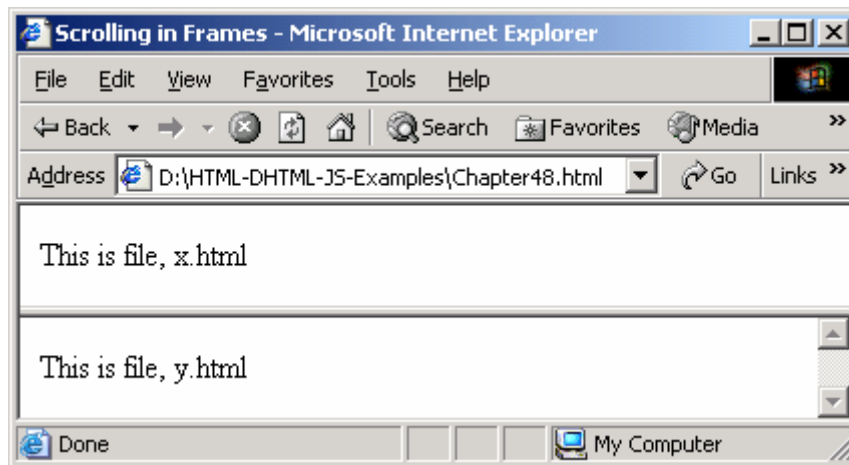
Thuộc tính	Mô tả
NAME	Thuộc tính này gán tên cho khung hiện thời
SRC	Thuộc tính này xác định vị trí tài liệu ban đầu được chứa trong khung
NORESIZE	Đây là thuộc tính logic. Nó quy định cửa sổ khung không được thay đổi kích thước

SCROLLING	Thuộc tính này xác định kiểu cuộn cho cửa sổ khung. Các giá trị có thể là: Auto - Xuất hiện thanh cuộn khi cần thiết. Đây là giá trị mặc định Yes - Luôn luôn xuất hiện thanh cuộn trong cửa sổ của khung No - Không xuất hiện thanh cuộn trong cửa sổ của khung
FRAMEBORDER	Xác định viền của khung. Các giá trị có thể là: 1 - Là giá trị mặc định. Có sự phân cách giữa khung hiện thời với các khung xung quanh 0 - Không có sự phân cách giữa khung hiện thời với các khung lân cận. Tuy nhiên, nếu các khung lân cận có thiết lập thì vẫn xuất hiện sự phân cách này.
MARGINWIDTH	Xác định khoảng cách giữa nội dung trong khung với lề trái và lề phải của khung. Giá trị phải lớn hơn một
MARGINHEIGHT	Xác định khoảng cách giữa nội dung trong khung với lề trên và lề dưới của khung. Giá trị phải lớn hơn một

Ta không thể đóng cửa sổ của khung. Khung được đóng khi cửa sổ tạo ra nó bị đóng lại. Khung không có thanh trạng thái vì vậy ta phải sử dụng thanh trạng thái của khung chính trong tài liệu.

Ví dụ 8:

```
<HTML>
  <HEAD>
    <TITLE>Scrolling in Frames</TITLE>
  </HEAD>
  <FRAMESET ROWS="50%,*">
    <FRAME SRC="x.html" SCROLLING=no>
    <FRAME SRC="y.html" SCROLLING=yes>
  </FRAMESET>
</HTML>
```

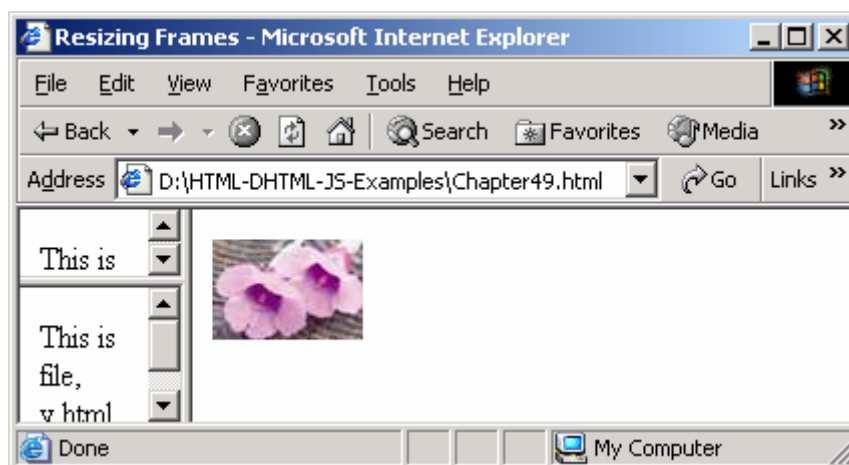
Hình 4.10: Kết quả ví dụ 8

Ví dụ 9:

```
<HTML>

<HEAD>
  <TITLE>Resizing Frames</TITLE>
</HEAD>
<FRAMESET cols="20%, 80%">
  <FRAMESET rows="100, 200">
    <FRAME src="x.html" noresize>
    <FRAME src="y.html">
  </FRAMESET>
  <FRAME src="flowers.jpg">
</FRAMESET>

</HTML>
```

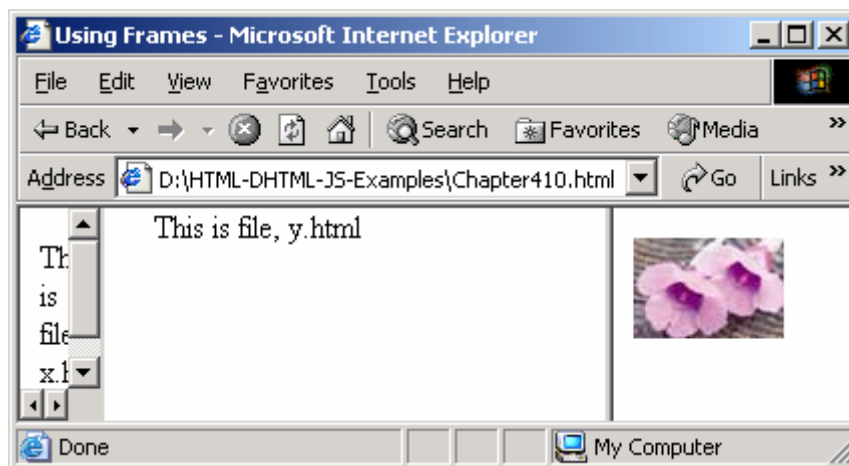


Hình 4.11: Kết quả ví dụ 9

Đoạn mã sau tạo 3 cột: Cột 2 có độ rộng là 250 pixel, cột 1 chiếm 25% khoảng còn lại và cột 3 chiếm 75% khoảng còn lại

Ví dụ 10:

```
<HTML>
  <HEAD>
    <TITLE>Using Frames</TITLE>
  </HEAD>
  <FRAMESET cols="1*, 250, 3*">
    <FRAME src = "x.html" frameborder = 0>
    <FRAME src = "y.html" frameborder = 0 marginwidth=25>
    <FRAME src = "flowers.jpg">
  </FRAMESET>
</HTML>
```

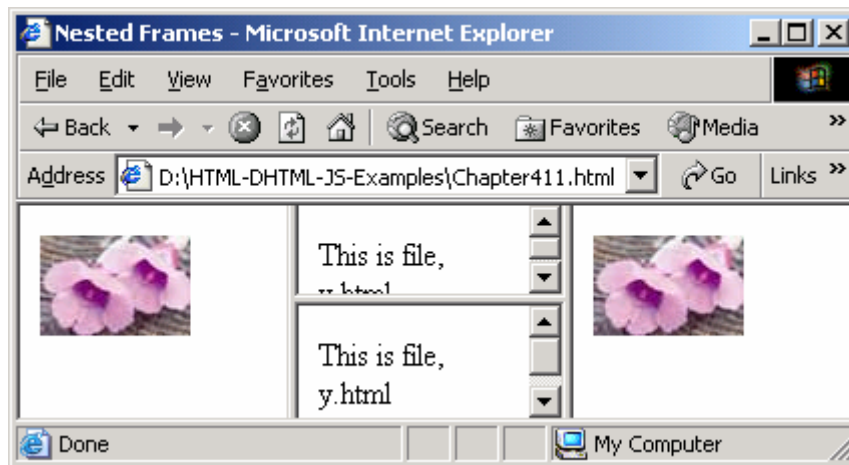


Hình 4.12: Kết quả ví dụ 10

Chú ý trong ví dụ 9, ta sử dụng phần tử FRAMESET lồng nhau (Nested framesets). Ta có thể tạo ra các frameset (tập khung) lồng nhau ở bất kỳ mức nào như ví dụ minh họa dưới đây:

Ví dụ 11:

```
<HTML>
  <HEAD>
    <TITLE>Nested Frames</TITLE>
  </HEAD>
  <FRAMESET cols="33%, 33%, 34%">
    <FRAME src = "flowers.jpg">
    <FRAMESET rows="40%, 50%">
      <FRAME src = "x.html">
      <FRAME src = "y.html">
    </FRAMESET>
  </FRAMESET>
</HTML>
```



Hình 4.13: Kết quả ví dụ 11

☐ Liên kết các khung

Khi tạo liên kết trong trang Web, ta có thể thiết lập một khung như một mục tiêu của liên kết (link). Trình duyệt tuân theo các bước sau:

- ☐ Nếu ta xác định một khung trong thuộc tính đích (TARGET) của phần tử, thì tài liệu được phần tử chỉ ra sẽ được tải vào khung đó khi phần tử được kích hoạt.
- ☐ Nếu thuộc tính TARGET không được thiết lập thì thuộc tính TARGET của phần tử BASE sẽ được sử dụng để xác định khung
- ☐ Nếu cả phần tử và phần tử BASE không đề cập đến TARGET, thì tài liệu được tải vào khung chứa phần tử đó.
- ☐ Nếu không tìm thấy khung thì trình duyệt tạo một cửa sổ và khung mới sau đó tải tài liệu vào khung mới này.

Thuộc tính TARGET được sử dụng để xác định tên khung mà tài liệu được mở trong đó. Khi tạo khung, ta cần phải đặt thuộc tính tên. Tên này được dùng khi tạo liên kết. Sau khi thay đổi nội dung của một khung thì định nghĩa frameset ban đầu bị mất đi. Nếu có nhiều liên kết đến cùng một đích, ta có thể thiết lập một TARGET mặc định trong phần tử BASE. Sau đó, việc xác định thuộc tính TARGET trong mỗi phần tử sẽ không cần thiết nữa.

Ví dụ 12:

```
<HTML>
  <FRAMESET COLS="40%, 60%">
    <FRAME SRC="FLOWERS.JPG" NAME="Flowers" SCROLLING="yes">
    <FRAME SRC="LINK.HTML" NAME="Links" SCROLLING="no"
      FRAMEBORDER="no">
  </FRAMESET>
</HTML>
```

Link.html

```
<HTML>
  <BODY>
```

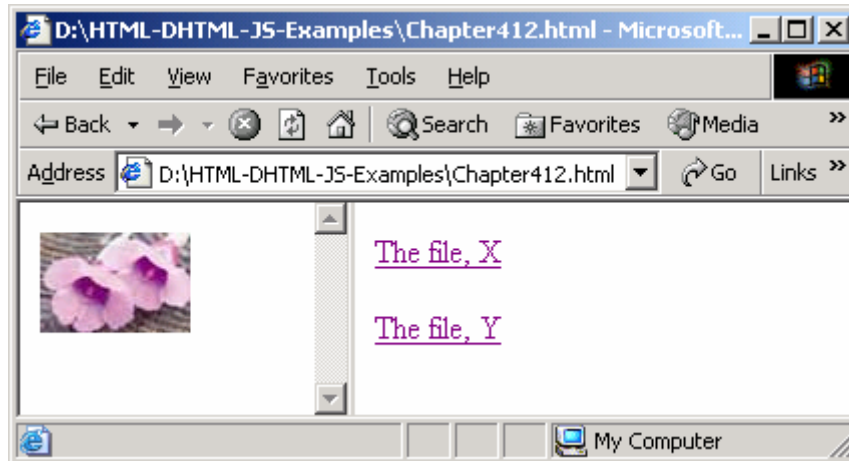
```
<BASE TARGET="Main">

<P><A HREF="X.HTML">The file, X</A><P>

<P><A HREF="Y.HTML">The file, Y</A><P>

</BODY>

</HTML>
```



Hình 4.14: Kết quả ví dụ 12

□ Phần tử NOFRAMES

Nếu trình duyệt không hỗ trợ khung, với tư cách là người phát triển ứng dụng ta nên cung cấp một cách khác để hiển thị nội dung. Phần tử NOFRAMES được sử dụng để làm việc đó. Nó chỉ hoạt động trong trình duyệt không hỗ trợ khung.

Ví dụ 13:

```
<HTML>

<FRAMESET COLS="40%,60%">

  <FRAME SRC="Flowers.jpg" NAME="Flowers" SCROLLING=yes>

  <FRAMESET ROWS="60,*">

    <FRAME SRC="x.html" NAME="x" SCROLLING=no

    FRAMEBORDER=no>

    <FRAME SRC="y.html" NAME="y">

    <NOFRAMES>

      Frames are not being displayed. Click here <A

      href="main.htm">for a non-frames version</A>

    </NOFRAMES>

  </FRAMESET>

</FRAMESET>

</HTML>
```

4.3.3 Phần tử IFRAME – Khung trên dòng (inline frame)

Phần tử IFRAME được sử dụng để tạo khung trên dòng (inline frame) hay khung nổi (floating frame). Ta có thể tạo và chèn một khung vào một khối văn bản. Khi trình

duyệt không hỗ trợ khung thì nội dung nằm trong thẻ IFRAME bị trả lại. Khung trên dòng (inline frame) không thể thay đổi kích thước.

Thuộc tính	Mô tả
NAME	Gán tên cho khung hiện thời
WIDTH	Xác định độ rộng của khung trên dòng
HEIGHT	Xác định chiều cao của khung trên dòng

Ví dụ 14:

<HTML>

```
<P>It has been good fun. You have been learning about frames
```

```
<BR><BR>
```

```
<IFRAME src="x.html" width="100" height="150" scrolling=auto  
frameborder=1>
```

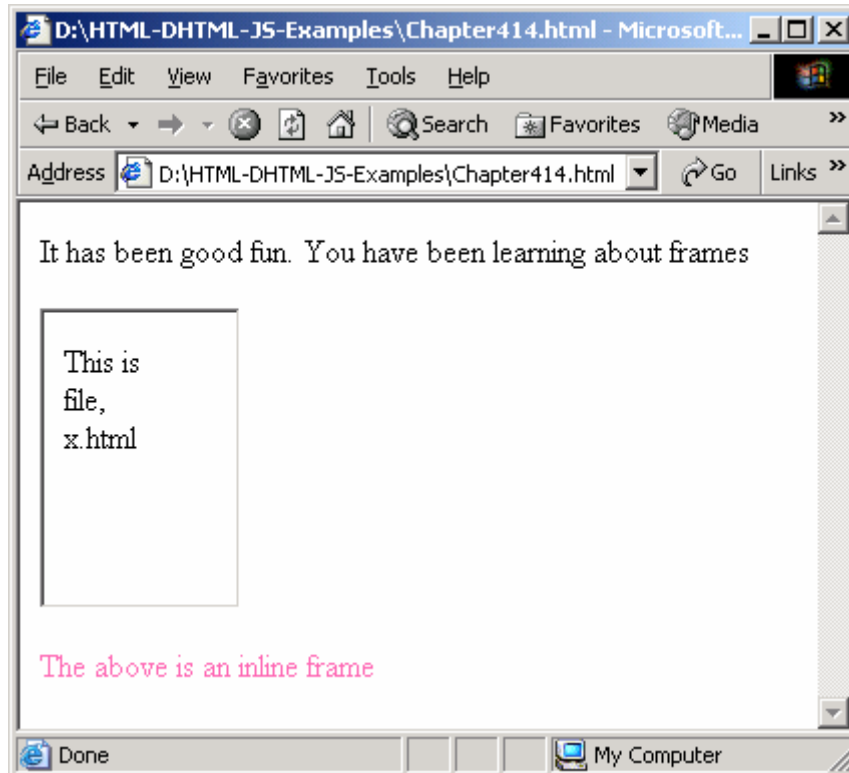
```
[The browser does not support frames or is currently configured  
not to display frames. These are the <A href="x.html">contents</A>]
```

```
</IFRAME>
```

```
<BR>
```

```
<P><FONT color = hotpink>The above is an inline frame
```

</HTML>



Hình 4.15: Kết quả ví dụ 14 (trình duyệt có hỗ trợ khung)

CHƯƠNG 5: SỬ DỤNG STYLE

Kết thúc chương này, bạn có thể:

- Khái quát về DHTML
- Sử dụng style sheet

5.1 GIỚI THIỆU

Trước đây, mỗi khi một trang web được hiển thị trong một trình duyệt, người ta không thể thay đổi bất cứ thứ gì trên đó. Cả người dùng lẫn tác giả của trang web đều không thể có bất kỳ điều khiển nào đối với các phần tử của HTML trên trang web. Sau này, với những phiên bản mới hơn của những trình duyệt đã hỗ trợ một đặc tính gọi là HTML động. Trong phần này, chúng ta sẽ thảo luận về HTML động và một số những điểm mới lạ mà nó mang đến cho những nhà thiết kế web.

Thêm vào đó, phần này cũng thảo luận về những stylesheet (style sheet cách). Stylesheet là một đặc tính quan trọng có thể được dùng trong HTML động. Mặc dù trang Web không cần có một stylesheet, nhưng việc sử dụng một stylesheet sẽ mang lại những lợi ích nhất định. Chúng ta sẽ thảo luận về stylesheet như là một kỹ thuật và bằng cách nào để có thể sử dụng nó trong việc thiết kế và phát triển Web

5.2 DHTML

“HTML động” có thể được định nghĩa như là một phần mềm được sử dụng cho việc mô tả sự kết hợp giữa HTML, các stylesheet và ngôn ngữ script làm cho tài liệu trở nên sinh động.

Mọi thứ bắt đầu ra sao? Vào những ngày đầu, HTML được phát triển như một định dạng tài liệu được dùng để trao đổi thông tin trên Internet. Việc truyền tải dữ liệu nó độc lập với nền tảng(platform). Tuy nhiên, trọng tâm đã chuyển từ các vấn đề khoa học và hàn lâm qua các vấn đề của người dùng hằng ngày, những người hiện nay đang xem Internet như là nguồn thông tin và giải trí. Các trang web trở nên hấp dẫn và nhiều màu sắc hơn làm thu hút người dùng. Tuy nhiên, các dáng vẽ cơ bản và nội dung của những trang web vẫn cố định. Người phát triển rất ít hoặc không có sự điều khiển nào cả đối với một trang web khi nó được hiển thị. HTML vẫn còn “tĩnh”

5.2.1 Giới thiệu DHTML

Sự ra đời của lập trình script đã thêm vào phần động cho các trang web. Người dùng có thể tương tác với trang web. Tuy nhiên, một số hạn chế vẫn còn tồn tại. Bất kỳ sự xác nhận dữ liệu nào hoặc việc lập trình script đều phải được thực hiện trên máy chủ. Để thay đổi nội dung hoặc một kiểu của trang, thì trang đó phải được viết đè lên hoàn toàn. Bất cứ tương tác nào của người dùng cũng đều thông qua máy Web server.

Với mỗi phiên bản trình duyệt mới, lại thêm vào các đặc tính tốt hơn cho HTML. Ngày nay Internet và Netscape Navigator hỗ trợ một mô hình đối tượng tài liệu “*Document Object Model*” mà ở đó các phần tử HTML và các thẻ được coi như một đối tượng. Những đối tượng có những phương thức, thuộc tính và sự kiện có thể lập trình để điều khiển các hoạt động của chúng. Ví dụ, cú pháp thêm vào để thay đổi màu của văn bản trong phần tử phân đoạn <p> khi người dùng kích chuột lên nó.

Các script (là một chương trình nhỏ) có thể thực thi trong trình duyệt. Điều này có nghĩa là dữ liệu có thể được thao tác, định dạng và thay đổi một cách năng động ở máy khách (client) mà không cần quá nhiều sự hỗ trợ từ máy chủ. Tương tác của người dùng giờ đây có thể được xử lý bởi chính máy khách.

Chú ý: Một ứng dụng Client/Server được tách ra hai phần một là giao diện người dùng “front-end client” và phần “back-end server”. Client là một phần của ứng dụng, nó trình bày dữ liệu đối với người dùng. Thông thường Client “giao diện người dùng” không thực thi các chức năng của cơ sở dữ liệu, thay vào đó, client gửi các yêu cầu dữ liệu đến một máy chủ “server”, định dạng và hiển thị kết quả. Vai trò của máy chủ “server” cung cấp xử lý hoặc thông tin đến cho client. Máy chủ cung cấp dữ liệu đến client, nhưng đôi khi, Máy chủ “server” có thể cần thực hiện một số công việc xử lý đem lại một kết quả dữ liệu yêu cầu. Ví dụ nếu một client yêu cầu về dữ liệu bán hàng cho một vùng cụ thể, Server cần thực hiện chính xác một số xử lý dữ liệu từ tập tất cả các dữ liệu và định dạng nó theo yêu cầu từ phía client.

Mỗi công ty Microsoft hay Netscape đều có cách triển khai HTML động riêng của họ. Microsoft tập trung vào dùng các Cascading Style Sheet (CSS). Chúng ta có thể dùng script để tương tác những phần tử CSS.

Netscape, ngược lại, dựa vào các phương thức dùng các tầng. Thẻ LAYER được dùng để cung cấp hầu hết các thuộc tính của HTML động.

5.2.2 Các đặc điểm của DHML

DHTML không dừng lại ở một số phần mở rộng của HTML. Trong phần này sẽ thảo luận về các đặc điểm của DHTML và cách thức dùng nó để thêm vào những tính năng động cho trang web.

- **Kiểu động (Dynamic Style)-** Trong các phiên bản của HTML trước đây, nếu chúng ta muốn thay đổi kiểu hay nội dung của một trang web sau khi nó được hiển thị trong trình duyệt thì toàn bộ trang đó phải được nạp lại ‘refresh’. Điều đó có nghĩa là yêu cầu phải được gửi đến máy chủ thành một trang mới để hiển thị. Đối với người dùng thì đây là một quy trình rõ ràng. Tuy nhiên, nếu trang đó phải được nạp lại ‘refresh’ thường xuyên sẽ tốn nhiều thời gian và làm cho máy chủ trở nên quá tải.

Trong HTML động, Cách làm này khác một chút. Bằng cách dùng các bảng kiểu ‘style sheets’, Chúng ta có thể xác định màu, kiểu hoặc kích cỡ của nội dung trang. Chúng ta có thể thay đổi kiểu của trang mà không cần gửi đến máy chủ ‘Web server’ cho mỗi lần thêm vào các thẻ và thuộc tính. Điều đó có nghĩa là chúng ta có thể thay đổi màu, font, kích cỡ hoặc nội dung văn bản khi đáp lại những tương tác của người dùng. Trong HTML động, kiểu liên quan đến cách thức, định dạng xuất hiện trên trang web hơn là nội dung. Kiểu ‘style’ bao gồm màu sắc, kiểu chữ, khoảng cách, thụt đầu dòng, định vị và xuất hiện của văn bản.

- **Nội dung động (Dynamic Content)-** Được hỗ trợ bởi Internet Explorer. Ở đây chúng ta có thể thay đổi chữ và hình ảnh trên trang web sau khi nó hiển thị. Chúng ta cũng có thể thay đổi nội dung của trang đó khi đáp lại dữ kiện nhập vào hay sự kiện người dùng kích chuột vào.

- **Định vị (Positioning)**- Các phiên bản của HTML trước đây, không kiểm soát được vị trí của một phần tử trên trang web. Theo đó vị trí chính xác của trang đó về mặt tọa độ thì không thể chỉ ra được. Việc định vị dành cho trình duyệt. HTML chỉ có thể mô tả nội dung và vị trí tương đối của các phần tử.

Trong HTML động, Chúng ta có thể chỉ ra vị trí chính xác (tuyệt đối hay tương đối), mối quan hệ giữa tọa độ x và y. Một khi trang web được hiển thị, chúng ta có thể di chuyển các phần tử trên trang đó làm cho nó trở nên động.

- Định vị tuyệt đối – chỉ rõ vị trí chính xác theo các điểm ‘pixels’.
- Định vị tương đối – chỉ ra vị trí tương đối của các phần tử. Trình duyệt xử lý việc định vị hiện thời

Đặc điểm việc định vị cũng cho phép chúng ta xác định thứ tự sắp xếp ‘z-order’ của các phần tử. Có nghĩa là các đối tượng này nằm chồng lên đối tượng khác.

- **Liên kết dữ liệu (Data Binding)** – Trong HTML động, chúng ta có thể kết nối một cơ sở dữ liệu vào bảng của trang web. Nó được hỗ trợ bởi Internet Explorer. Khi trang được nạp lên, dữ liệu từ cơ sở dữ liệu trên máy chủ được hiển thị trong bảng. Dữ liệu có thể được sắp xếp, lọc và hiển thị cho phù hợp với yêu cầu.
- **Downloadable Fonts (Có khả năng tải Font chữ)**- Được Netscape hỗ trợ. Downloadable fonts là một đặt điểm cho phép ta chèn các font mà tùy chọn trên trang web. Chúng ta có thể gói font trong trang. Điều này đảm bảo rằng văn bản trong trang web đó luôn luôn hiển thị theo font mà ta chọn. Đây là đặc điểm quan trọng bởi vì thông thường nếu các font được chỉ ra không có trong máy của người dùng thì trình duyệt sẽ dùng font mặc định có sẵn. Điều này sẽ làm hủy bỏ mục đích chỉ ra kiểu font của bạn.
- **Scripting** – Chúng ta có thể viết các script để thay đổi kiểu và nội dung của trang web. Script này được lồng vào trong trang web.
- **Cấu trúc đối tượng (Object Structure)** – HTML động theo một cấu trúc đối tượng theo đó mỗi phần tử được đối xử như một đối tượng trong cấu trúc. Mỗi đối tượng có thể được truy cập và lập trình độc lập.

5.3 STYLE SHEETS

Stylesheet được tạo nên từ các qui tắc kiểu cách, mà nó báo cho trình duyệt biết được cách trình bày một tài liệu. Stylesheet là một kỹ thuật thêm vào các kiểu (font, màu, khoảng cách) cho những trang web.

5.3.1 Khái niệm, chức năng và lợi ích của Style Sheets

Một tính năng quan trọng của HTML động là những kiểu(style) động. Nghĩa là bạn có thể thay đổi kiểu của những phần tử HTML trên trang sau khi chúng được hiển thị trên trang ấy. Sự thay đổi này có thể đáp ứng đối với sự tương tác người dùng hoặc thậm chí đối với sự thay đổi tình trạng như sự kiện thay đổi kích thước.

Có hai cách để thay đổi kiểu trên trang web của chúng ta:

- Thay đổi kiểu nội tuyến (inline style)

☐ Viết kịch bản để thay đổi kiểu

Khi sử dụng kiểu nội tuyến, chúng ta có thể tạo ra các kiểu động mà không thêm bất cứ một kịch bản nào vào trang của chúng ta.

Một kiểu nội tuyến là kiểu được gán trực tiếp cho một phần tử nào đó. Kiểu này không áp dụng vào tất cả các phần tử thuộc một loại nào hay một lớp nào đó. Kiểu nội tuyến được định nghĩa bằng thuộc tính STYLE đối với phần tử của thẻ đó. Ví dụ nếu muốn đặt màu cho phần tử <H1> màu đỏ, chúng ta phải đặt thuộc tính STYLE bên trong thẻ H1 như sau:

```
<H1 STYLE=color:red>
```

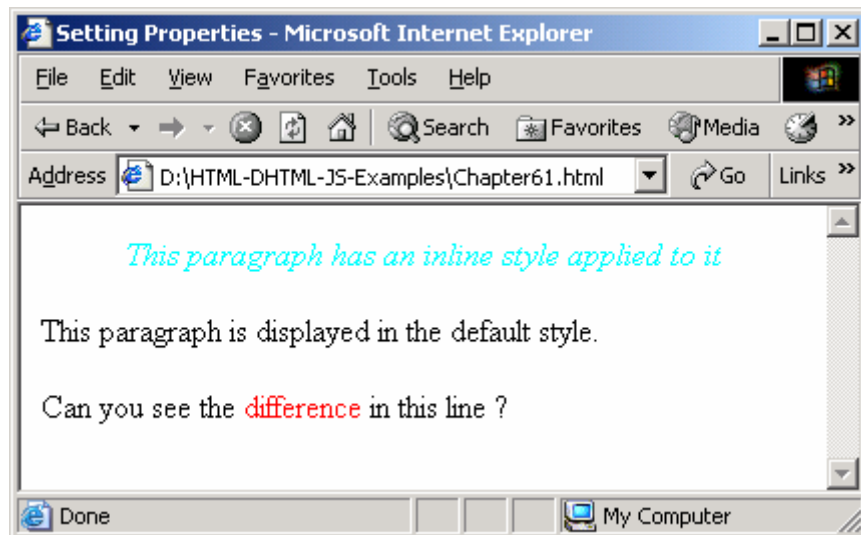
Nếu chúng ta muốn sử dụng kịch bản để thay đổi kiểu nội tuyến vào bất cứ lúc nào, thì lúc ấy bạn phải sử dụng đến đối tượng kiểu (**Style Object**). Đối tượng kiểu hỗ trợ mọi tính chất mà CSS hỗ trợ đối với các kiểu. Để dùng thuộc tính trong kịch bản:

- ☐ Bỏ dấu gạch nối ra khỏi tên của kiểu CSS
- ☐ Thay đổi chữ cái đầu tiên của từ sau dấu gạch nối thành từ viết hoa.

Ví dụ, **font-weight** của CSS trở thành **fontWeight** trong DHTML hoặc **text-align** thành **textAlign**

Ví dụ 1:

```
<HTML>
  <HEAD>
    <TITLE>Setting Properties</TITLE>
  </HEAD>
  <BODY>
    <P style = color:aqua;font-Style:italic;text-Align:center;>
    This paragraph has an inline style applied to it
    <BR>
    <P> This paragraph is displayed in the default style.
    <BR>
    <P>Can you see the <SPAN style = color:red>difference </SPAN>
    in this line ?
  </BODY>
</HTML>
```



Hình 5.1: Kết quả ví dụ 1

- **Lợi ích của các stylesheet** – Các stylesheet có thể được dùng để:
 - **Nạp chồng trình duyệt:** - Mỗi trình duyệt đều có thể hiển thị các trang web theo cách riêng của nó. Trước đây các nhà phát triển không kiểm soát được các trang web hiển thị trên trình duyệt. Suy cho cùng bạn không biết trình duyệt nào mà người dùng cách nửa vòng trái đất sử dụng. Nhờ có các stylesheet bạn có thể nạp chồng các qui ước của trình duyệt và đặt theo cách riêng của chúng ta. Chẳng hạn, bạn có thể xác định kiểu mà trong đó một phần tử <H1> cần hiển thị:

```
<H1><FONT SIZE=3 COLOR=AQUA>  
<B>Overriding the browser</B></FONT></H1>
```
 - **Bố cục trang (Page layout)** – Những stylesheet có thể dùng để hiển thị font thay đổi màu mà không làm thay đổi cấu trúc của trang web. Điều này có nghĩa là với tư cách là một nhà thiết kế bây giờ bạn có thể tách biệt những yêu cầu về thiết kế hình ảnh trực quan từ cấu trúc logic của trang web và địa chỉ là hai chuyên hoàn toàn khác nhau.

Khi sử dụng các biện pháp liên quan trong stylesheet của bạn, bạn có thể thể hiện các tài liệu sao cho đẹp mắt trên bất kỳ màn hình nào và theo bất kỳ độ phân giải nào.
 - **Sử dụng lại các stylesheet** – Một khi đã định nghĩa kiểu thông tin, chúng ta có thể nhúng stylesheet bên trong tài liệu HTML. Lần lượt thay thế, chúng ta có thể kết nối tất cả các trang trên website đến stylesheet. Điều này chắc chắn rằng các trang web của chúng ta đều có cùng diện mạo thông tin khi được hiển thị. Vì vậy, bạn có thể có được nền chung của trang ví dụ như logo của trang và một số thông tin chuẩn(cho các trang) trong một stylesheet. Điều này sẽ đảm bảo được cách nhìn và cảm nhận thông dụng về trang website. Cứ thử hình dung xem có vài trăm trang web và bạn phải xác định kiểu của mỗi trang một cách độc lập.

- **Chỉ cần làm một lần thật tốt** – Chúng ta có thể tạo một stylesheet và được liên kết đến nhiều tài liệu. Tất cả những tài liệu sẽ có diện mạo giống nhau. Tuy nhiên, quan trọng nhất là khi bạn thực hiện thay đổi stylesheet thì tất cả các tài liệu được kết nối vào stylesheet sẽ bị thay đổi theo.

5.3.2 Quy tắc Style Sheets

Stylesheet phân cấp(cascading style sheet) định nghĩa các kiểu có thể được áp dụng vào các trang hoặc các phần tử của trang.

- **Qui tắc kiểu (Style Rule)**- Stylesheet phân cấp là một tập hợp các qui tắc. Qui tắc định nghĩa kiểu của tài liệu. Ví dụ, chúng ta có thể tạo ra một qui tắc kiểu được xác định cho tất cả phần tiêu đề <H1> hiển thị màu vàng xanh. Qui tắc kiểu có thể ứng dụng vào các thành phần được chọn của trang web. Ví dụ, chúng ta có thể xác định một đoạn văn bản bất kỳ in đậm và in nghiêng trên trang. Điều này được gọi là **khai báo kiểu có sẵn** mà nhờ đó các kiểu được áp dụng vào các phần tử HTML đơn lẻ trên một trang web.
- **Style Sheet** – Là một danh mục các qui tắc kiểu và có thể nhúng vào bên trong tài liệu HTML. Trong trường hợp đó, nó được gọi là stylesheet nhúng. Stylesheet cũng có thể được tạo ra bằng một file bên ngoài và được liên kết với tài liệu HTML.
- **Các qui tắc (Rules)** – Bảng kiểu có thể có một hay nhiều qui tắc. Phần đầu của qui tắc gọi là bộ chọn (Selector). Mỗi bộ chọn có các thuộc tính và các giá trị liên quan đến nó.

RuleSelector {Declarations property: value; property: value; ... }

Phần của qui tắc được kèm theo trong phạm vi các dấu ngoặc móc được gọi là **khai báo**. Khai báo có hai phần, phần trước dấu hai chấm (:) là thuộc tính và phần nằm sau dấu hai chấm là giá trị của thuộc tính đó.

Các khai báo được phân cách bởi dấu chấm phẩy (;). Ta nên đặt dấu chấm phẩy sau lần khai báo cuối cùng.

Ví dụ như

```
H1 {color: blue}
```

Ở đây, H1 là bộ chọn, color: blue là khai báo

Trong phạm vi khai báo:

{Property: Value}

Color là thuộc tính, blue là giá trị.

Mỗi qui tắc có thể tách rời nhau trong phạm vi thẻ STYLE.

Ví dụ 2:

```
<HTML>
```

```
<HEAD>
```

```
<STYLE TYPE="text/css">
```

```
H1 {color:limegreen}
H1 {font-family:Arial}
H2 {color:limegreen}
H2 {font-family:Arial}

</STYLE>

</HEAD>

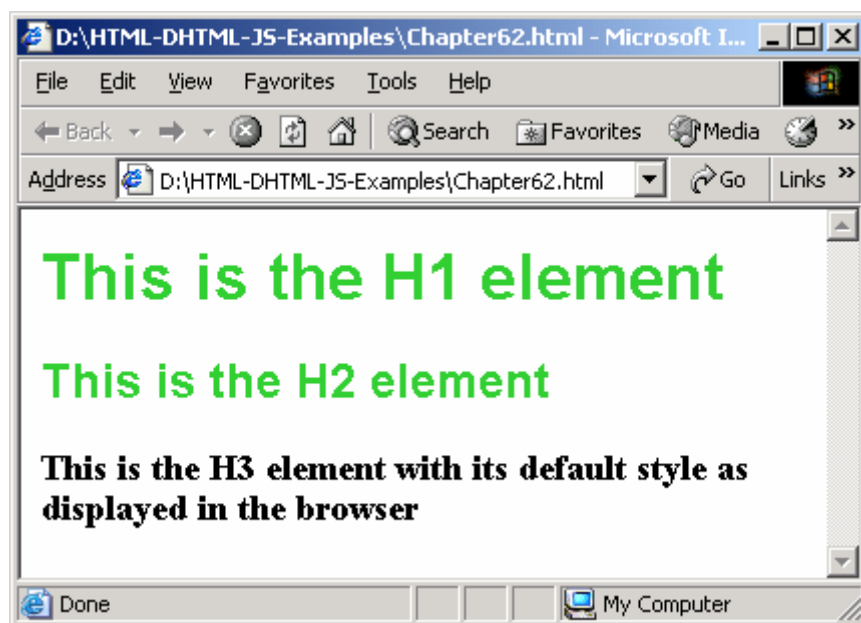
<H1>This is the H1 element</H1>

<H2>This is the H2 element</H2>

<H3>This is the H3 element with its default style as displayed in the
browser</H3>

</HTML>
```

Lưu ý: text/css chỉ ra kiểu ‘style’ được dùng ở đây là stylesheet phân cấp ‘cascading style sheet’



Hình 5.2: Kết quả ví dụ 2

Thay vào đó chúng ta có thể nhóm các qui tắc. Mỗi khai báo được tách ra bởi dấu chấm phẩy.

Ví dụ 3:

```
<HTML>

  <HEAD>

    <STYLE TYPE="text/css">

      H1, H2{color:limegreen;font-family:Arial;}

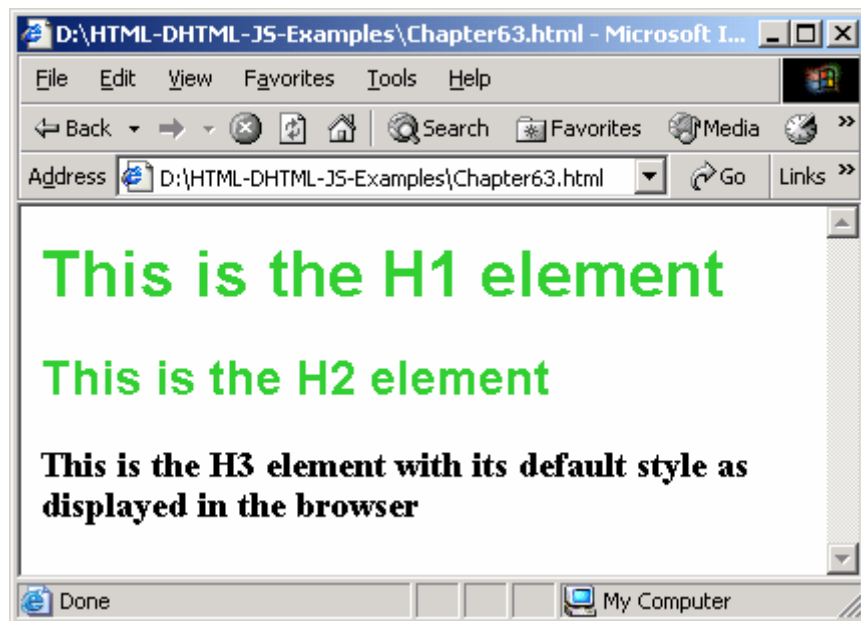
    </STYLE>

  </HEAD>

  <H1>This is the H1 element</H1>

  <H2>This is the H2 element</H2>
```

```
<H3>This is the H3 element with its default style as displayed in the browser</H3>  
</HTML>
```



Hình 5.3: Kết quả Ví dụ 3

5.3.3 Các Selector trong Style Sheets

Ta có thể dùng các giả lớp trong các selector nhưng không thể dùng chúng trong HTML. Về cơ bản, chúng dùng những thông tin bên ngoài để tác động đến việc định dạng. Ví dụ, với việc sử dụng lớp giả, các liên kết đã được ghé qua có thể hiển thị khác đi so với các liên kết chưa được ghé qua như sau:

```
A:link { color: red }          /* unvisited link */  
A:visited { color: blue }     /* visited links */  
A:active { color: lime }      /* active links */
```

Selector có thể định nghĩa như là “một chuỗi kí tự xác định ra các phần tử và các quy tắc tương ứng áp dụng cho các phần tử ấy”.

Có hai kiểu selector cơ bản:

☐ Selector đơn

Đây là những selector dễ sử dụng nhất. Selector đơn mô tả một phần tử bất chấp vị trí của nó ở đâu trong cấu trúc tài liệu. Bộ nhận dạng tiêu đề H1 là một điển hình. Sau đây là một số kiểu của selector đơn.

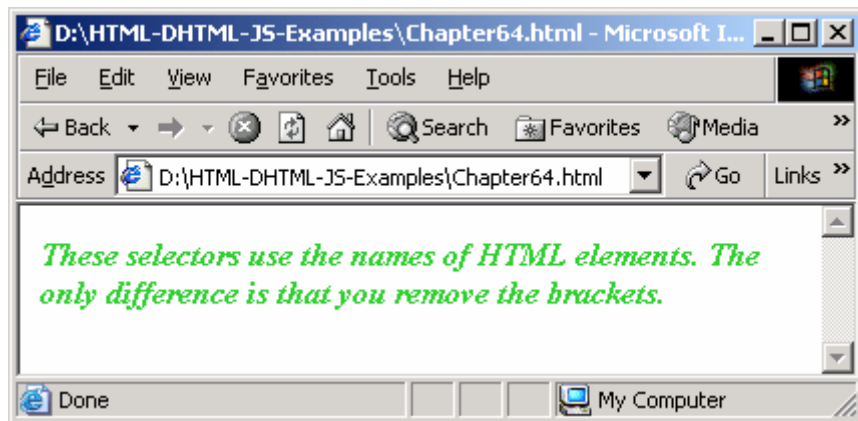
☐ Selector HTML

Những selector này sử dụng tên của phần tử HTML và bỏ đi dấu móc . Vì vậy, thẻ <P> trong HTML trở thành P và khi đó, nó được xem như là một selector. Ví dụ sau đây minh họa điều đó

Ví dụ 4:

```
<HTML>
```

```
<HEAD>
    <STYLE TYPE="text/css">
        P{font-style:italic;           font-weight:bold;
        color:limegreen}
    </STYLE>
</HEAD>
<Body>
    <P> These selectors use the names of HTML elements. The
    only difference is that you remove the brackets. </P>
</BODY>
</HTML>
```



Hình 5.4: Kết quả ví dụ 4

□ Selector lớp

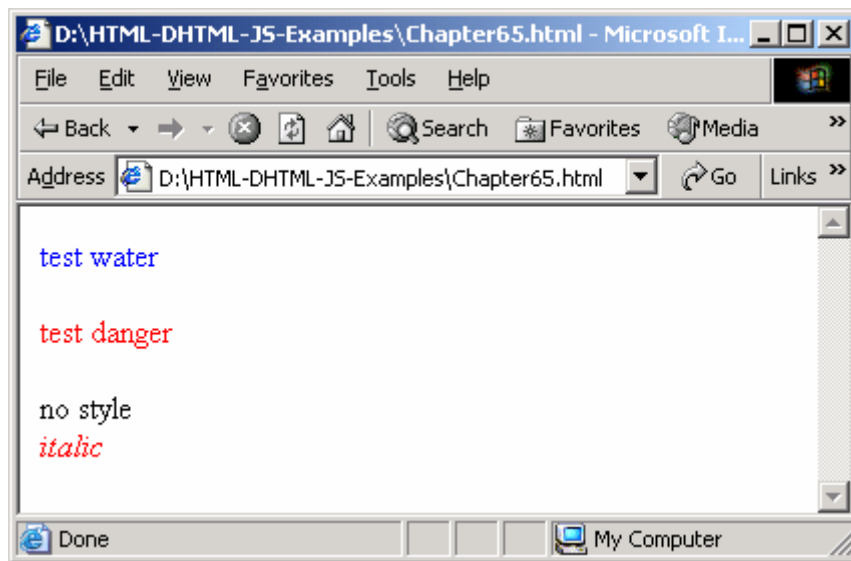
Những selector này sử dụng thuộc tính CLASS của các phần tử HTML. Mọi phần tử khi hiển thị có một thuộc tính CLASS được sử dụng để gán vào một định danh (identifier). Ta có thể gán một tên lớp duy nhất cho mọi phần tử khác. Ngoài ra, ta cũng có thể gán định danh lớp cho nhiều phần tử cùng loại khi ta muốn hiển thị các trạng thái khác nhau so với dạng chuẩn. Ví dụ, ta có thể muốn <H2> xuất hiện với nhiều màu khác nhau. Trong trường hợp đó, ta sử dụng định danh lớp cho <H2>.

Selector lớp có dấu chấm (.) đứng trước gọi là ký tự cờ(flag), theo sau là tên lớp do chúng ta chọn. Tốt hơn hết nên chọn những tên lớp theo mục đích của chúng chứ không nên chọn tên mô tả màu sắc hay kiểu của chúng. Ví dụ, ta có thể muốn đoạn A hiển thị chữ nghiêng, những đoạn khác thì hiển thị theo kiểu khác. Trong trường hợp đó, đoạn A có thể có bộ nhận dạng lớp là .slant

Ví dụ 5:

```
<HTML>
    <HEAD>
        <STYLE TYPE="text/css">
            .water { color:blue }
            .danger { color:red }
```

```
</STYLE>
</HEAD>
<BODY>
    <P class=water>test water
    <P class=danger>test danger
    <P> no style
    <BR>
    <EM class=danger>italic</EM>
</BODY>
</HTML>
```



Hình 5.5: Kết quả ví dụ 5

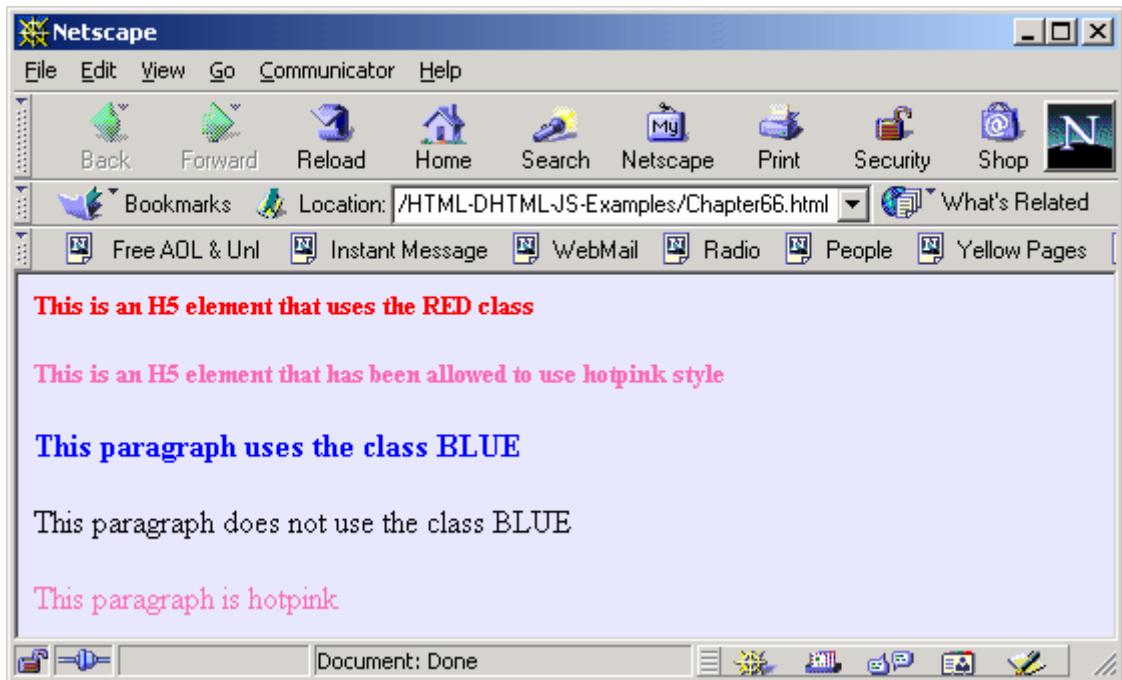
Khi xác định một lớp kiểu, ta có thể xác định được phần tử HTML nào có thể sử dụng kiểu này. Chúng ta nên sử dụng từ khóa **all**, để tất cả các phần tử đều có thể sử dụng nó.

Ví dụ 6:

```
<HTML>
<HEAD>
    <STYLE type="text/css">
        all.hotpink {color:hotpink;}
        P.BLUE {color:blue; font-weight:bold;}
        H5.RED {color:red; font-weight:bold;}
    </STYLE>
</HEAD>
<H5 CLASS=RED>This is an H5 element that uses the RED
class</H5>
    <H5 CLASS=hotpink>This is an H5 element that has been allowed
to use hotpink style.</H5>
```

```
<BODY bgColor=lavender>
  <P CLASS=BLUE>This paragraph uses the class BLUE</P>
  <P>This paragraph does not use the class BLUE</P>
  <P CLASS=hotpink>This paragraph is hotpink
</BODY>
</HTML>
```

Chú ý: Xem kết quả trong Netscape



Hình 5.6: Kết quả ví dụ 6

☐ Selector ID

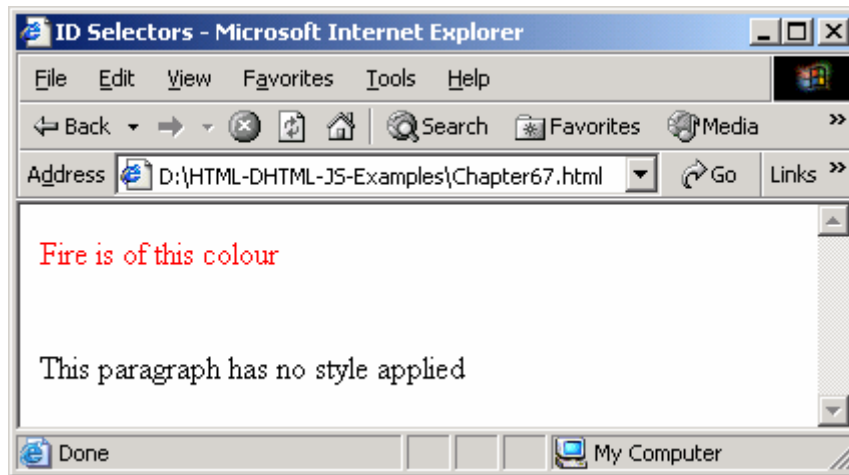
Selector ID sử dụng thuộc tính ID của phần tử HTML. Selector ID được dùng để áp dụng một kiểu vào riêng một phần tử nào đó trên trang Web. Ví dụ, ta có thể sử dụng một selector ID để áp dụng một kiểu đặc biệt nào đó cho phần tử <H2>. Điều đó không có nghĩa là một kiểu tương tự được áp dụng cho một phần tử <H2> khác trên trang đó, nếu không được xác định. Tương tự với việc sử dụng kiểu nội tuyến mà nhờ đó có thể áp dụng riêng cho một phần tử nào đó. Selector ID được bắt đầu bằng dấu thăng (#).

Ví dụ 7:

```
<HTML>
  <HEAD>
    <TITLE> ID Selectors</TITLE>
    <STYLE TYPE="text/css">
      #control {color:red}
    </STYLE>
  </HEAD>
```



```
<BODY>
    <P id="control">Fire is of this color</H2>
    <BR>
    <P>This paragraph has no style applied
</BODY>
</HTML>
```

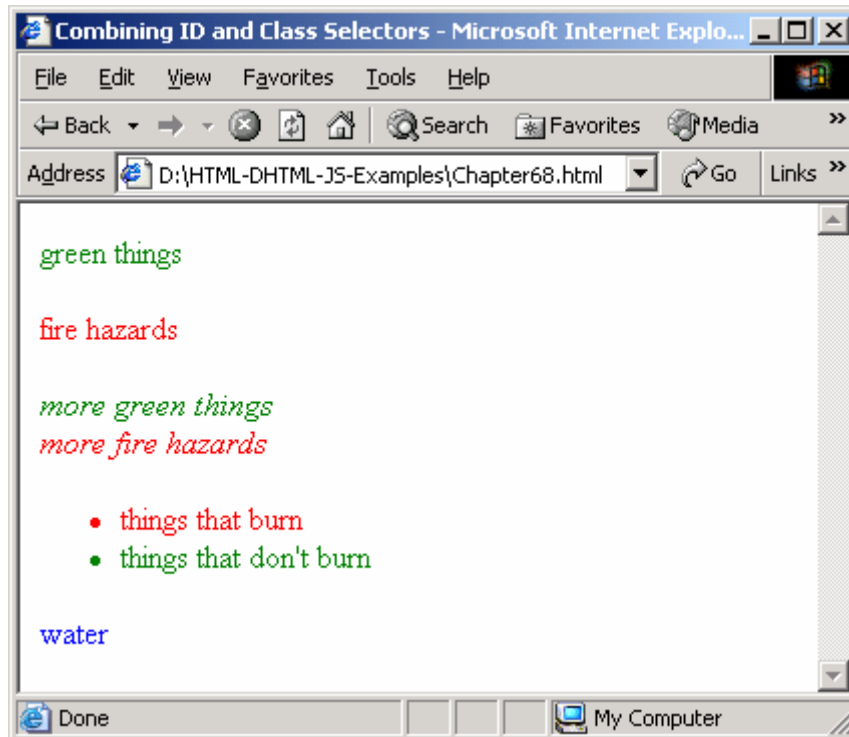


Hình 5.7: Kết quả ví dụ 7

Ví dụ 8:

```
<HTML>
  <HEAD>
    <TITLE> Combining ID and Class Selectors</TITLE>
    <STYLE TYPE="text/css">
      .forest {color:green}
      .danger {color:red}
      #control {color:blue}
    </STYLE>
  </HEAD>
  <BODY>
    <P class=forest>green things
    <P class=danger>fire hazards
    </P>
    <EM class=forest>more green things</EM>
    <BR>
    <EM class=danger>more fire hazards</EM>
    <UL>
      <LI class=danger>things that burn
      <LI class=forest>things that don't burn
    </UL>
```

```
<P id=control> water </P>
</BODY>
</HTML>
```



Hình 5.8: Kết quả ví dụ 8

□ Selector ngữ cảnh

Selector ngữ cảnh liên quan đến ngữ cảnh của phần tử. Ví dụ, thỉnh thoảng ta có hai phần tử cùng giá trị. Phần tử chính hay phần tử cha có một phần tử con bên trong nó. Để thay đổi kiểu của phần tử con, ta phải sử dụng selector theo ngữ cảnh. Điều này dựa trên khái niệm kế thừa, phần tử con kế thừa kiểu được gán cho thẻ cha.

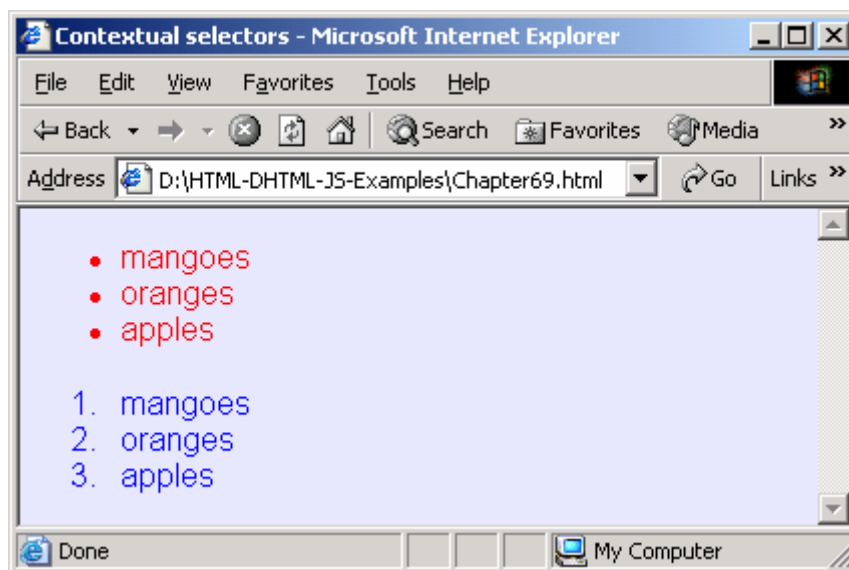
Một ví dụ điển hình là phần tử <BODY>. Khi thêm một phần tử vào thẻ <BODY>, thì mỗi phần tử bên trong sẽ kế thừa các kiểu của <BODY>.

Bây giờ làm sao để kiểm soát điều đó? Suy cho cùng, ta không muốn tất cả các phần tử trên trang Web xuất hiện cùng một kiểu. Trong trường hợp đó, chúng ta phải có sự thay đổi đặc biệt đối với các phần tử con, nói một cách khác đó là sự nạp chồng kế thừa.

Ví dụ 9:

```
<HTML>
  <HEAD>
    <TITLE>Contextual selectors</TITLE>
    <STYLE TYPE="text/css">
      BODY {color:blue;
        background:lavender;
```

```
        font-family:Arial; }
        UL {color:red }
    </STYLE>
<HEAD>
<BODY>
    <UL>
        <LI> mangoes
        <LI> oranges
        <LI> apples
    </UL>
    <OL>
        <LI> mangoes
        <LI> oranges
        <LI> apples
    </OL>
</BODY>
</HTML>
```



Hình 5.9: Kết quả ví dụ 9

Selector UL trong style sheet xác định các mục trong danh sách(list item) được hiển thị màu đỏ. Chúng kế thừa phông chữ Arial từ khai báo BODY, màu đỏ từ khai báo UL. Nếu ta xác định font-family trong khai báo UL, nó sẽ nạp chồng lên khai báo của selector BODY. Không có selector OL trong style sheet, vì thế các thuộc tính của OL kế thừa từ selector BODY.

5.3.4 Kết hợp, liên kết và chèn một Style Sheet vào tài liệu HTML

Có một số cách ta có thể kết hợp style sheet với HTML

1. Phần tử STYLE

2. Thuộc tính Style

3. Phần tử Link

□ Phần tử STYLE

Phần tử STYLE được chèn vào phần tử <HEAD> của tài liệu, tất cả các quy tắc được định nghĩa giữa thẻ mở và thẻ đóng. Khi hiển thị các trang, thì chỉ tài liệu nào có những STYLE mới được tác động. Ví dụ:

```
<Style Type      = "text/css">
    H1      {color:maroon;}
    P      {color:hotpink; font-family:Arial;}
</Style>
```

Ví dụ trên là cách sử dụng phần tử <STYLE>

□ Thuộc tính STYLE

Thuộc tính STYLE được sử dụng để áp dụng style sheet cho từng phần tử. Một style sheet có thể nhỏ như một luật. Khi sử dụng thuộc tính Style ta có thể bỏ qua phần tử STYLE và đặt khai báo trực tiếp vào thuộc tính Style trong thẻ mở của phần tử. Ví dụ:

```
<H2 style="color: green;
        font-family: Arial">
</H2>
```

Rõ ràng là chúng ta chỉ nên dùng kiểu này khi thay đổi kiểu cho một phần tử đặc biệt nào đó. Nếu ta có dự định áp dụng cùng kiểu trên khắp tài liệu thì lúc ấy đây không phải là cách hay. Ví dụ trên là cách sử dụng thuộc tính STYLE.

□ Phần tử Link

Nếu ta muốn sử dụng phần tử LINK, thì stylesheet của ta phải là một tài liệu riêng. Sau đó chúng ta phải thêm địa chỉ Web của stylesheet vào. Ví dụ,

```
<LINK      REL      = stylesheet
        HREF      = "stylesmine.css"
        Type      = "text/css" >
```

Thuộc tính “rel=stylesheet” phải được khai báo nếu không thì trình duyệt sẽ không tải được stylesheet .

Ví dụ 10:

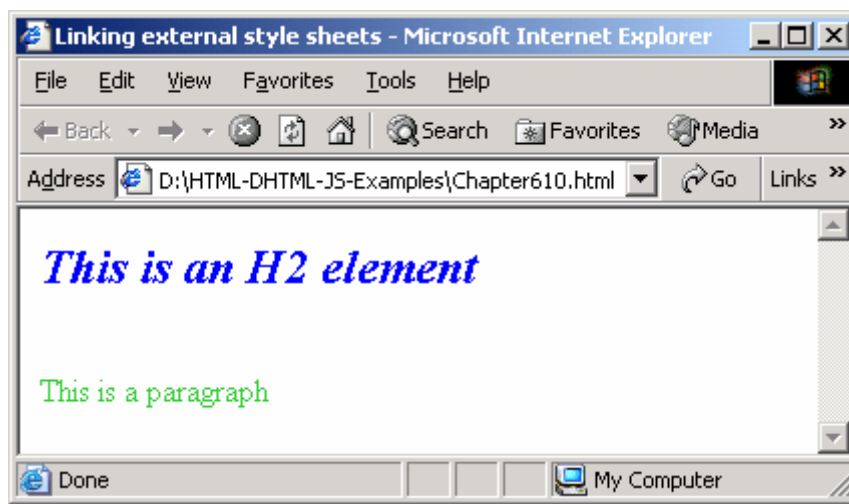
Sheet1.css

```
H2 {color:blue; font-style:italic;}
P {color: limegreen;}
```

Tập .htm

```
<HTML>
    <HEAD>
        <TITLE>Linking external style sheets</TITLE>
```

```
<LINK REL=STYLESHEET TYPE="text/css"
      HREF="sheet1.css">
</HEAD>
<H2>This is an H2 element</H2>
<BR>
<BODY>
      <P>This is a paragraph
</BODY>
</HTML>
```



Hình 5.10: Kết quả ví dụ 10

5.3.5 Thiết lập thuộc tính trong Style Sheet

Chúng ta có thể thiết lập nhiều thuộc tính trong stylesheet. Bảng 11.1 cho thấy nhiều thuộc tính có thể được sử dụng trong stylesheet.

Thuộc tính	Tên CSS
Các thuộc tính font	Font
	font-size
	font-style
Các thuộc tính văn bản	text-align
	text-indent
	vertical-align
Các thuộc tính hộp	border
	border-width
	border-bottom

	border-color
Các thuộc tính về vị trí	Clip
	height
	Left
	Top
	z-index

CHƯƠNG 6: JAVASCRIPT, NỀN TẢNG VÀ CÚ PHÁP

Kết thúc chương này, bạn có thể:

- Hiểu được cơ bản về JavaScript
- Nắm vững kiến thức cơ bản cú pháp JavaScript

6.1 GIỚI THIỆU

JavaScript là một ngôn ngữ kịch bản (scripting language) được dùng để tạo các script ở máy **client** (client-side script) và máy **server** (server-side script). Các script ở máy **client** được thực thi tại trình duyệt, các script ở máy server được thực hiện trên server. Chương này sẽ giới thiệu cho chúng ta về ngôn ngữ Javascript, và cách chèn một script vào trong tài liệu HTML.

6.2 GIỚI THIỆU VỀ JAVASCRIPT

HTML lúc đầu được phát triển như là một định dạng của tài liệu có thể chuyển dữ liệu trên Internet. Tuy nhiên, không lâu sau đó, trọng tâm của HTML nặng tính hàn lâm và khoa học dần chuyển hướng sang người dùng thường nhật vì ngày nay người dùng xem Internet như là một nguồn thông tin và giải trí. Các trang Web ngày càng mang tính sáng tạo và đẹp mắt hơn nhằm thu hút nhiều người dùng hơn. Nhưng thực chất kiểu dáng và nội dung bên trong vẫn không thay đổi. Và người dùng hầu như không thể điều khiển trên trang Web mỗi khi nó được hiển thị.

Javascript được phát triển như là một giải pháp cho vấn đề nêu trên. Javascript là một ngôn ngữ kịch bản được Sun Microsystems và Netscape phát triển. Nó được dùng để tạo các trang Web động và tương tác trên Internet. Đối với những người phát triển HTML, Javascript rất hữu ích trong việc xây dựng các hệ thống HTML có thể tương tác với người dùng.

6.2.1 Javascript là gì?

Sun Microsystems đã viết ra một ngôn ngữ phức tạp và mạnh mẽ mà chúng ta đã biết đó là ngôn ngữ Java. Mặc dù Java có tính khả dụng cao nhưng nó lại phù hợp nhất đối với các nhà lập trình có kinh nghiệm và cho các công việc phức tạp hơn. Netscape Communications nhận thấy nhu cầu cần một ngôn ngữ thiết kế web có khả năng tương tác với người sử dụng hay với các Java Applet, để sử dụng ngay cả với những người lập trình ít kinh nghiệm.

LiveScript là một ngôn ngữ mới chỉ ở dạng phác thảo, tuy nhiên nó hấp dẫn người sử dụng vì hứa hẹn sẽ đáp ứng tốt những yêu cầu trên. LiveScript được thiết kế theo tinh thần của nhiều ngôn ngữ script đơn giản nhưng nó lại có tính khả dụng cao được thiết kế đặc biệt để xây dựng các trang Web (chẳng hạn như HTML và các form tương tác). Để giúp ‘bán chạy’ ngôn ngữ mới này, Netscape hợp tác với Sun cho ra đời ngôn ngữ Javascript. Trên thực tế, Microsoft là người tiên phong triển khai phiên bản của Javascript (còn có tên là Jscript), nhưng họ không sử dụng các đặc tả chính thức của Javascript.

Mục tiêu của JavaScript là nhằm cung cấp cho các nhà phát triển Web một số khả năng và quyền điều khiển chức năng cho trang Web. Mã Javascript có khả năng nhúng

trong tài liệu HTML để điều khiển nội dung của trang Web và kiểm tra sự hợp lệ của dữ liệu mà người dùng nhập vào. Khi một trang hiển thị trong trình duyệt, các câu lệnh được trình duyệt thông dịch và thực thi.

Định nghĩa

JavaScript là một ngôn ngữ kịch bản dựa trên đối tượng nhằm phát triển các ứng dụng Internet chạy trên phía client và phía server.

Các ứng dụng client được chạy trong một trình duyệt như Netscape Navigator hoặc Internet Explorer, và các ứng dụng server chạy trên một Web server như Microsoft's Internet Information Server hoặc Netscape Enterprise Server.

6.2.2 Hiệu ứng và quy tắc Javascript

JavaScript là một ngôn ngữ lập trình được nhúng được trong các trang HTML. JavaScript nâng cao tính động và khả năng tương tác cho web-site bằng cách sử dụng các hiệu ứng của nó như thực hiện các phép tính, kiểm tra form, viết các trò chơi, bổ sung các hiệu ứng đặc biệt, tùy biến các chọn lựa đồ họa, tạo ra các mặt khẩu bảo mật và hơn thế nữa.

Chúng ta có thể sử dụng JavaScript để:

- **Tương tác với người dùng.** Chúng ta có thể viết mã để đáp lại các sự kiện. Các sự kiện này sẽ có thể phát sinh bởi người dùng - - nhấp chuột hay được phát sinh từ hệ thống - định lại kích thước của trang và v.v.
- **Thay đổi nội dung động.** Mã JavaScript có thể dùng để thay đổi nội dung và vị trí các phần tử một cách động trên một trang nhằm đáp lại sự tương tác với người dùng.
- **Kiểm tra tính hợp lệ dữ liệu.** Chúng ta có thể viết mã nhằm kiểm tra tính hợp lệ của dữ liệu do người dùng nhập vào trước khi nó được gửi lên Web server để xử lý.

Giống như các ngôn ngữ khác, JavaScript cũng tuân thủ một số quy tắc ngữ pháp căn bản. Việc nắm vững các quy tắc ngữ pháp này có thể giúp ta đọc được script và tự viết các script không bị lỗi.

Một số trong các luật này bao gồm:

- **Dùng Caps.** JavaScript phân biệt chữ hoa chữ thường
- **Dùng Pairs.** Trong JavaScript, luôn luôn có cặp ký hiệu mở và đóng. Lỗi sẽ xuất hiện khi bỏ sót hoặc đặt sai một trong hai ký hiệu này.
- **Dùng Spaces (các ký tự trắng).** Như HTML, JavaScript thường bỏ qua ký tự trắng. Trong JavaScript, ta có thể thêm vào các ký tự trắng hoặc các tab giúp cho ta dễ dàng đọc hay sửa các file script.
- **Dùng Chú thích (Comments).** Các chú thích giúp ta ghi chú về chức năng của đoạn script, thời gian và người tạo ra đoạn script.

Mặc dù cả client-side JavaScript và server-side JavaScript đều dựa trên một ngôn ngữ nền tảng như nhau, nhưng mỗi loại còn có thêm những tính năng chuyên biệt phù hợp với môi trường mà nó chạy. Nghĩa là, client-side JavaScript bao gồm các đối tượng

được định nghĩa sẵn chỉ có thể sử dụng trên trình duyệt, Server-side JavaScript bao gồm các đối tượng và các hàm được định nghĩa sẵn chỉ có thể sử dụng trong các ứng dụng phía server (server-side applications)

6.2.3 Các công cụ Javascript và IDE, môi trường thực thi.

Các công cụ sinh mã JavaScript và IDE giúp tạo ra mã JavaScript rất hữu hiệu. . Các công cụ này còn giúp ta nhanh chóng phát triển website của mình.

Một vài công cụ JavaScript và IDE được đề cập dưới đây:

- **Dialog Box.** Công cụ này tự động tạo mã để sinh ra các hộp thoại tùy biến trên các trình duyệt khác nhau (**alert**, **confirm**, **prompt**, v.v.) mang lại kiểu dáng chuyên nghiệp cho website.
- **Pop-up Menu builder.** Chỉ cần đưa vào các lựa chọn, công cụ này sẽ tự động tạo ra các pop-up menu trên các trình duyệt khác nhau.
- **Remotes.** Tự động sinh mã để mở ra một cửa sổ popup.

Như chúng ta đã biết, JavaScript có thể được chạy trên máy khách (client) và máy chủ (server). Bên phía máy khách, trình duyệt sẽ thực thi mã lệnh javascript trong trang web khi mở nó. Bên phía máy chủ, mã javascript sẽ được thực thi tại máy chủ và do máy chủ thực hiện.

➤ **Client-side Java Script**

Khi máy client yêu cầu một trang HTML, server sẽ kiểm tra xem trang đó có chứa script hay không. Nếu nó chứa các client-side script, server sẽ chuyển toàn bộ tài liệu bao gồm mã lệnh JavaScript và nội dung HTML của nó cho trình duyệt. Khi trình duyệt nhận được tài liệu đó, nó thực thi các mã lệnh HTML và JavaScript mà không cần bất kỳ sự tương tác nào với server.

➤ **JavaScript trên Web Server**

Chúng ta có thể nhúng các lệnh JavaScript chạy trên server (server-side script) vào trong tài liệu HTML. Quá trình tạo ra các ứng dụng server-side là một quá trình gồm hai giai đoạn.

1. Các trang HTML có chứa các câu lệnh JavaScript của cả client-side và server-side đều được tạo ra cùng với các file JavaScript. Tất cả các file này sẽ được biên dịch thành dạng mã thực thi được là bytecode.
2. Khi trình duyệt yêu cầu trang HTML, run-time engine sẽ thực thi mã lệnh server-side JavaScript rồi trả trang HTML về cho trình duyệt.

Một số công dụng của script server-side bao gồm:

- Kết nối vào các cơ sở dữ liệu
- Chia sẻ thông tin cho những người dùng của một ứng dụng
- Truy cập vào hệ thống file trên server

6.2.4 Nhúng Javascript vào trang Web

Chúng ta có thể chèn các lệnh JavaScript vào trong một tài liệu HTML theo những cách sau đây:

1. Nhúng các câu lệnh trực tiếp vào trong tài liệu bằng cách sử dụng thẻ `<SCRIPT>`
2. Liên kết file nguồn JavaScript với tài liệu HTML
3. Đặt các biểu thức JavaScript làm giá trị cho thuộc tính của thẻ HTML.
4. Dùng như trình xử lý sự kiện trong các thẻ HTML.

Chúng ta có xem chi tiết một số ví dụ sau đây:

➤ Dùng thẻ **SCRIPT**

Mã JavaScript cụ thể được nhúng vào trong tài liệu HTML bằng thẻ `SCRIPT`. Chúng ta có thể nhúng nhiều script vào trong cùng một tài liệu, mỗi script nằm trong một thẻ `SCRIPT`. Khi trình duyệt gặp phải một thẻ `<SCRIPT>` nào đó, nó sẽ đọc từng dòng một cho đến khi gặp thẻ đóng `</SCRIPT>`. Tiếp đến nó sẽ kiểm tra lỗi trong các câu lệnh JavaScript. Nếu gặp phải lỗi, nó sẽ cho hiển thị lỗi đó trong chuỗi các hộp cảnh báo (alert boxes) lên màn hình. Nếu không có lỗi, các câu lệnh sẽ được biên dịch sao cho máy tính có thể hiểu được lệnh đó.

Cú pháp như sau:

```
<script language="JavaScript">
<!--
JavaScript statements;
//-->
</script>
```

- Thuộc tính `language` trong thẻ script chỉ ra ngôn ngữ mà trình duyệt sẽ dùng để biên dịch script. Chúng ta cũng có thể chỉ rõ phiên bản JavaScript nào sẽ được trình duyệt sử dụng. Ví dụ:

```
<script language="JavaScript1.2">
```

- `<!--statements-->` là các thẻ chú thích. Những thẻ này được dùng để báo cho trình duyệt bỏ qua các câu lệnh chứa trong nó. `<!--` là thẻ mở của thẻ chú thích, `-->` là thẻ đóng. Các thẻ này không bắt buộc phải có, nhưng ta nên đưa chúng vào trong các đoạn script. Chỉ có Netscape 2.0 và các phiên bản sau đó mới hỗ trợ JavaScript. Các thẻ chú thích đảm bảo các phiên bản cũ hoặc các trình duyệt không hỗ trợ JavaScript sẽ bỏ qua các câu lệnh được nhúng trong tài liệu HTML.

Các câu lệnh JavaScript phải được kết thúc bằng dấu chấm phẩy (;).

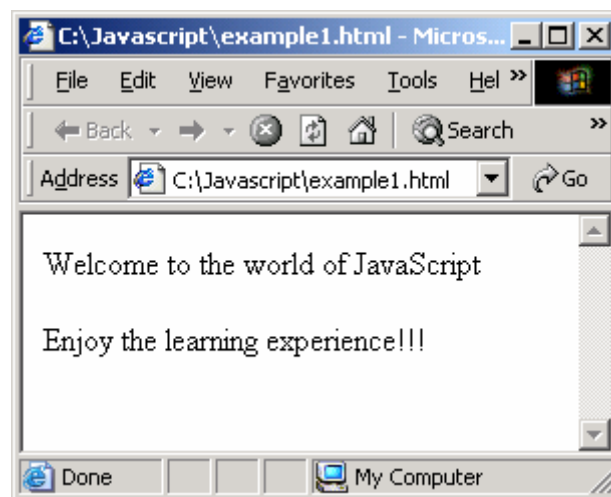
Trong ví dụ sau đây, ngôn ngữ script được đặt là JavaScript. Các thẻ chú thích được dùng để các phiên bản cũ của trình duyệt bỏ qua script nằm trong các thẻ chú thích. Các phiên bản trình duyệt mới sẽ thực thi các câu lệnh JavaScript.

Ví dụ 1:

```
<HTML>
```

```
<HEAD>
  <script language = "JavaScript">
    <!-- hide from older browsers>
      document.write("Welcome to the world of JavaScript");
    // Script hiding ends here -->
  </script>
</head>
<body>
  <P>Enjoy the learning experience!!!
</BODY>
</HTML>
```

Kết quả:



Hình 6.1: Dùng JavaScript

Trên lý thuyết các câu lệnh JavaScript có thể đặt ở bất kỳ nơi nào trong tài liệu HTML. Tuy nhiên, nên đặt các câu lệnh script trong phần `<head>` và `</head>`. Điều này đảm bảo tất cả các câu lệnh đều được đọc và biên dịch trước khi nó được gọi từ trong phần BODY.

➤ Dùng file bên ngoài

Thường các câu lệnh JavaScript được nhúng trong một tài liệu HTML. Tuy nhiên, chúng ta có thể tạo ra một file riêng chứa mã JavaScript. File này có thể được liên kết với một tài liệu HTML. Thuộc tính SRC (source) của thẻ SCRIPT dùng để chỉ ra file chứa JavaScript mà nó cần sử dụng. Khi xác định file nguồn, ta có thể dùng tên đường dẫn tương đối và tuyệt đối ở trong thuộc tính SRC.

```
<script language="JavaScript" src="filename.js">
</script>
```

Đó là file văn bản chứa các mã lệnh JavaScript, tên file có phần mở rộng là “.js”. Nó chỉ có thể chứa các câu lệnh và các hàm JavaScript. Ta không thể đưa các thẻ HTML vào trong nó.

Trong ví dụ sau đây, có hai file được tạo ra. File thứ nhất - “test.htm” là một file tài liệu HTML. File thứ hai - “test.js” là file văn bản có chứa mã JavaScript. File này được liên kết với file tài liệu HTML.

Ví dụ 2:

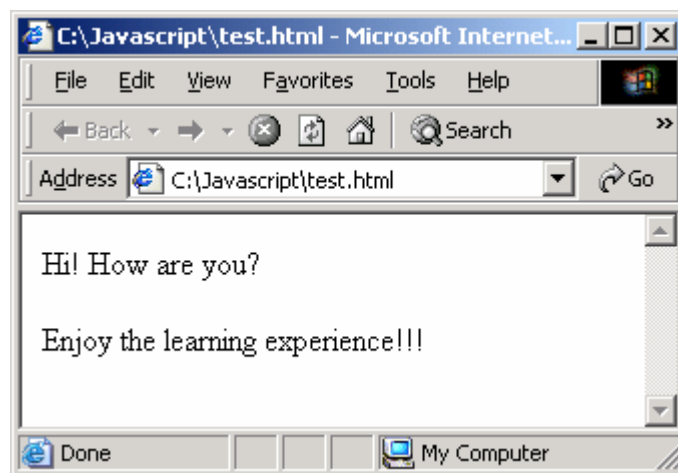
File HTML: (test.htm)

```
<HTML>
  <HEAD>
    <script src = "test.js"> </script>
  </head>
  <body>
    <P>Enjoy the learning experience!!!
  </body>
</html>
```

File nguồn JavaScript: (test.js)

```
document.write("Hi! How are you?")
```

Kết quả:



Hình 6.2: Dùng JavaScript – File ngoài

Chú ý: Lưu các file **test.htm** và **test.js** trong cùng một thư mục.

Đây là một ví dụ đơn giản mô tả tính năng liên kết các file riêng chứa mã lệnh JavaScript. Tuy nhiên, việc thực hiện chức năng liên kết các file sẽ rất có lợi khi ta muốn chia sẻ các hàm cho nhiều tài liệu HTML. Trong trường hợp này, chúng ta có thể tạo ra một file .js với các hàm thông thường. File này được liên kết với các tài liệu cần nó. Nếu ta muốn điều chỉnh hoặc thêm vào một vài hàm, ta chỉ cần thực hiện thay đổi trong một file mà thôi thay vì phải trong nhiều tài liệu HTML.

Đuôi .js phải được server ánh xạ đến kiểu MIME là application/x-javascript. Server lúc ấy sẽ gởi nó trong phần đầu (header) của giao thức HTTP.

Để ánh xạ đuôi .js với kiểu MIME ta thêm dòng code sau vào file chứa các kiểu MIME trong thư mục config của server. Sau đó ta sẽ khởi động lại server.

type=application/x-javascript exts=js

Nếu server không ánh xạ đuôi .js đến application/x-javascript MIME thì trình duyệt sẽ không tải đúng file JavaScript được xác định trong thuộc tính SRC.

➤ **Đặt các biểu thức JavaScript là giá trị cho các thuộc tính của thẻ HTML**

Chúng ta có thể dùng biểu thức JavaScript làm giá trị cho thuộc tính của thẻ HTML. Các giá trị đó được thực hiện một cách động mỗi khi trang được trình duyệt tải vào. Cú pháp như sau:

& {expression};

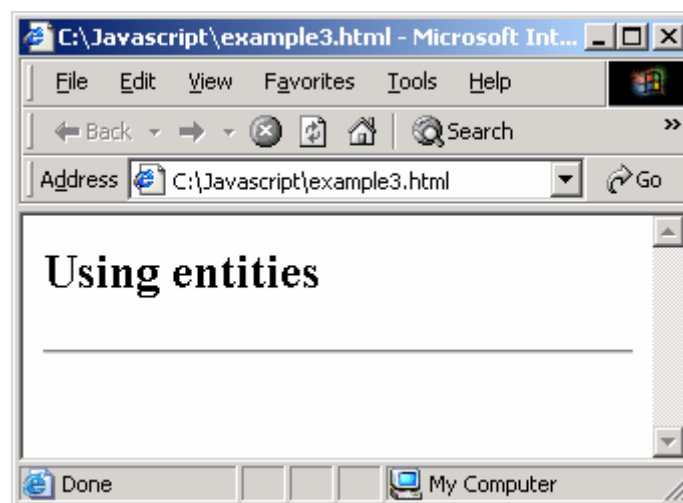
Trong đó expression là biểu thức JavaScript sẽ được thực hiện.

Chẳng hạn ta có thể định nghĩa một biến chứa độ rộng. Chúng ta có thể dùng biến này để xác định độ rộng của một đường kẻ ngang trên trang Web. Trong ví dụ sau đây, giá trị độ rộng được đặt là 10:

Ví dụ 3:

```
<HTML>
  <HEAD>
    <SCRIPT>
      var linewidth = 10;
    </SCRIPT>
  <BODY>
    <H2>Using entities</H2>
    <HR width="&{linewidth};%" align = "left">
  </BODY>
</HTML>
```

Khi mã được thực thi, thì phần tử HR sẽ dùng giá trị của biến linewidth như minh họa trong Hình 6.3



Hình 6.3: Dùng các biểu thức JavaScript

Một biểu thức JavaScript chỉ có thể được sử dụng bên phía phải của phần tên/giá trị (name/value pair) trong thẻ HTML. Chẳng hạn khi chúng ta dùng `HR WIDTH="&{linewidth};%"` câu lệnh sẽ được dịch là:

`<HR WIDTH= "10%">`

Nếu chúng ta sử dụng `<H4>&{myTitle};</H4>`, chuỗi **&myTitle;** sẽ được hiển thị thay vì giá trị của biến myTitle.

➤ Dùng JavaScript trong trình xử lý sự kiện

Chúng ta có thể tạo một trình xử lý sự kiện cho một thẻ HTML dùng mã JavaScript. Một sự kiện là một hành động được hỗ trợ bởi một đối tượng. Một trình xử lý sự kiện là đoạn mã sẽ được thực thi nhằm đáp trả một sự kiện. Cú pháp là:

`<TAG event handler="JavaScript code">`

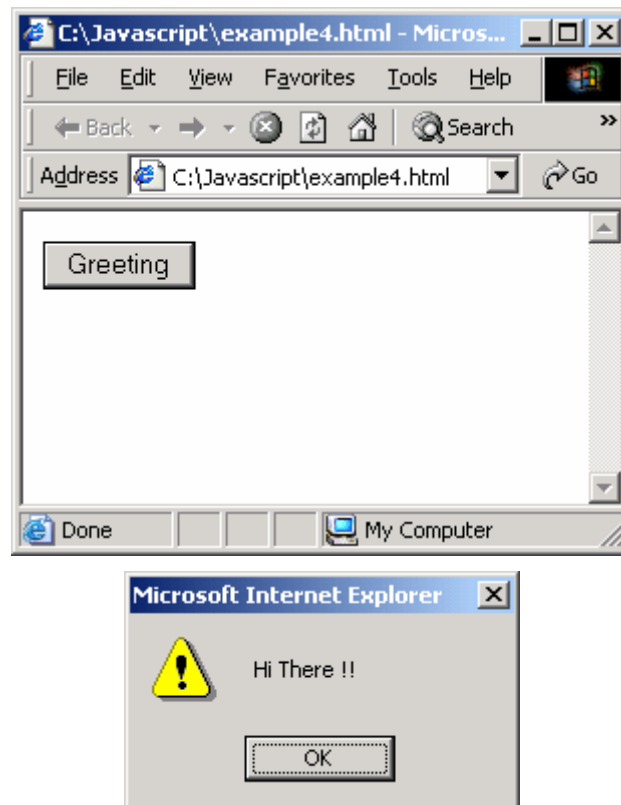
TAG là một thẻ HTML. Event handler là tên của trình xử lý sự kiện, và JavaScript code là một loạt các câu lệnh JavaScript được thực thi khi sự kiện được kích hoạt.

Trong ví dụ sau đây, sự kiện `onClick` được kích hoạt khi người dùng nhấp chuột vào phần tử `BUTTON`. Trình xử lý sự kiện được gọi để đáp trả sự kiện đó. Trình xử lý sự kiện có chứa mã JavaScript được thực thi bởi trình duyệt.

Ví dụ 4:

```
<HTML>
  <SCRIPT LANGUAGE = "JavaScript">
    function greeting () {
      alert ("Hi There !!");
    }
  </SCRIPT>
  <BODY>
    <FORM>
      <INPUT TYPE="button" VALUE="Greeting"
        onClick="greeting()" ">
    </FORM>
  </BODY>
</HTML>
```

Kết quả:



Hình 6.4: Dùng JavaScript - Alert

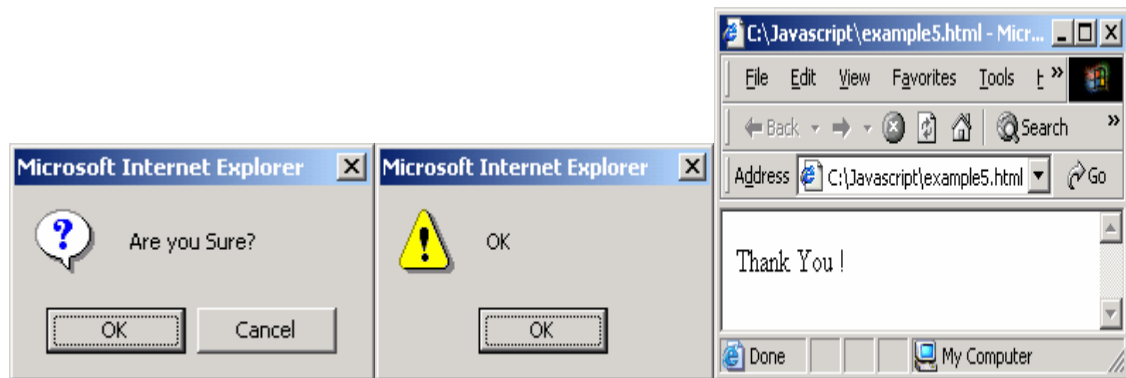
6.2.5 Ví dụ đơn giản sử dụng hộp thông báo và phương thức write

Trong ví dụ minh họa dưới đây, hướng dẫn cách sử dụng các phương thức confirm, alert và write.

Ví dụ 5:

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE = "Javascript">
      confirm ("Are you Sure?");
      alert("OK");
      document.write(" Thank You !");
    </SCRIPT>
  </HEAD>
</HTML>
```

Kết quả:



Hình 6.5: Dùng JavaScript – Confirm, Alert và Write

6.3 CÁC BIẾN

Biến là một tham chiếu đến một vị trí trong bộ nhớ. Nó dùng để chứa các giá trị có thể thay đổi khi script đang được thực thi. Chúng ta phải đặt tên cho biến. Chẳng hạn ta có thể tạo một biến có tên First. Tại mỗi thời điểm thực hiện, biến có thể chứa một giá trị mới. Khi chúng ta muốn xem giá trị của biến hoặc thay đổi giá trị của nó, ta chỉ cần dùng tên của biến.

Các biến thường có các quy ước đặt tên như sau:

- Tên biến phải bắt đầu bằng một chữ cái hoặc ký tự gạch dưới (" _")
- Tên biến có thể chứa chữ số
- JavaScript phân biệt rõ chữ hoa và chữ thường, vì vậy tên biến bao gồm các ký tự từ "A" đến "Z" (chữ hoa) và các ký tự từ "a" đến "z" (chữ thường)

6.3.1 Khai báo biến

Sử dụng từ khóa '**var**' để khai báo biến. Biến sẽ tồn tại ngay sau khi ta khai báo nó. Chúng ta có thể khởi tạo giá trị cho biến ngay khi khai báo:

```
var A = 10;
```

Chúng ta cũng có thể khai báo biến bằng cách gán giá trị cho nó mà không cần từ khóa var.

```
A = 10
```

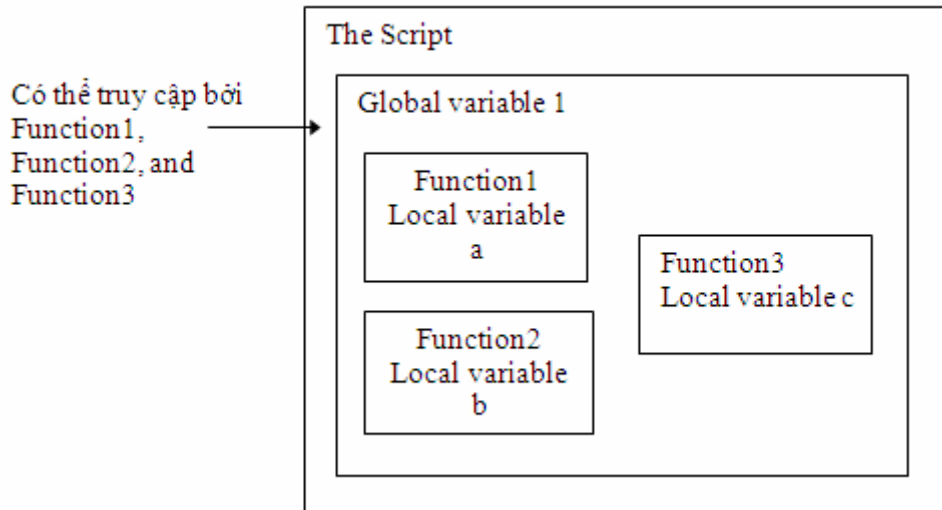
Chúng ta có thể khai báo nhiều biến trên cùng một dòng bằng cách tách tên các biến bằng dấu phẩy.

Đây là một số ví dụ các biến:

```
var new_amount  
var answer = null  
var old_cost = 12.50  
var result = 'Unknown result'  
result = true
```

6.3.2 Phạm vi của biến

Phạm vi của biến được xác định tại vị trí mà nó được khai báo trong script. Nếu chúng ta khai báo một biến ngay phần đầu của script, thì nó được xem là một biến **global (toàn cục)** và có thể truy cập ở bất kỳ nơi đâu trong script. Nếu ta khai báo biến trong một hàm, nó được xem là biến **local (cục bộ)** và nó chỉ được sử dụng chỉ đối với hàm đó mà thôi. Những hàm khác trong script không thể truy xuất vào biến đó được. Hình 6.6 minh họa phạm vi của các biến trong script.



Hình 6.6: Phạm vi của biến (variables)

Các biến toàn cục không cần thiết phải sử dụng từ khóa var khi khai báo. Tuy nhiên, các biến cục bộ luôn luôn phải sử dụng từ khóa var khi khai báo.

Literals

Các literals là những giá trị cố định mà ta có thể dùng trong script. Giá trị của literal không bị thay đổi trong quá trình thực hiện script.

6.4 CÁC KIỂU DỮ LIỆU

JavaScript có một tập hợp các kiểu dữ liệu như sau:

Kiểu dữ liệu	Mô tả
Numbers	Các số nguyên hoặc số thực. Ví dụ: 487 hoặc 25.95
Logical hoặc Boolean	True hoặc False
Strings	"Hello World"
Null	Keyword đặc biệt chỉ giá trị null.

JavaScript luôn phân biệt các chữ hoa và chữ thường; null không giống như Null hoặc NULL

Dữ liệu luôn luôn thuộc một kiểu nào đó. Biến có kiểu dữ liệu phụ thuộc vào dữ liệu mà nó lưu trữ, kiểu dữ liệu của biến sẽ bị thay đổi khi giá trị của nó thay đổi.

Khi chúng ta khai báo một biến thì không cần phải chỉ ra kiểu dữ liệu của nó. Các kiểu dữ liệu sẽ tự động thay đổi nếu cần thiết trong quá trình thực thi script. Ví dụ:

```
var x = 10
```

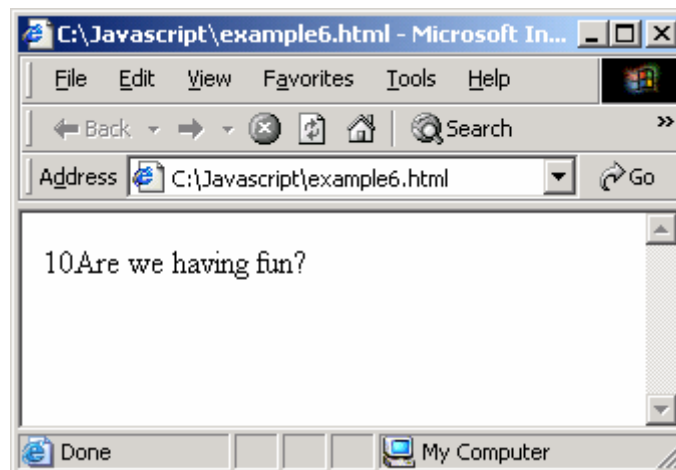
Sau đó, chúng ta có thể một chuỗi vào biến x trong script:

```
X = "Are you having fun?"
```

Ví dụ 6:

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE = "Javascript">
    var x = 10;
    document.write(x);
    var x = "Are we having fun?";
    document.write(x);
  </SCRIPT>
</HEAD>
</HTML>
```

Kết quả:



Hình 6.7: Các biến JavaScript:

Trong JavaScript, ta có thể kết hợp hai biến thuộc các kiểu khác nhau. Chẳng hạn, nếu chúng ta kết hợp một chuỗi với một giá trị số với nhau thì JavaScript sẽ chuyển đổi các giá trị số sang các chuỗi.

Ví dụ:

```
A = " This apple costs Rs." + 5
```

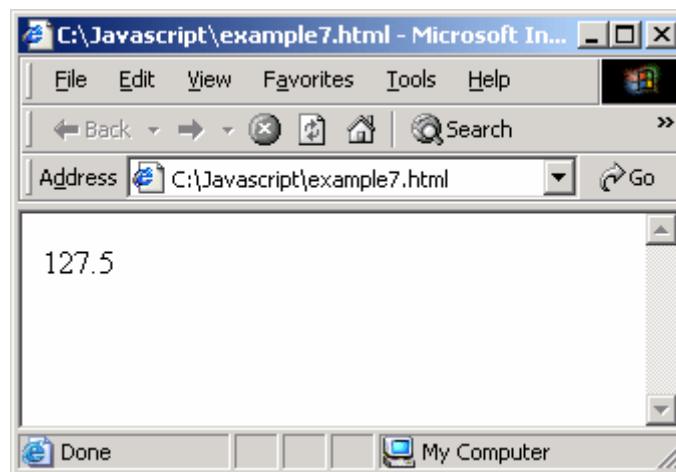
Nó sẽ cho ra kết quả là một chuỗi có giá trị "This apple costs Rs. 5"

Tuy nhiên, chúng ta không thể chuyển một chuỗi thành một số. Nếu chúng ta cộng một số thực 7.5 với một chuỗi là "12", kết quả sẽ là 127.5.

Ví dụ 7:

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE = "Javascript">
      var A = "12" + 7.5;
      document.write(A);
    </SCRIPT>
  </HEAD>
</HTML>
```

Kết quả:



Hình 6.8: JavaScript: variables (1)

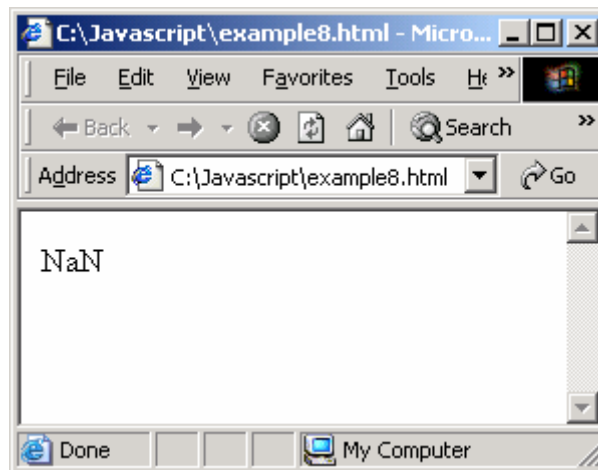
Trong JavaScript có hàm `parseInt()` và `parseFloat()` có chức năng chuyển đổi các chuỗi thành các số nguyên hoặc các số thực. Chẳng hạn, hàm `parseInt("15")` sẽ chuyển đổi chuỗi "15" sang giá trị là một số nguyên. Hàm `parseFloat("35.45")` sẽ chuyển một chuỗi thành một số thực là 35.45. Nếu hàm `parseFloat()` nhận ra một ký tự chứ không phải là một ký hiệu (+ hoặc -), một số (0 - 9), kí hiệu dấu thập phân hoặc một số mũ thì nó sẽ bỏ qua ký tự đó và tất cả các ký tự theo sau nó. Nếu ký tự đầu tiên không thể chuyển đổi được, hàm đó sẽ trả về giá trị "NaN" (Not a Number).

Trong ví dụ sau đây, chuỗi "a15" được chuyển vào hàm `parseFloat()`. Kết quả là NaN vì chuỗi đó không mở đầu bằng các ký tự có thể chuyển đổi được.

Ví dụ 8:

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE = "Javascript">
      var x = "a15"
      a = parseFloat(x);
      document.write(a);
    </SCRIPT>
  </HEAD>
</HTML>
```

Kết quả:



Hình 6.9: JavaScript: Ví dụ NaN

Literals có thể có bất kỳ các kiểu dữ liệu nào sau đây:

- Số nguyên** – Chúng có thể được biểu diễn trong hệ thập phân, hệ thập lục phân và hệ nhị phân.
- Số thực (Floating-point)** -- Các literal số thực (Floating-point literals) phải có ít nhất một chữ số. Số đó có thể có dấu thập phân hoặc “e” hoặc “E” theo sau một số nguyên. Số nguyên có thể dương (“+”) hoặc âm (“-”). Kí hiệu e hay E được hiểu là “lũy thừa cơ số 10”. Ví dụ: 10.24, 1.20e+22, 4E-8, .1815, v.v.
- Chuỗi** -- là chuỗi rỗng hoặc các ký tự được đặt trong dấu ngoặc đơn (`'Unexpected error'`) hoặc dấu ngoặc kép `"Hi! How are you"`. Chuỗi đó phải bắt đầu và kết thúc bằng một dấu ngoặc của cùng một kiểu.

Sau đây là một số các ví dụ về literals chuỗi:

```
'Unexpected error'
"Hi! How are you"
'4531'
```

Khi dùng chuỗi, ta có thể chèn các ký tự đặc biệt vào chuỗi đó. Các ký tự đặc biệt sẽ thực hiện một công việc cụ thể. Ví dụ: chúng ta có thể thêm một dấu vạch chéo ngược (\) trong chuỗi đó. Dưới đây là bản liệt kê các ký tự đặc biệt và nghĩa của chúng:

Ký tự	Ý nghĩa
\b	backspace
\f	form feed
\n	new line (xuống dòng và đưa con trỏ về đầu dòng đó)
\r	Carriage return (đưa con trỏ về đầu dòng hiện tại)
\t	Tab

Những ví dụ dưới đây minh họa cách sử dụng các ký tự đặc biệt của JavaScript:

➤ **Ký tự backspace**

```
<SCRIPT LANGUAGE = "Javascript">  
  alert(" This is line one \b This is line two");  
</SCRIPT>
```

➤ **Ký tự form feed (sang trang)**

```
<SCRIPT LANGUAGE = "Javascript">  
  alert(" This is line one \f This is line two");  
</SCRIPT>
```

➤ **Ký tự new line (xuống dòng)**

```
<SCRIPT LANGUAGE = "Javascript">  
  alert(" This is line one \n This is line two");  
</SCRIPT>
```

➤ **Ký tự carriage return (Phím xuống dòng)**

```
<SCRIPT LANGUAGE = "Javascript">  
  alert(" This is line one \r This is line two");  
</SCRIPT>
```

➤ **Ký tự tab**

```
<SCRIPT LANGUAGE = "Javascript">  
  alert(" This is line one \t This is line two");  
</SCRIPT>
```

Ngoài các ví dụ nêu trên, chúng ta có thể chèn các ký tự khác trong một chuỗi bằng cách đặt trước nó dấu backslash (\). Đây được xem là *ký tự thoát (escaping character)*

Dấu backslash (\) được dùng để bỏ qua ý nghĩa sử dụng của ký tự đứng sau nó, vì nhiều ký tự đặc biệt được thiết kế sẵn để phục vụ một chức năng cụ thể nào đó. Ví dụ, nếu ta muốn cho hiển thị ký tự backslash, đây là một ký tự đặc biệt nên ta phải loại bỏ ý nghĩa của nó.

d. Boolean - - Nó chỉ có thể nhận hai giá trị: True hoặc False. Kiểu dữ liệu này rất hữu dụng khi thực hiện các quyết định hoặc so sánh dữ liệu.

e. null - - Kiểu null chỉ có duy nhất một giá trị.: null. Null ngụ ý là không có dữ liệu.
– Nó thực hiện chức năng là giữ chỗ trong một biến với ngụ ý là ở đó không có dữ liệu gì.

Số không hay một chuỗi rỗng và null là các giá trị khác nhau.

6.5 CÁC TOÁN TỬ

Các toán tử thực hiện tính toán trên một hoặc nhiều biến hoặc giá trị (toán hạng) và trả về một giá trị mới. Ví dụ: toán tử '+' có thể cộng hai số để cho ra một số thứ ba. Các toán tử được dùng trong các biểu thức với các giá trị liên quan đến nhau--nhằm thực hiện các phép toán hoặc so sánh các giá trị.

JavaScript sử dụng cả các toán tử hai ngôi và các toán tử một ngôi.

- Toán tử hai ngôi cần hai toán hạng. Ví dụ:

`4 + 5`

trong đó 4 và 5 là các toán hạng và + là toán tử

- Toán tử một ngôi đòi hỏi phải có một toán hạng. Ví dụ:

`a++ or b++`

Các toán tử được phân loại tùy thuộc vào chức năng mà chúng thực hiện:

6.5.1 Toán tử số học

Các toán tử số học sử dụng các giá trị số (literals hay các biến) làm toán hạng của chúng và trả về một giá trị số. Các toán tử số học cơ bản là:

Toán tử	Mô tả	Ví dụ
+	Phép cộng	$A = 5 + 8$
-	Phép trừ	$A = 8 - 5$
/	Phép chia	$A = 20 / 5$
%	Phép chia lấy số dư	$10 \% 3 = 1$
++	Tăng lên một đơn vị. Toán tử này nhận một toán hạng. Giá trị của toán hạng sẽ tăng lên 1 đơn vị . Giá trị được trả về sẽ tùy thuộc vào toán tử ++ nằm sau hay nằm trước toán hạng.	++x sẽ trả về giá trị của x sau khi tăng. x++ sẽ trả về giá trị của x trước khi tăng.
--	Giảm một đơn vị. Toán tử này nhận một toán hạng. Giá trị được trả về tùy thuộc vào toán tử -- nằm trước hay nằm sau toán hạng.	--x sẽ trả về giá trị của x sau khi giảm. x-- sẽ trả về giá trị của x trước khi giảm. .
-	Lấy số đối. Nó sẽ trả về số đối của toán hạng	Nếu a là 5, thì $-a = -5$.

Ví dụ 9:

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE="JavaScript">
    var x = 10;
    alert ("The value of x = " + x);
```

```

    alert ("The value of x + x = " + (x + x));
    alert ("The value of x - x = " + (x - x));
    alert ("The value of x * x = " + (x * x));
    alert ("The value of x / 3 = " + (x / 3));
    alert ("The value of x % 3 = " + (x % 3));
    alert ("The value of --x = " + (--x));
    alert ("The value of ++x = " + (++x));
    alert ("The value of -x = " + (-x));

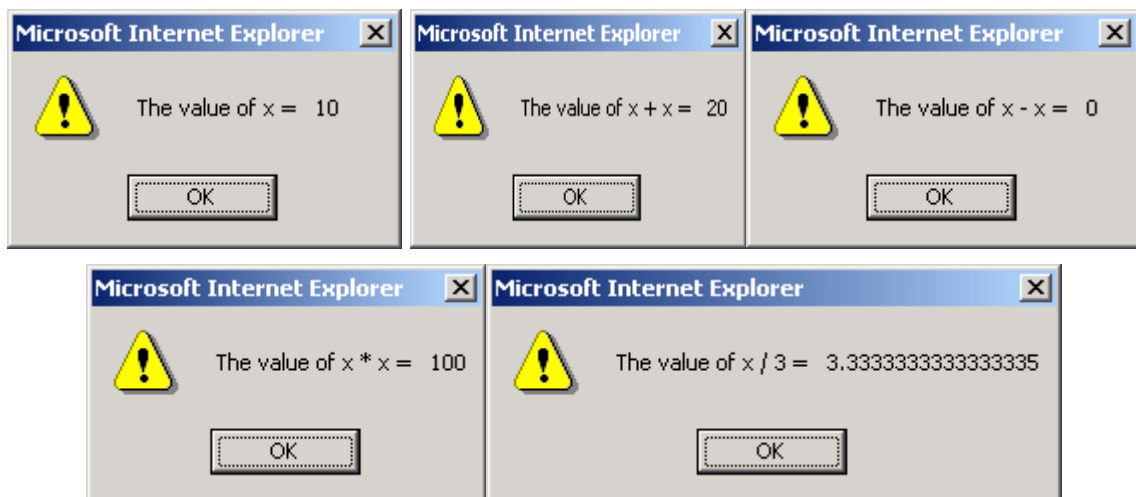
</SCRIPT>

</HEAD>

</HTML>

```

Dưới đây là một số hộp thông báo:



Hình 6.10: JavaScript: Các toán tử số học

6.5.2 Toán tử so sánh

Toán tử so sánh thường so sánh các toán hạng của nó và trả về một giá trị logic dựa trên phép so sánh đó đúng hay không. Các toán hạng có thể là các giá trị số hoặc chuỗi. Khi so sánh chuỗi, nó sẽ dựa trên thứ tự các kí tự trong bảng chữ cái.

Toán tử	Mô tả	Ví dụ
==	Bằng. Trả về giá trị true nếu các toán hạng bằng nhau.	a == b
!=	Không bằng. Trả về giá trị true nếu các toán hạng không bằng nhau.	Var2 != 5
>	Lớn hơn. Trả về giá trị true nếu toán hạng trái lớn hơn toán hạng phải.	Var1 > var2
>=	Lớn hơn hoặc bằng. Trả về giá trị true nếu toán hạng trái lớn hơn hoặc bằng toán hạng phải.	Var1 >= 5 Var1 >= var2

<	Nhỏ hơn. Trả về giá trị true nếu toán hạng trái nhỏ hơn toán hạng phải.	Var2 < var1
<=	Nhỏ hơn hoặc bằng. Trả về giá trị true nếu toán hạng trái nhỏ hơn hoặc bằng toán hạng phải.	Var2 <= 4 Var2 <= var1

6.5.3 Toán tử logic

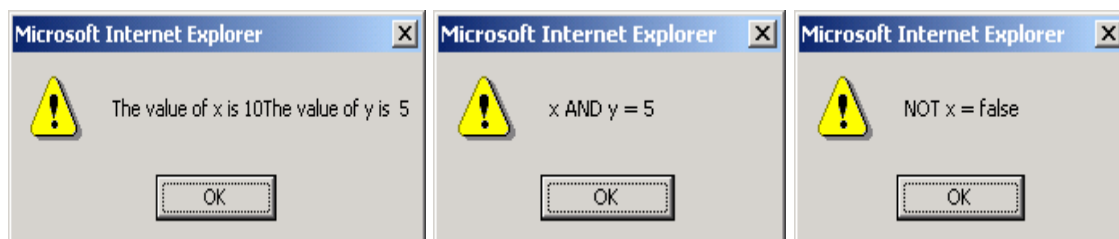
Toán tử logic được dùng để kết hợp nhiều toán tử so sánh thành một biểu thức điều kiện. Một toán tử logic thường có hai toán hạng, mỗi một toán hạng đó đều có giá trị true hoặc false và trả về kết quả true hoặc false. Ví dụ, có thể chúng ta muốn kiểm tra xem (total>100) AND (StateTax=true).

Toán tử	Giá trị	Mô tả
And (&&)	expr1 && expr2	Trả về giá trị của expr1 nếu nó là false. Nếu không thì nó trả về giá trị của expr2.
Or ()	expr1 expr2	Trả về giá trị của expr1 nếu nó là true. Nếu không thì nó sẽ trả về giá trị của expr2.
Not (!)	!expr	Trả về giá trị false nếu biểu thức đúng và trả về giá trị true nếu biểu thức sai.

Ví dụ 10:

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript">
      var x = 10;
      var y = 5;
      alert ("The value of x is " + x + "The value of y is  " + y);
      alert("x AND y = " + (x && y));
      alert("x OR y = " + (x || y));
      alert("NOT x = " + (!x));
    </SCRIPT>
  </HEAD>
</HTML>
```

Kết quả:



Hình 6.11: JavaScript: Toán tử logic**6.5.4 Toán tử chuỗi**

Toán tử chuỗi nhận hai toán hạng và tạo ra một chuỗi mới bằng cách ghép hai chuỗi ban đầu lại với nhau.

Ví dụ:

```
x = 'yellow';
y = 'green';
z = x + y + 'white'; which means z is "yellowgreenwhite"
w = y + 9, which means w is "green9"
```

6.5.4 Toán tử Evaluation

Một số toán tử ít sử dụng trong JavaScript và không được xếp vào loại cụ thể nào. Những toán tử này được liệt kê dưới đây:

Toán tử điều kiện

(condition) ? trueVal : falseVal gán một giá trị xác định cho một biến nếu điều kiện đúng, và ngược lại gán một giá trị khác nếu điều kiện là false. Ví dụ:

```
status = (age >= 18) ? "adult" : "minor"
```

Nếu tuổi lớn hơn hoặc bằng 18, biến status được gán giá trị “adult”. Nếu không, nó sẽ có giá trị “minor”.

Trong ví dụ sau đây, phương thức getSeconds() được dùng để lấy số giây từ đối tượng Date và gán nó với biến “seconds”. Nếu giá trị “seconds” lớn hơn hoặc bằng 3 hoặc nhỏ hơn hoặc bằng 50, thì màu nền của tài liệu chuyển sang màu đỏ (red). Ngược lại màu nền là màu xanh lục (green).

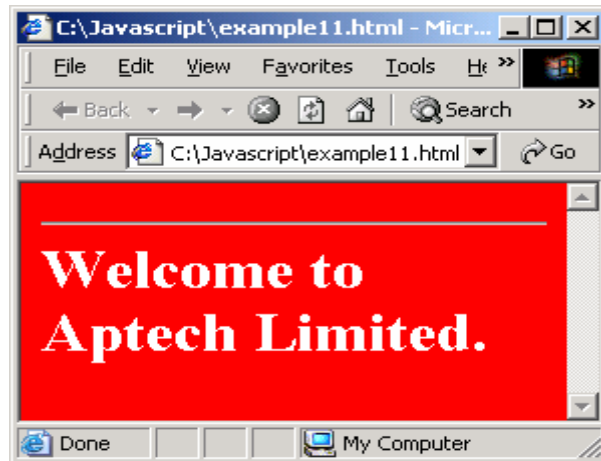
Ví dụ 11:

```
<html>
  <head>
    <script language="JavaScript">
      <!--
        var todays_date = new Date();
        var seconds = todays_date.getSeconds();
        var b_color = (seconds >=3 && seconds <=50) ?
                      "Red" : "Green";

        document.write("<body    text=White    bgcolor="    +    b_color    +
">");
      //-->
    </script>
  </head>
  <h1>
  <hr>
```

```
Welcome to Aptech Limited.  
</hr> </h1>  
</body>  
</html>
```

Kết quả:



Hình 6.12: JavaScript: Toán tử Logic

➤ **typeof**

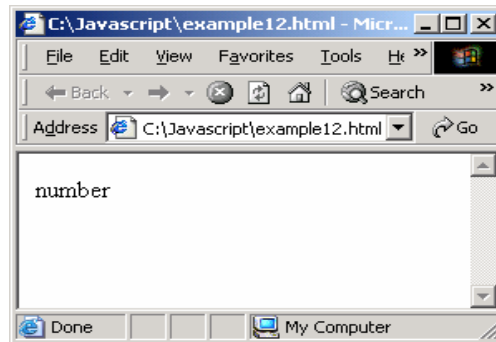
Toán tử typeof trả về chuỗi cho biết tên kiểu dữ liệu của toán hạng

Ví dụ 12:

```
<html>  
  <head>  
    <script language="JavaScript">  
      <!--  
      var x = 5;  
      document.write(typeof(x));  
      //-->  
    </script>  
  </head>  
</html>
```

Toán tử typeof sẽ trả về: number

Kết quả:



Hình 6.13: JavaScript: Toán tử typeof

6.5.5 Mức ưu tiên của các toán tử

Khi có nhiều toán tử trong cùng một biểu thức, độ ưu tiên của toán tử xác định thứ tự thực hiện của toán tử đó. Một biểu thức được đọc từ trái sang phải và được thực hiện từ các toán tử có độ ưu tiên cao đến các toán tử có độ ưu tiên thấp hơn. Nếu chúng ta muốn thay đổi trật tự thực hiện của các toán tử, ta phải sử dụng các dấu ngoặc. Bảng dưới đây liệt kê độ ưu tiên của các toán tử từ thấp đến cao:

Kiểu toán tử	Các toán tử đơn
Gán	= += -= *= /= %= <<= >>= >>>= &= ^= =
Điều kiện	?:
logic -or	
logic -and	&&
bitwise-or	
bitwise-xor	^
bitwise-and	&
bằng/không bằng	== !=
quan hệ	< <= > >=
dịch bit	<< >> >>>
cộng/trừ	+ -
nhân/chia	* / %
phủ định/tăng	! ~ - ++ -- typeof void

6.6 MẢNG

Có những lúc ta muốn lưu nhiều giá trị vào trong một biến. Khi đó ta sử dụng mảng. Mảng được dùng để lưu một tập hợp các biến có cùng tên. Chỉ số của mảng dùng để phân biệt các biến này. Trong JavaScript chỉ số của mảng bắt đầu từ 0.

Tuy nhiên, JavaScript không có kiểu dữ liệu mảng. Nhưng nó có một đối tượng mảng được xây dựng sẵn. Để dùng mảng trong chương trình, ta phải sử dụng đối tượng mảng và các phương thức của nó.

Tạo mảng

Cú pháp sau đây dùng để tạo một mảng:

```
arrayObjectName = new Array([element0, element1, ..., elementN])
```

Trong đó **arrayObjectName** là tên của đối tượng mảng, và **elementN** là danh sách các giá trị của các phần tử mảng. Mảng được khởi tạo với các giá trị xác định như là các phần tử của nó. Thuộc tính `length` lưu số lượng các phần tử có trong mảng.

Thêm các phần tử

Chúng ta có thể đưa các phần tử vào một mảng ngay khi chúng ta tạo nó. Chúng ta cũng có thể đưa dữ liệu vào một mảng bằng cách gán giá trị cho các phần tử của nó. Ví dụ,

```
emp[0] = "Ryan Dias"
emp[1] = "Graham Browne"
emp[2] = "David Greene"
```

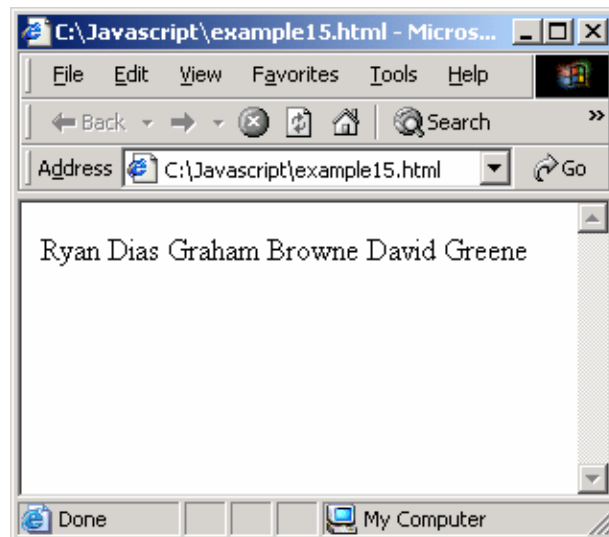
Ryan Dias	Graham Browne	David Greene
-----------	---------------	--------------

Đoạn mã trên tạo ra một mảng và gán giá trị cho từng phần tử của nó.

Ví dụ 15:

```
<html>
  <head>
    <script language="JavaScript">
      <!--
emp = new Array(3)
emp[0] = "Ryan Dias"
emp[1] = "Graham Browne"
emp[2] = "David Greene";
document.writeln(emp[0]);
document.writeln(emp[1]);
document.writeln(emp[2]);
      //-->
    </script>
  </head>
</html>
```

Kết quả:



Hình 6.16: JavaScript: Mảng

Truy cập các phần tử mảng

Có hai cách để truy cập các phần tử của mảng. Chúng ta có thể:

- Chỉ ra tên của phần tử. Ví dụ, `emp ["Ryan Dias"]`
- Chỉ ra chỉ số của phần tử. Ví dụ, `emp [0]`.

Sử dụng các phương thức của mảng

Để thao tác trên mảng, chúng ta phải dùng các phương thức của đối tượng mảng.

Phương thức	Mô tả
Join	Kết hợp các phần tử của mảng thành một chuỗi
Pop	Trả về phần tử cuối cùng của mảng, sau khi xoá nó từ mảng.
Push	Thêm một hoặc nhiều phần tử vào cuối mảng. Trả lại phần tử cuối cùng thêm vào.
Reverse	Đảo ngược các phần tử của mảng: phần tử đầu tiên của mảng trở thành phần tử cuối cùng và phần tử cuối cùng trở thành phần tử đầu tiên.
Shift	Xoá phần tử đầu tiên của mảng và trả về phần tử đó.
Sort	Sắp xếp các phần tử của mảng.

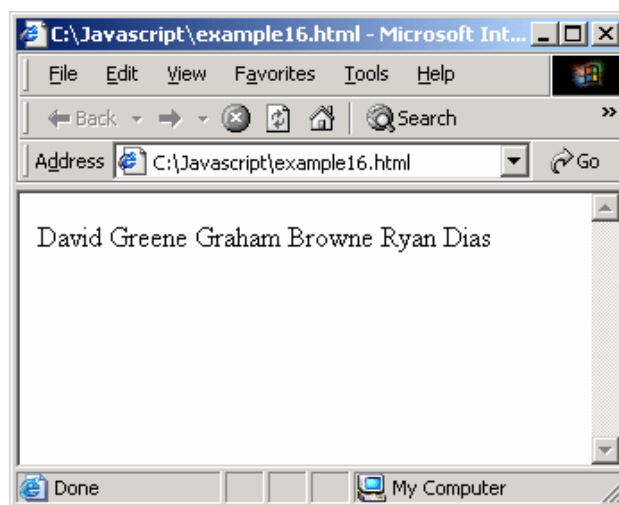
- **Phương thức sort**

Phương thức `emp.sort()` sẽ sắp xếp mảng dưới đây

Ví dụ 16:

```
<html>
  <head>
    <script language="JavaScript">
      <!--
        emp = new Array(3)
        emp[0] = "Ryan Dias"
        emp[1] = "Graham Browne"
        emp[2] = "David Greene";
        emp.sort();
        document.writeln(emp[0]);
        document.writeln(emp[1]);
        document.writeln(emp[2]);
      //-->
    </script>
  </head>
</html>
```

Kết quả:



Hình 6.17: JavaScript: Sắp xếp các mảng (Sorting Arrays)

Mảng nhiều chiều

Một mảng có thể có nhiều hơn một chiều. Ví dụ, ta có thể tạo ra một mảng hai chiều để lưu trữ mã nhân viên và tên của nhân viên đó.

The diagram illustrates a 2D array structure. A horizontal double-headed arrow at the top is labeled "Second Dimension". A vertical double-headed arrow on the left is labeled "First Dimension Array Index". The array is represented as a table with 4 rows and 3 columns. The first column contains indices 0, 1, 2, and 3. The second column contains employee names: Ryan Dias, Mike Donne, Joe Dean, and Robert Matey. The third column contains codes: 001, 002, 003, 004, and 005. Arrows point from the labels "Employee" and "Code" to the respective columns.

	0	001
0	Ryan Dias	002
1	Mike Donne	003
2	Joe Dean	004
3	Robert Matey	005

Hình 6.18: Mảng hai chiều

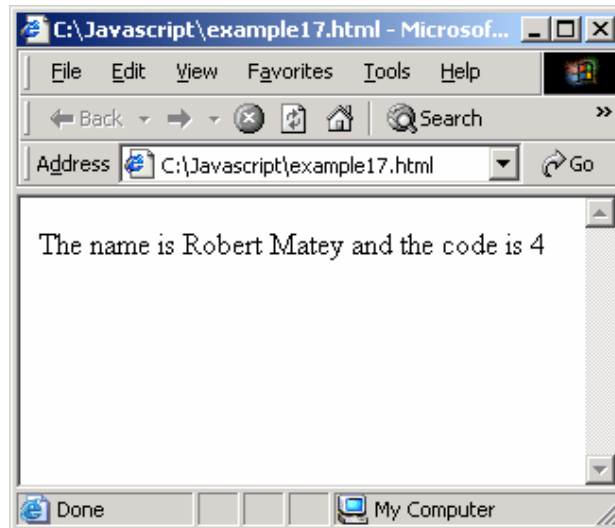
Đoạn mã dưới đây tạo ra một mảng hai chiều và cho hiển thị giá trị của một trong những phần tử trong mảng.

Ví dụ 17:

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE = "JavaScript">
    <!--
    MyArray = new Array(5,5);

    MyArray[0, 0] = "Ryan Dias";
    MyArray[0, 1] = 1;
    MyArray[1, 0] = "Mike Donne";
    MyArray[1, 1] = 2;
    MyArray[2, 0] = "Joe Dean";
    MyArray[2, 1] = 3;
    MyArray[3, 0] = "Robert Matey";
    MyArray[3, 1] = 4;
    document.write ("The name is      " + MyArray[3, 0]);
    document.write("and the code is " + MyArray[3,1]);
    -->
  </SCRIPT>
</HEAD>
</HTML>
```

Kết quả:



Hình 6.19: Mảng hai chiều (Two dimensional Array)

6.7 CÁC CÂU LỆNH ĐIỀU KIỆN

Câu lệnh điều kiện được dùng để kiểm tra một điều kiện. Kết quả của điều kiện xác định câu lệnh hoặc khối lệnh sẽ được thực thi. Trong JavaScript các câu lệnh sau đây được dùng để kiểm tra một điều kiện:

Câu lệnh `if ... else`

Câu lệnh này dùng để kiểm tra điều kiện, nó thực hiện việc tính toán trên một biểu thức. Nếu điều kiện là đúng (true) thì khối lệnh được thực thi.

```
if (condition)
    { statements; }
```

Ta cũng có thể chỉ ra khối lệnh cần thực hiện khi điều kiện là sai (false) bằng việc dùng mệnh đề `else`

```
if (condition)
    { statements; }
else
    { statements2; }
```

Nếu điều kiện là sai khối lệnh sau `else` được thực thi.

Lệnh `switch`

Khi ta có nhiều tùy chọn `If...else` thì tốt hơn nên sử dụng lệnh `switch`. Lệnh này còn được xem là lệnh `case`, lệnh `switch` thực thi một trong các khối lệnh tùy thuộc vào giá trị của biểu thức. Nếu không tìm thấy một giá trị nào trong danh sách các `case` của nó, khối lệnh trong phần `default` sẽ được thực hiện. Lệnh `break` dùng để thoát ra khỏi câu lệnh `switch`.

```
switch (expression){
    case label:
        statement;
        break;
```



```
case label:
    statement;
    break;
...
default: statement;
}
```

Ví dụ,

```
switch (dayofweek) {
case 1:
    statements;
    break;
case 2:
    statements;
    break;
case 3:
    statements;
    break;
case 4:
    statements;
    break;
case 5:
    statements;
    break;
default: statements;
}
```

6.8 CÁC LỆNH VÒNG LẶP

Các cấu trúc điều khiển việc thực hiện lặp đi lặp lại trong một chương trình được gọi là vòng lặp. Có nhiều loại vòng lặp:

- Vòng lặp thực hiện lặp đi lặp lại các lệnh cho đến khi điều kiện là False
- Vòng lặp thực hiện lặp đi lặp lại các lệnh cho đến khi điều kiện là True
- Vòng lặp thực hiện lặp đi lặp lại các lệnh theo một số lần nhất định

Vòng lặp “for”

Vòng lặp for sẽ thực hiện lặp đi lặp lại khối lệnh cho đến khi điều kiện là false. Số lần thực hiện của vòng lặp thường được điều khiển thông qua một biến đếm.

Lệnh for bao gồm ba phần, cách nhau bởi dấu chấm phẩy. Cả ba phần đó đều không bắt buộc phải có, và chúng điều khiển việc thực hiện của vòng lặp for.

```
for (lệnh khởi tạo; điều kiện; lệnh tăng)
```

```
{ statements; }
```

Lệnh khởi tạo -- được thực hiện duy nhất một lần và thường dùng để khởi tạo biến đếm

Lệnh điều kiện -- điều kiện của vòng lặp.

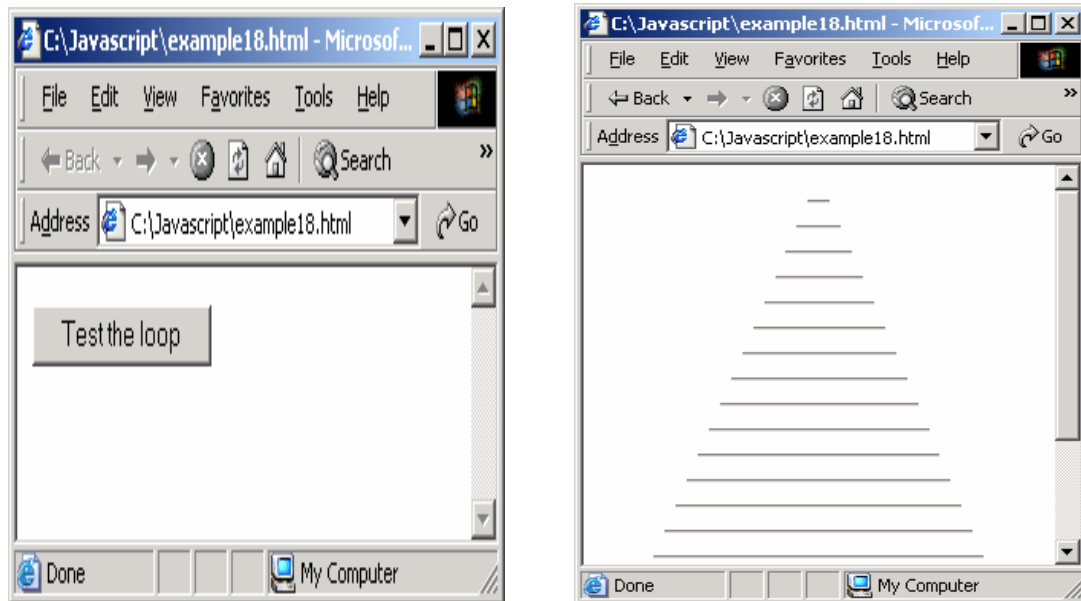
Lệnh tăng -- thay đổi biến đếm trong vòng lặp.

Trong ví dụ trên, khi người dùng nhấp chuột ở một nút, sự kiện onclick sẽ được kích hoạt. Hàm “doloop()” được gọi bởi sự kiện onclick. Hàm doloop đặt “align=center” và gán giá trị cho thuộc tính width của thẻ <HR>. Phương thức open dùng để mở một tài liệu. Vòng lặp for sử dụng biến đếm size và khởi tạo cho nó giá trị 5. Mỗi lần vòng lặp được thực hiện, giá trị của biến size được tăng lên 5. Vòng lặp được thực hiện đến khi size có giá trị lớn hơn 1000. Trong thân vòng lặp, biểu thức String1 được thực hiện và một đường kẻ ngang với chiều rộng xác định được in ra màn hình. Biểu thức String1 có dạng như sau: <HR align=center width=size%>

Ví dụ 18:

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE = "JavaScript">
    function doloop() {
      var String1 = '<hr align="center" width=';
      document.open();
      for (var size = 5; size <= 100; size+=5)
        document.writeln(String1 + size + '%">');
      document.close();
    }
  </SCRIPT>
</HEAD>
<BODY>
  <FORM>
    <INPUT type="button"
      value="Test the loop"
      onclick = "doloop()" >
  </FORM>
</BODY>
</HTML>
```

Output:



Hình 6.20: Vòng lặp for

Do ... while

Vòng lặp Do...while được dùng để thực thi một khối lệnh cho đến khi điều kiện là false. Cú pháp là:

```
do
{ statements;}
while (condition)
```

Lệnh while

Lệnh while là một cấu trúc lặp khác trong JavaScript. Nó được dùng để thực hiện một khối các câu lệnh chừng nào điều kiện là true. Nếu có nhiều câu lệnh thực hiện trong thân của vòng lặp chương trình phải sử dụng cặp dấu { và } để chứa các câu lệnh trong đó.

Khác biệt chính giữa vòng lặp while và do...while là các lệnh trong thân vòng lặp while có thể không được thực hiện một lần nào vì có thể ngay từ ban đầu điều kiện đã là false. Tuy nhiên vòng lặp do...while bao giờ cũng được thực hiện ít nhất một lần. Cú pháp là:

```
while (condition)
{ statements; }
```

Câu lệnh break & continue

Vòng lặp while loop và for sẽ kết thúc thực hiện khi điều kiện là false. Tuy nhiên ta cũng có thể kết thúc vòng lặp nếu muốn. Lệnh break dùng để kết thúc việc thực thi của một câu lệnh. Khi được sử dụng trong một vòng lặp, lệnh break làm dừng ngay vòng lặp đó và không thực hiện thêm nữa.

Một lệnh đặc biệt khác cũng có thể được sử dụng trong vòng lặp là lệnh continue. Continue dừng ngay lần lặp hiện tại và quay lại kiểm tra điều kiện để thực hiện lần lặp tiếp theo.

➤ **for...in**

Câu lệnh `for .. in` được dùng để duyệt các thuộc tính của một đối tượng hay các phần tử của một mảng. Ví dụ, chúng ta có thể muốn thực hiện một khối các câu lệnh cho mỗi phần tử của mảng.

Chú pháp là:

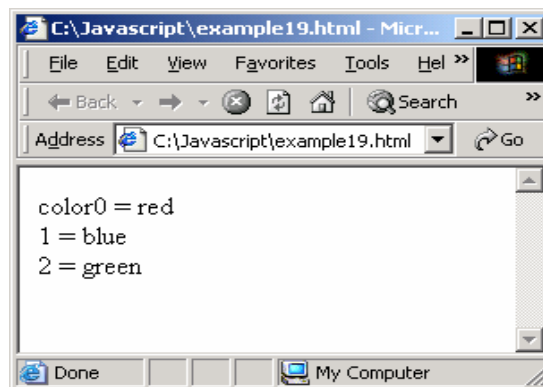
```
for (variable in object)
    { statements; }
```

Trong ví dụ dưới đây, một mảng “color” được tạo. Các phần tử của mảng là “red”, “blue” và “green”. Vòng lặp `for .. in` được dùng để duyệt qua mảng màu và hiển thị các phần tử trong nó.

Ví dụ 19:

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE = "JavaScript">
    color = new Array("red", "blue", "green");
    var record = "color";
    for (var prop in color)
    {record += prop + " = " + color[prop] + "<BR>"}
    record += "<br>"
    document.write(record)
  </SCRIPT>
</HEAD>
</HTML>
```

Kết quả:



Hình 6.21: For..in loop

➤ **with**

Câu lệnh “with” được dùng để thực thi một tập các lệnh cùng tham chiếu đến một đối tượng xác định. Đó là đối tượng được chỉ ra trong câu lệnh « with »

Chú pháp:

```
with (object)
{ statements; }
```

Đối tượng Math là một đối tượng được xây dựng sẵn trong JavaScript. Nó gồm các thuộc tính và phương thức thể hiện các hàm toán học chuẩn.

6.9 HÀM (FUNCTION)

JavaScript có nhiều hàm định nghĩa trước mà chúng ta sẽ dùng trong script. Ví dụ, nếu chúng ta muốn kiểm tra người dùng có phải nhập vào một số hay không, chúng ta có thể sử dụng hàm isNaN (Not a Number) để thực hiện điều này. Trong phần này chúng ta sẽ xem xét một vài hàm JavaScript định nghĩa sẵn.

➤ Hàm eval

Hàm eval được dùng để đánh giá một chuỗi và không cần tham chiếu đến bất kỳ một đối tượng cụ thể nào.

eval(string)

Chuỗi có thể là một biểu thức JavaScript, câu lệnh, hoặc một nhóm các câu lệnh. Trong biểu thức có thể bao gồm các biến và thuộc tính của một đối tượng.

Nếu chuỗi được xác định là một biểu thức, hàm eval sẽ tính toán biểu thức. Nếu nó là một hoặc một tập các lệnh, nó sẽ thực thi các câu lệnh. Ví dụ, chúng ta tạo biểu thức theo mẫu như chuỗi sau - - "3 * 5". Chúng ta có thể dùng eval để tính toán ra kết quả.

➤ Hàm isNaN

Hàm isNaN được dùng để kiểm tra xem đối số truyền vào có phải là một số hay không.

```
isNaN(testValue)
```

Tạo hàm do người dùng định nghĩa

Một hàm JavaScript khá giống với «thủ tục» ("procedure") hay « chương trình con » ("subroutine") trong ngôn ngữ lập trình. Một hàm là một tập các câu lệnh, thực hiện các tác vụ xác định. Chúng ta có thể truyền các giá trị (tham số) cho hàm. Hàm cũng có thể trả về một giá trị.

Các thành phần của một hàm là:

- Từ khoá function.
- Tên hàm.
- Danh sách các đối số của hàm cách nhau bởi dấu phẩy (,) đặt trong cặp ngoặc đơn ().
- Các câu lệnh của hàm đặt trong cặp dấu móc {}.

```
function funcName(argument1, argument2, etc)
{ statements; }
```

Một hàm có thể không có đối số, nhưng chúng ta vẫn phải có cặp dấu ngoặc đơn ().

```
funcName ( )
```

Các hàm không thể lồng nhau.

Định nghĩa hàm

Trước khi chúng ta gọi hàm, chúng ta phải định nghĩa nó trong chương trình. Định nghĩa một hàm là một quá trình khai báo tên của hàm và các lệnh sẽ được thực thi khi gọi hàm.

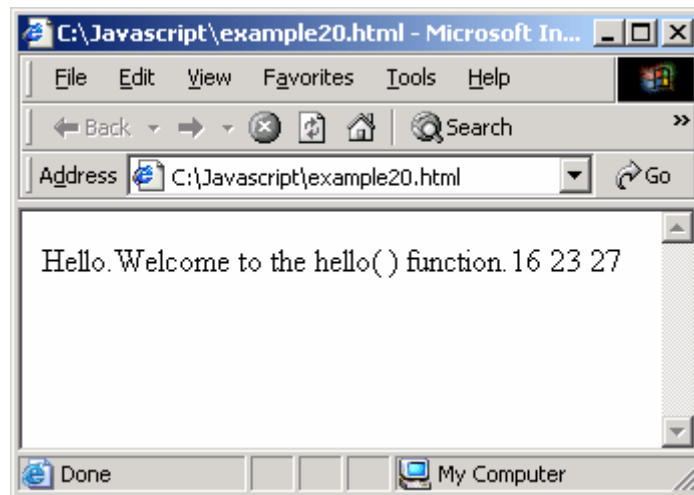
Gọi một hàm

Để thực thi một hàm, ta phải gọi nó. Để gọi một hàm ta chỉ ra tên hàm và danh sách các tham số nếu có:

Ví dụ 20:

```
<html>
  <head>
    <script language="JavaScript">
      <!--
      function hello( )
      {
        document.write ('Hello. ');
        document.write ('Welcome to the hello( ) function. ');
        return;
      }
      function sum_up ( one, two)
      {
        var result = one + two;
        return result;
      }
      function sum_all ( )
      {
        var loop=0, sum=0;
        for ( loop = arguments.length-1; loop >=0; loop--)
          sum += arguments[loop];
        return sum;
      } // Add it up now
      hello ( );
      var total = sum_up(7, 9);
      document.write ( total + ' ' + sum_up ( 8, 15));
      document.write ( ' ' + sum_all ( 1, 5, 8,7, 6) );
      //-->
    </script>
  </head> </html>
```

Kết quả:



Hình 6.22: Định nghĩa và gọi một hàm

Câu lệnh return

Câu lệnh này được dùng để trả về một giá trị. Dùng lệnh return trong một hàm là không bắt buộc vì không phải tất cả các hàm đều trả về một giá trị cụ thể.

Cú pháp là:

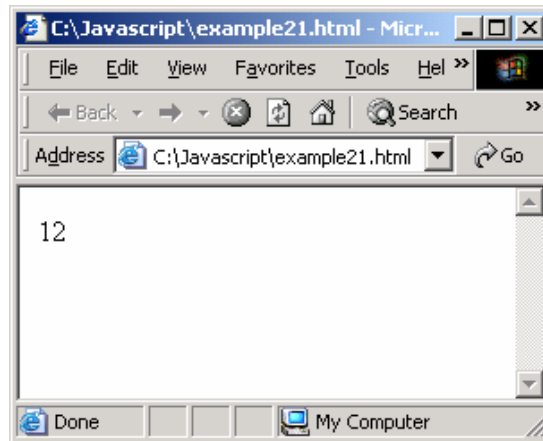
```
return value;  
Or  
return ( value ) ;
```

Ví dụ 21:

```
<html>  
  <head>  
    <script language="JavaScript">  
      function testreturn(x) {  
        var i = 0;  
        while (i < 6) {  
          if (i == 3)  
            break  
          i++  
        }  
        document.write (i*x);  
        return (i*x);  
      }  
    </script>  
  </head>  
<BODY>  
  <SCRIPT>
```

```
testreturn(4);  
</SCRIPT>  
</BODY>  
</html>
```

Kết quả:



Hình 6.23: Giá trị trả về hàm

Tại bất kì lúc nào, chúng ta cũng có thể thoát ra khỏi hàm đơn giản chỉ cần sử dụng lệnh `return` mà không cần đến bất kì cấu trúc phức tạp nào. Quyển điều khiển ngay lập tức trả về cho câu lệnh đứng sau lệnh gọi hàm.

CHƯƠNG 7: CÁC ĐỐI TƯỢNG CƠ BẢN TRONG JAVASCRIPT

Kết thúc chương này, bạn có thể:

- Làm việc trên các đối tượng cơ bản
- Sử dụng các thuộc tính và phương thức của đối tượng

7.1 GIỚI THIỆU

Khi tạo ra một trang web chúng ta có thể sử dụng các đối tượng do trình duyệt, ngôn ngữ kịch bản và các thành phần HTML cung cấp. Trong chương này, chúng ta cùng thảo luận về những đối tượng khác nhau có thể được chèn vào trong một tài liệu HTML.

7.2 CÁC ĐỐI TƯỢNG JAVASCRIPT

Một đối tượng là một gói dữ liệu toàn diện. Các thuộc tính (biến) dùng để định nghĩa đối tượng và các phương thức (hàm) tác động tới dữ liệu đều nằm trong đối tượng. Ví dụ như, một chiếc xe hơi là một đối tượng. Các thuộc tính của chiếc xe hơi là cấu tạo, kiểu dáng và màu sắc của nó. Hầu hết các chiếc xe hơi đều có một vài phương thức chung như `go()`, `brake()`, `reverse()`.

```
carobj.make = "Fiat"
carobj.model = "Uno"
carobj.color = "red"
```

Để truy cập các thuộc tính của đối tượng, chúng ta phải chỉ ra tên đối tượng và thuộc tính của nó:

`objectName.propertyName`

Ví dụ:

```
Document.bgcolor
```

Trong đó: `bgcolor` (background color) là thuộc tính của đối tượng `document`.

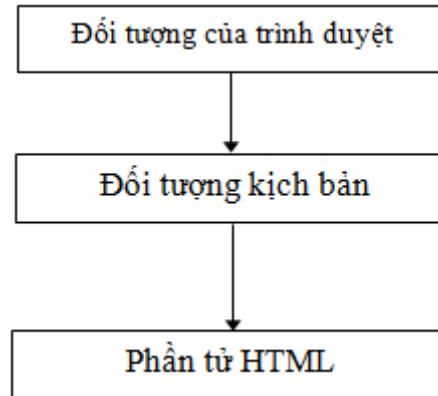
Để truy cập các phương thức của một đối tượng, chúng ta phải chỉ ra tên đối tượng và phương thức yêu cầu:

```
objectName.method()
```

Khi tạo ra một trang web chúng ta có thể chèn:

- Các đối tượng của trình duyệt
- Các đối tượng có sẵn (thay đổi phụ thuộc vào ngôn ngữ kịch bản được sử dụng).
- Các phần tử HTML

Dĩ nhiên, chúng ta có thể tạo ra các đối tượng để sử dụng theo yêu cầu của mình.



Hình 7.1: Cây phân cấp đối tượng

Khi tài liệu HTML được hiển thị trong trình duyệt, một cây phân cấp đối tượng được tạo ra dựa trên các phần tử trong trang. Các đối tượng trình duyệt chẳng hạn như văn bản, cửa sổ, khung, vị trí, ... nằm trên cùng của cây phân cấp đối tượng. Sau đó là các đối tượng JavaScript. Đây là các đối tượng được cung cấp bởi JavaScript chẳng hạn như đối tượng Date, Array, ... Các phần tử HTML nằm ở sau cùng và chính là các thẻ HTML tạo nên văn bản hiển hành.

Sự hiểu biết về cây phân cấp này rất quan trọng bởi vì các đối tượng được truy cập theo sự phân cấp. Ví dụ như, để truy cập một đối tượng form bạn cần phải chỉ ra tên form và đối tượng chứa trong đó như văn bản, câu lệnh sẽ là:

```
document.form1
```

Để truy cập các thuộc tính của một phần tử văn bản, Text1 trong một đối tượng form:

```
document.form.text1.value = "Having fun"
```

7.2.1 Câu lệnh *This*

Câu lệnh '**this**' không chỉ là một thuộc tính nội tại. Giá trị của nó chỉ ra đối tượng hiện hành và có thể có các thuộc tính chuẩn chẳng hạn như tên, độ dài, và giá trị được áp dụng phù hợp. Câu lệnh 'this' chỉ được dùng trong phạm vi của một hàm hay các tham chiếu khi gọi hàm.

```
this[property]
```

Nếu nó không có đối số thì nó sẽ thông qua đối tượng hiện hành. Tuy nhiên, chúng ta nên gán vào một thuộc tính hợp lệ để đưa ra kết quả.

Ví dụ 1:

```

<HTML>
  <HEAD>
    <script language="JavaScript">
      function dispname(name) {
        alert("Welcome to the world of JavaScript, " + name);
      }
    </script>
  </HEAD>
  
```

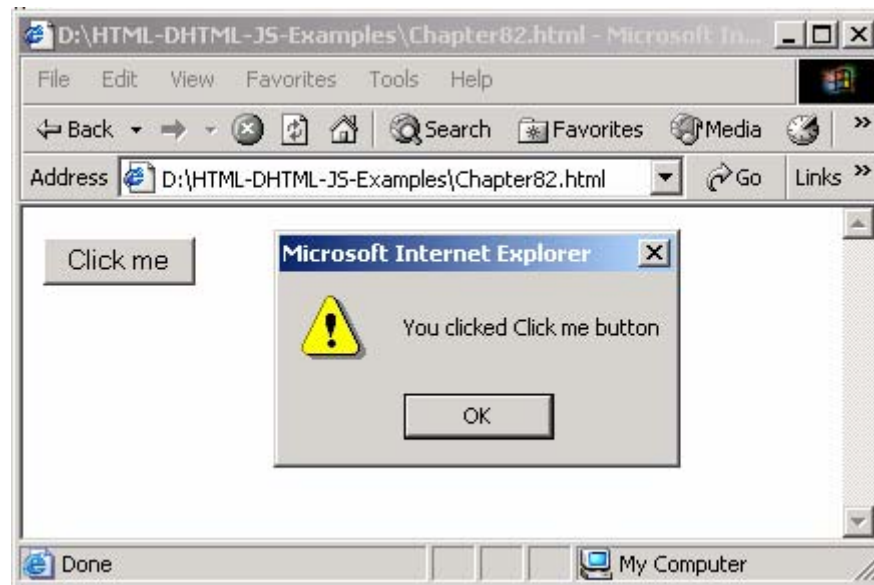
```
<FORM>
    <B>Enter your name:</B>
    <INPUT TYPE = "text" NAME = "text1" SIZE = 20
onChange="dispname(this.form.text1.value)">
</FORM>
</HTML>
```



Hình 7.2: Kết quả của ví dụ 1

Ví dụ 2:

```
<html>
    <head>
        <script language="JavaScript">
            function show(value) {
                alert("You clicked " + value + " button");
            }
        </script>
    </head>
    <FORM>
        <B>Enter your name:</B>
        <INPUT TYPE = "button" NAME = "button1" value = "Click me" SIZE
        = 20 onClick="show(this.form.button1.value)">
    </FORM>
</html>
```



Hình 7.3: Kết quả của ví dụ 2

7.2.2 Đối tượng String

Đối tượng string được dùng để thao tác và làm việc với chuỗi văn bản. Chúng ta có thể tách nó ra thành các chuỗi con và biến đổi chuỗi đó thành các chuỗi hoa hoặc thường trong một chương trình. Cú pháp tổng quát là:

```
stringName.propertyName
```

Hoặc

```
stringName.methodName
```

Có 3 phương thức khác nhau để tạo ra chuỗi.

Dùng lệnh var và gán cho nó một giá trị. Ví dụ như

```
var newstr = " This is my script"
```

Dùng một toán tử (=) có gán với một tên biến. Ví dụ như:

```
newstr = "This is my script"
```

Dùng hàm khởi tạo String (string). Ví dụ như:

```
var newstr = String ("This is my script")
```

Các thuộc tính:

Tên	Mô tả
length	Trả lại độ dài của chuỗi

Phương thức:

Tên	Mô tả
Big	Tăng kích thước của chuỗi văn bản
Blink	Tạo hiệu ứng nhấp nháy cho chuỗi (Internet Explorer không hỗ

	trợ phương thức này)
Bold	In đậm chuỗi
Fontcolor	Xác định màu của font chữ
Italics	Hiển thị chuỗi ở dạng in nghiêng
Small	Giảm kích thước của chuỗi văn bản
Strike	Hiển thị chuỗi có đường gạch ngang nằm giữa (strikethrough)
Sub	Hiển thị văn bản dưới dạng chỉ số dưới
Sup	Hiển thị văn bản dưới dạng chỉ số trên
ToLowerCase	Chuyển chuỗi thành ký tự thường
ToUpperCase	Chuyển chuỗi thành ký tự hoa

Ví dụ dưới đây hiển thị một vài phương thức và công dụng của chúng:

Ví dụ 7:

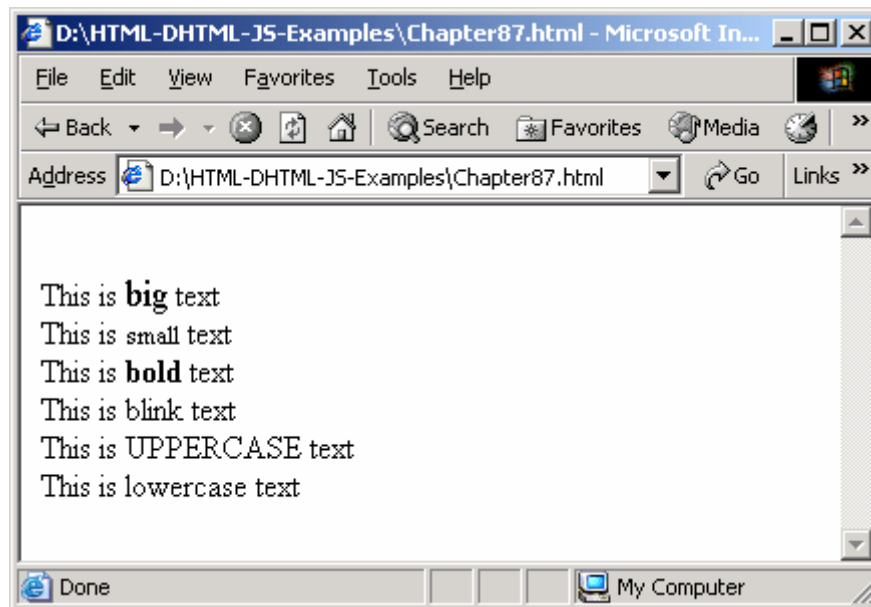
```
<HTML>
```

```
<HEAD>
```

```
  <script language = "Javascript">
    var bstr = "big";
    var sstr = "small";
    var blstr = "bold";
    var blkstr = "blink"
    var ucase = "Uppercase";
    var lcase = "Lowercase";
    document.write ("<BR>This is "+ bstr.big() + " text");
    document.write ("<BR>This is "+ sstr.small() + " text");
    document.write ("<BR>This is "+ blstr.bold() + " text");
    document.write ("<BR>This is "+ blkstr.blink() + " text");
    document.write ("<BR>This is "+ ucase.toUpperCase() + " text");
    document.write ("<BR>This is "+ lcase.toLowerCase() + " text");
  </script>
```

```
</HEAD>
```

```
</HTML>
```



Hình 7.8: Kết quả của ví dụ 7

7.2.3 Đối tượng Math

Đối tượng Math có các thuộc tính và phương thức biểu thị các phép tính toán học nâng cao.

Thuộc tính:

Thuộc tính	Mô tả
PI	Giá trị của π , bằng khoảng 3.1415.
LN10	Giá trị của lg (logarit cơ số 10), bằng khoảng 2,302
E	Giá trị của hằng số Euler, bằng khoảng 2.718. Hằng số Euler được dùng như số cơ sở cho các ln

Phương thức:

Phương thức Method	Mô tả Description
Abs (number)	Trả về giá trị tuyệt đối của một số
Sin (number)	Trả về giá trị sin của của một số (tính bằng radian)
Cos (number)	Trả về giá trị cosin của của một số (tính bằng radian)
Tan (number)	Trả về giá trị tang của của một số (tính bằng radian)
Min (number1, number2)	Trả về giá trị nhỏ nhất của hai số number1 và number2
Max (number1, number2)	Trả về giá trị lớn nhất của hai số number1 và number2
Round (number)	Làm tròn đối số tới số nguyên gần nhất

Sqrt (number)	Trả về căn bậc hai của một số
---------------	-------------------------------

Ví dụ 8:

<HTML>

```
<SCRIPT LANGUAGE = "JavaScript">
```

```
function doCalc(x)
```

```
{
```

```
{
```

```
var a;
```

```
a = Math.PI * x * x;
```

```
alert ("The area of a circle with a radius of " + x + " " +  
+ "is" + " " + a);
```

```
}
```

```
</SCRIPT>
```

```
<BODY bgColor = white>
```

```
<FORM>
```

```
Enter the radius of the circle :
```

```
<INPUT TYPE = TEXT size = 5 name = "rad">
```

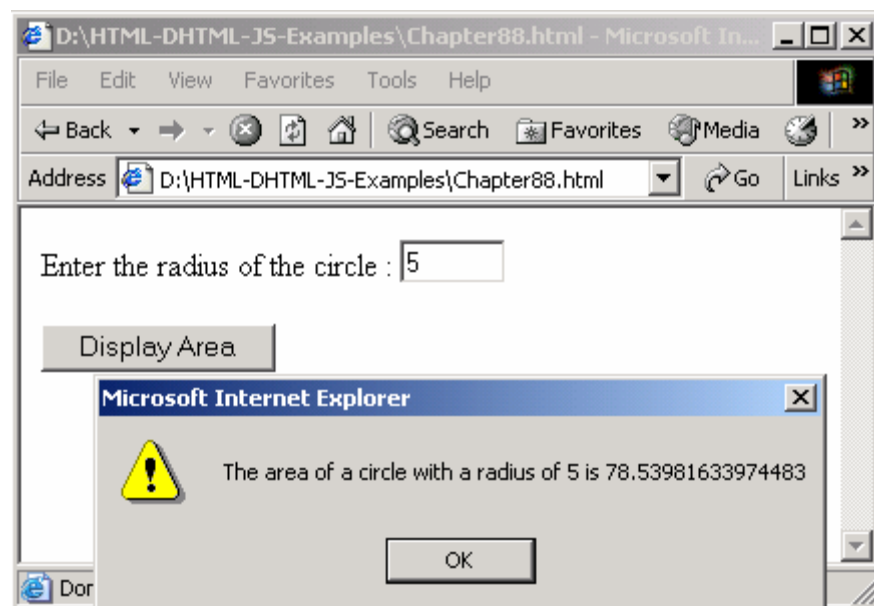
```
<BR><BR>
```

```
<INPUT type = button value = "Display Area"  
onclick="doCalc(rad.value)">
```

```
</FORM>
```

```
</BODY>
```

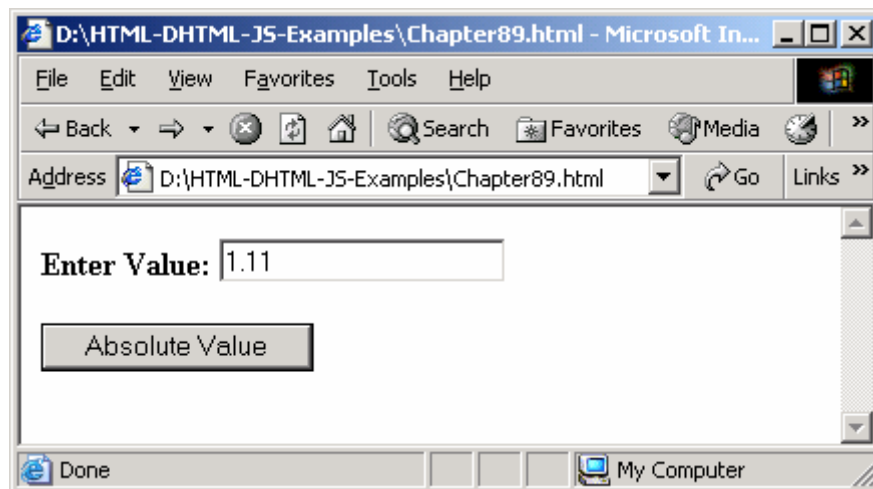
```
</HTML>
```



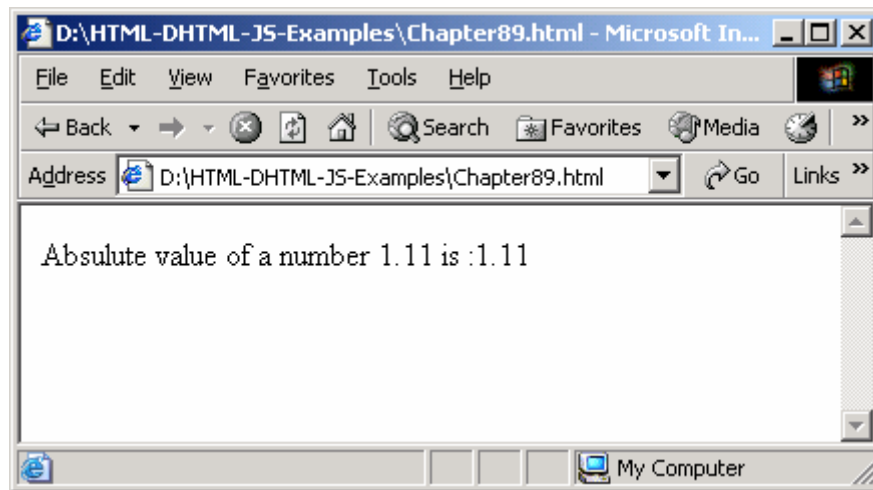
Hình 7.9: Kết quả của ví dụ 8

Ví dụ 9:

```
<html>
<head>
  <script language="JavaScript">
    function show(value)
    {
      with(Math)
        document.write("Absolute value of a number "+ value+ "
        is :"+abs(eval(value)));
    }
  </script>
</head>
<FORM>
  <B>Enter Value:</B>
  <INPUT TYPE = "text" NAME = "text1" SIZE = 20>
  <P>
  <INPUT TYPE = "button" NAME = "button1" value="Absolute Value"
  onClick="show(this.form.text1.value)"
</FORM>
</html>
```



Hình 7.10 a: Kết quả của ví dụ 9



Hình 7.10 b: Kết quả của ví dụ 9 (sau khi nhấn nút lệnh)

7.2.4 Đối tượng Date

Date là một đối tượng có sẵn chứa thông tin về ngày và giờ. Đối tượng Date không có thuộc tính nào. Tuy nhiên, nó có nhiều phương thức dùng để thiết lập, lấy và xử lý các thông tin về thời gian.

Đối tượng Date lưu trữ thời gian theo số mili giây tính từ 1/1/1970 00:00:00.

Tạo một đối tượng Date

```
DateObject = new Date (parameters)
```

Trong đó:

DateObject là một biến lưu trữ một đối tượng thời gian mới.

Parameters có thể là:

- Không có gì – Nếu không có tham số, nó sẽ trả về thời gian hiện tại của hệ thống. Ví dụ: `today = new Date()`.
- String (chuỗi) – biểu diễn ngày giờ theo dạng: “MM DD, YYYY, hh:mm:ss”. Ví dụ:
`tdate = new Date (“ July 29, 1998,10:30:00”)`

Phương thức:

Bảng sau mô tả các nhóm phương thức về thời gian:

Nhóm phương thức	Mô tả
Set	Gồm những phương thức dùng để thiết lập các giá trị thời gian.
Get	Gồm những phương thức được dùng để lấy các giá trị thời gian.
To	Gồm những phương thức được dùng để trả về các chuỗi giá trị từ các đối tượng Date.

parse và UTC	Gồm những phương thức được dùng để phân tích các chuỗi.
--------------	---

Bảng sau dùng các số nguyên để biểu diễn các giá trị:

Các giá trị	Số nguyên
Giây và phút	0 đến 59
Giờ	0 đến 23
Ngày (trong tuần)	0 đến 6
Ngày (trong tháng)	1 đến 31
Tháng	0 đến 11 (tháng một đến tháng mười hai)
Năm	Từ 1900 trở đi

Chúng ta hãy tìm hiểu các nhóm phương thức get, set, to và parse:

➤ **Nhóm phương thức get:**

Phương thức	Mô tả
getDate	Trả về ngày trong tháng từ đối tượng Date (1-31)
getDay	Trả về ngày trong tuần từ đối tượng Date (0-6)
getHours	Trả về giờ từ đối tượng Date (0 – 23)
getMinutes	Trả về phút từ đối tượng Date (0-59)
getSeconds	Trả về giây từ đối tượng Date (0 -59)
getMonth	Trả về tháng từ đối tượng Date (0 – 11)
getYear	Trả về năm từ đối tượng Date (năm (-)1900)
getTime	Trả về số mili giây của thời gian hiện tại (tính từ 1/1/1970)
getTimeZoneOffset	Trả về chênh lệch bằng phút giữa giờ địa phương và giờ chuẩn (GMT)

➤ **Nhóm phương thức set:**

Phương thức	Mô tả
setDate	Thiết lập ngày trong tháng cho đối tượng Date (0 –31)
setHours	Thiết lập giờ cho đối tượng Date (0-23)
setMinutes	Thiết lập phút cho đối tượng Date (0-59)

setSeconds	Thiết lập giây cho đối tượng Date (0-59)
setTime	Thiết lập giá trị thời gian (tính bằng mili giây) cho đối tượng Date
setMonth	Thiết lập tháng cho đối tượng Date (1-12)
setYear	Thiết lập năm cho đối tượng Date, năm phải lớn hơn 1900 (năm (-) 1900)

Nhóm phương thức to:

Phương thức	Mô tả
toGMTString	Chuyển một đối tượng Date từ một chuỗi thời gian sang dạng GMT
toLocaleString	Chuyển một đối tượng Date từ một chuỗi sang dạng thời gian địa phương

Nhóm phương thức Parse & UTC:

Phương thức	Mô tả
Date.parse(date string)	Số mili giây trong một date string (chuỗi thời gian) tính từ 1/1/1970
Date.UTC(year, month, day, hours, min., secs.)	Số mili giây của một đối tượng thời gian tính từ 1/1/1970

Ví dụ 10:

<HTML>

<HEAD>

<SCRIPT language="JavaScript">

<!--

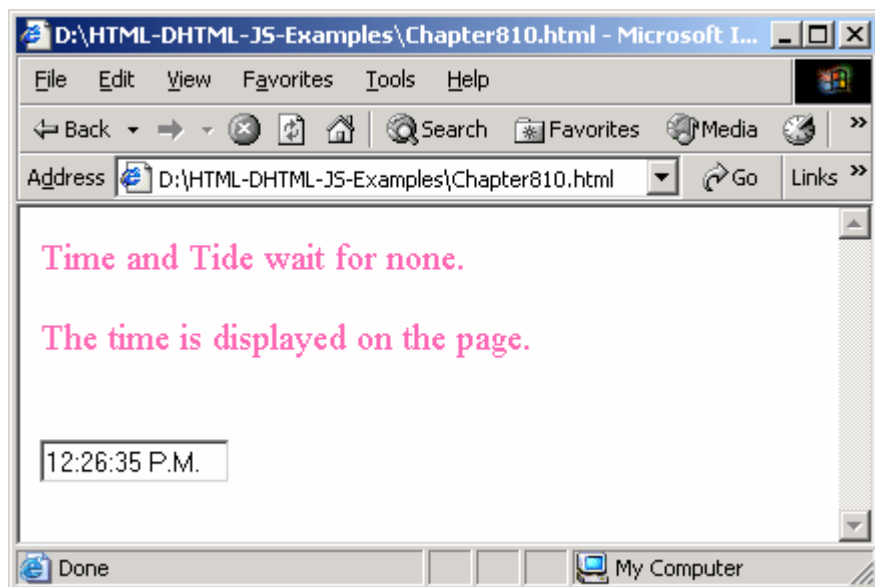
```
function disptime() {
    var time = new Date()
    var hour = time.getHours()
    var minute = time.getMinutes()
    var second = time.getSeconds()
    var temp = "" + ((hour > 12) ? hour - 12 : hour)
    temp += ((minute < 10) ? ":0" : ":") + minute
    temp += ((second < 10) ? ":0" : ":") + second
    temp += (hour >= 12) ? " P.M." : " A.M."
    document.MyPage.digits.value = temp
    id = setTimeout("disptime()",1000)
}
```

//-->

</SCRIPT>

```
</HEAD>
<BODY onLoad="disptime()">
    <P><font color = hotpink size = 4>Time and Tide wait
    for none.

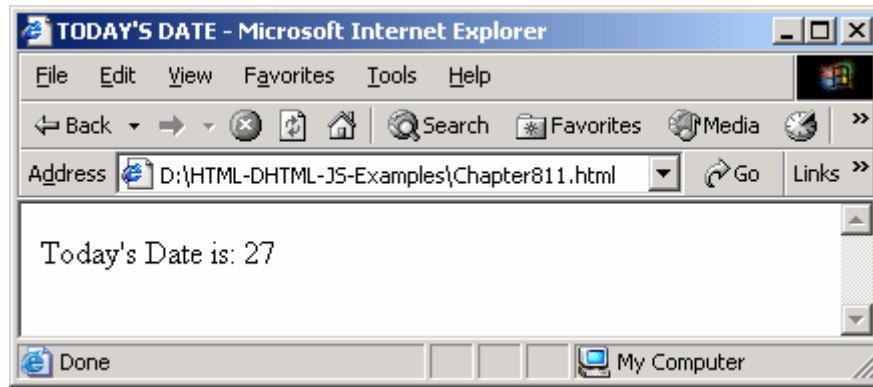
    <BR><BR>
    <P> The time is displayed on the page.
    <BR><BR><BR>
    <FORM NAME="MyPage">
        <INPUT TYPE="text" NAME="digits" SIZE=12 VALUE="">
    </FORM>
</BODY>
</HTML>
```



Hình 7.11: Kết quả của ví dụ 10

Ví dụ 11:

```
<HTML>
<HEAD>
    <TITLE> TODAY'S DATE </TITLE>
</HEAD>
<BODY>
    <script language="JavaScript">
        mydate=new Date()
        document.write("Today's Date is: "+ mydate.getDate());
    </script>
</BODY>
</HTML>
```



Hình 7.12: Kết quả của ví dụ 11

7.3 ĐỐI TƯỢNG EVENT – KHÁI NIỆM

Các chương trình JavaScript thường là hướng sự kiện. Các sự kiện là các hành động xảy ra trên trang web. Một sự kiện có thể do người dùng tạo ra – click chuột vào một button - hoặc do hệ thống tạo ra – thay đổi kích thước của trang.

Hầu hết các trình duyệt đều hỗ trợ một đối tượng Event. Mỗi sự kiện có một đối tượng Event tương ứng. Đối tượng Event cung cấp thông tin về sự kiện - loại sự kiện và vị trí của con trỏ tại thời điểm xảy ra sự kiện. Khi một sự kiện được phát sinh, nó được gửi đi như một đối số đến trình xử lý sự kiện. Dĩ nhiên, phải có một trình xử lý sự kiện trong trường hợp này.

Chẳng hạn, khi người dùng nhấp chuột, sự kiện **onmousedown** được phát sinh. Đối tượng Event chứa những thông tin sau:

- ✓ Loại sự kiện - Trong trường hợp này là nhấp chuột
- ✓ Vị trí x và y của con trỏ khi nhấp chuột
- ✓ Nút chuột nào được nhấn
- ✓ Các phím bổ trợ (Control, Alt, Meta, hoặc Shift) được nhấn vào thời điểm xảy ra sự kiện.

Đối tượng Event không thể được sử dụng trực tiếp với đối tượng window. Nó được sử dụng như một phần của trình xử lý sự kiện.

Một sự kiện bắt đầu bằng hành động hoặc điều kiện khởi tạo sự kiện và kết thúc bằng việc đáp lại của trình xử lý sự kiện. Vòng đời của một sự kiện thông thường gồm những bước sau:

1. Hành động người dùng hoặc điều kiện tương ứng với sự kiện xảy ra.
2. Đối tượng Event được cập nhật tức thì nhằm phản ánh trạng thái của sự kiện.
3. Sự kiện được kích hoạt.
4. Trình xử lý sự kiện tương ứng được gọi.
5. Trình xử lý sự kiện thực hiện hành động của nó và trả về điều khiển cho chương trình.

7.4 CÁC SỰ KIỆN JAVASCRIPT

Tập hợp các sự kiện tương ứng với các phần tử khác nhau trên trang là một phần của mô hình đối tượng tài liệu (Document Object Model), chứ không phải của JavaScript. Nghĩa là, các sự kiện được một phần tử nào đó hỗ trợ có thể khác nhau trên các trình duyệt.

Sau đây là một số sự kiện thường được hầu hết các đối tượng hỗ trợ:

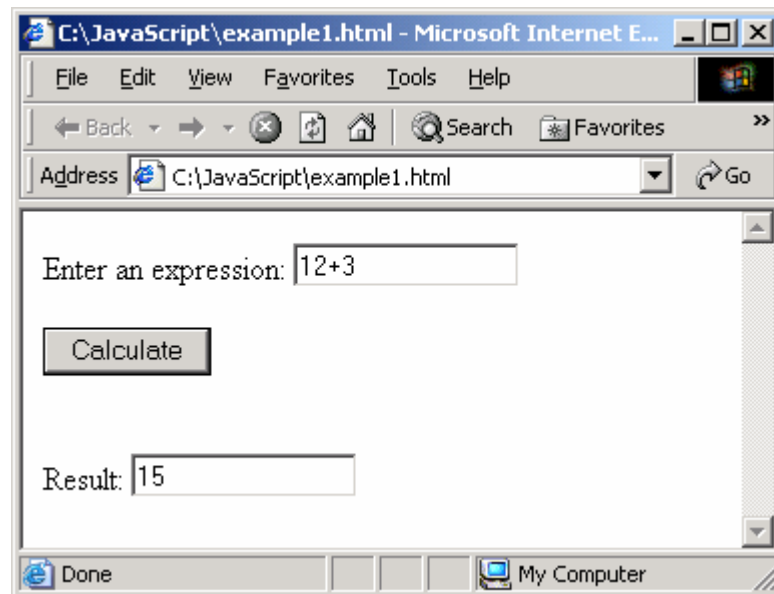
➤ **onClick**

Sự kiện `onClick` được tạo ra bất cứ khi nào người dùng nhấp chuột lên các phần tử form nào đó (`button`, `checkbox`, `radio button`, và phần tử **select**), hoặc lên các hyperlink.

Ví dụ 1:

```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE="JavaScript">
    function compute(form)
    {
      if (confirm("Are you sure?"))
        form.kết quả.value = eval(form.expr.value)
      else
        alert("Please come back again.")
    }
  </SCRIPT>
</HEAD>
<BODY>
  <FORM>
    Enter an expression:
    <INPUT TYPE="text" NAME="expr" SIZE=15 ><BR><BR>
    <INPUT TYPE="button" VALUE="Calculate" ONCLICK="compute(this.form)">
    <BR><BR><BR>
    Kết quả:
    <INPUT TYPE="text" NAME="kết quả" SIZE=15 >
    <BR>
  </FORM>
</BODY>
</HTML>
```

Kết quả đoạn mã trong ví dụ 1 được minh hoạ ở Hình 7.1.



Hình 7.1: Kết quả của ví dụ 1

➤ onChange

Sự kiện onChange xảy ra bất cứ khi nào một phần tử form thay đổi. Điều này có thể xảy ra khi nội dung của một trường văn bản thay đổi, hoặc khi một chọn lựa trong danh sách chọn lựa thay đổi. Tuy nhiên, sự kiện onChange không được tạo ra khi một radio button hoặc một checkbox được nhấp. Thay vào đó, sự kiện onClick sẽ được tạo ra.

Sự kiện onChange được gửi đi khi một phần tử hoàn tất việc thay đổi. Vì vậy, khi một textbox đang được hiệu chỉnh, sự kiện onChange chỉ được phát sinh sau khi việc hiệu chỉnh đã hoàn tất, và khi người dùng thoát khỏi textbox đó.

Ví dụ 2:

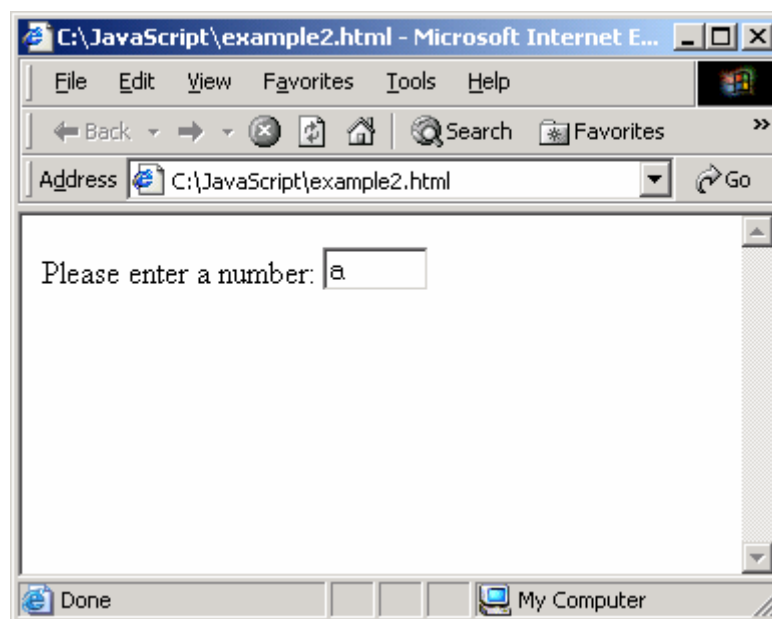
```
<HTML>
<HEAD>
  <SCRIPT LANGUAGE="JavaScript">
    <!-- hide script from old browsers
    function checkNum(num)
    {
      if (num == "")
      {
        alert("Please enter a number");
        return false;
      }
      if (isNaN (num))
      {
        alert("Please enter a numeric value");
```

```
        return false;
    }

    else alert ("Thank you");
}

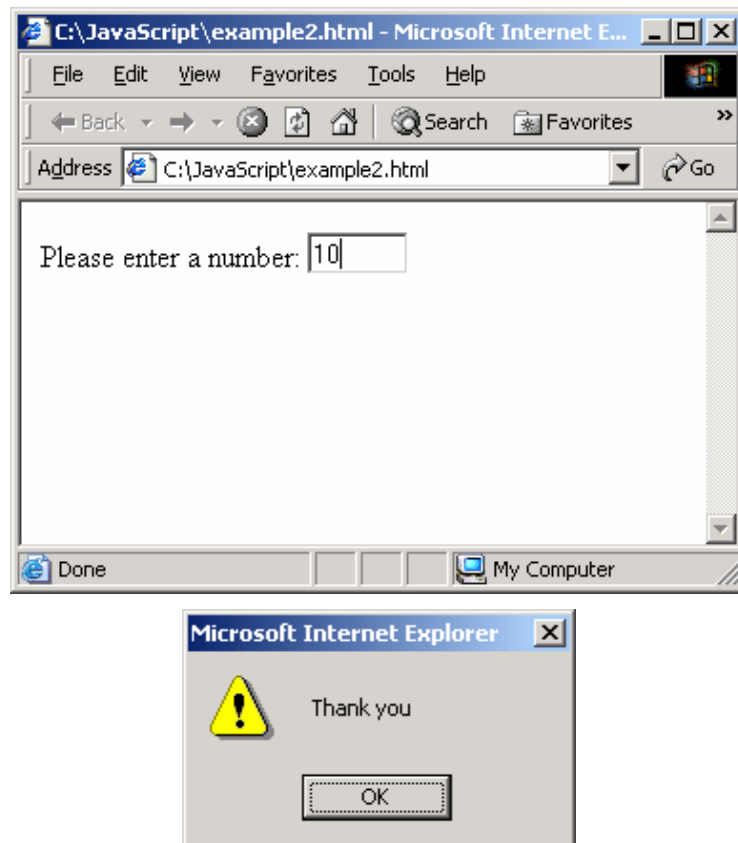
// end hiding from old browsers -->
</SCRIPT>
</HEAD>
<BODY bgColor = white>
    <FORM>
        Please enter a number:
        <INPUT type = text size = 5 onChange="checkNum(this.value)">
    </FORM>
</BODY>
</HTML>
```

Hình 7.2(1) và 17.2(2) minh họa kết quả của đoạn mã trong ví dụ 2.



Hình 7.2: Kết quả của ví dụ 2(1)

Nếu chúng ta nhập vào một giá trị số:



Hình 7.2: Kết quả của ví dụ 2(2)

➤ **onFocus**

Sự kiện onFocus được gửi đi bất cứ khi nào một phần tử form trở thành phần tử hiện thời. Chỉ khi một phần tử nhận được focus nó mới nhận được input từ người dùng. Điều này có thể xảy ra khi người dùng nhấp chuột lên phần tử, hoặc sử dụng phím Tab hoặc Shift+Tab (di chuyển tới các phần tử trên form).

➤ **onBlur**

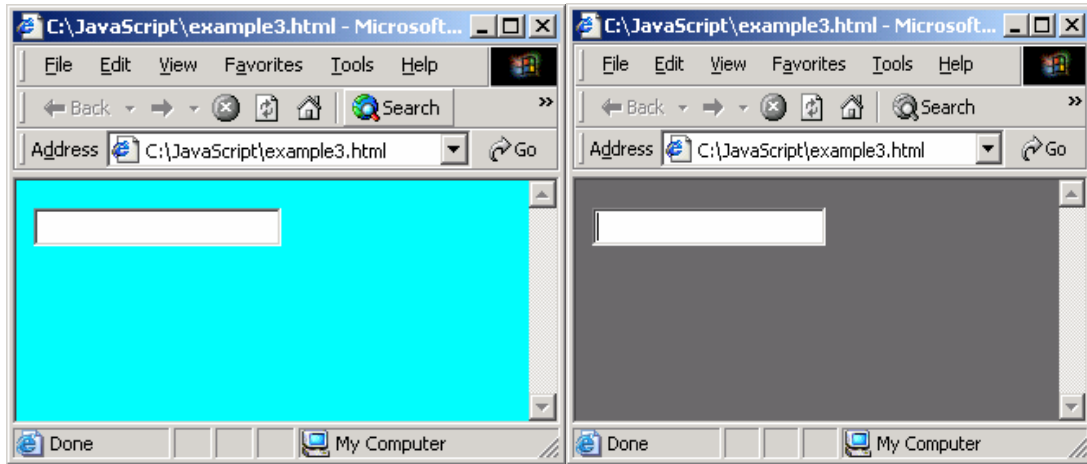
Blur ngược với focus. Khi người dùng rời khỏi một phần tử trên form, sự kiện onBlur được kích hoạt. Đối với một số phần tử, nếu nội dung của nó cũng bị thay đổi, thì sự kiện onChange cũng được kích hoạt.

Ví dụ 3:

```
<HTML>
  <BODY BGCOLOR="lavender">
    <FORM>
      <INPUT type = text name = text1
        onBlur="(document.bgColor='aqua')"
        onFocus="(document.bgColor='dimgray') ">
    </FORM>
  </BODY>
```

</HTML>

Khi textbox nhận được focus, màu nền sẽ chuyển sang DIMGRAY, khi mất focus (blur), màu nền sẽ chuyển sang AQUA.



Hình 7.3: Kết quả của ví dụ 3 – Blur (hình trái) and focus (hình phải)

➤ onmouseover

Sự kiện onmouseover được tạo ra bất cứ khi nào con trỏ chuột di chuyển lên trên một phần tử.

➤ onmouseout

Sự kiện onmouseout được tạo ra bất cứ khi nào con trỏ chuột di chuyển ra khỏi phần tử đó.

Ví dụ 4:

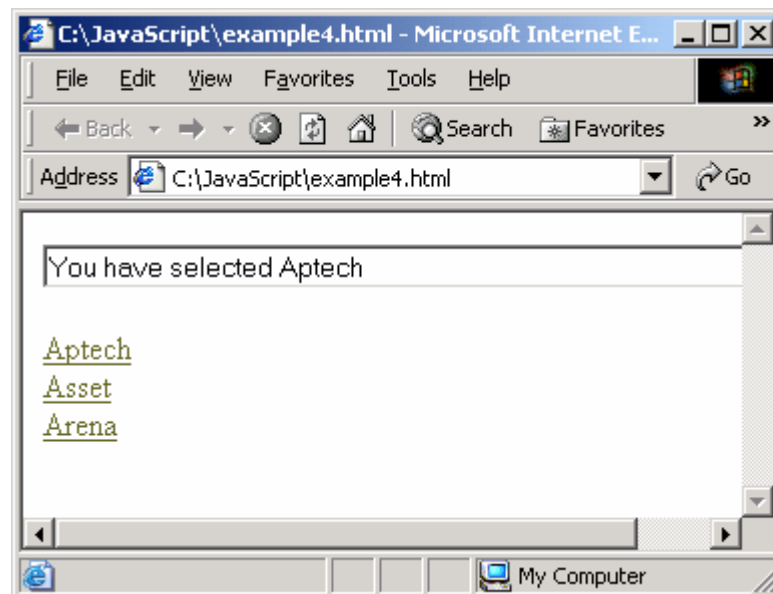
```
<html>
<head>
  <script language = "javascript">
    var num =0
    function showLink(num)
    {
      if (num==1)
      {
        document.forms[0].elements[0].value= "You have selected
        Aptech";
      }
      if (num==2)
      {
        document.forms[0].elements[0].value = "You have selected
        Asset";
      }
    }
  </script>
</head>
<body>
  <input type="text" value="Select a link" />
</body>
</html>
```

```
        if (num==3)
        {
            document.forms[0].elements[0].value = "You have selected
            Arena";
        }
    }
</script>
</head>
<body>
    <form>
        <input type=text size=60 >

    </form>

    <a href="#" onmouseover="showLink(1)" document.bgcolor=
    green">Aptech</a><br>
    <a href="#" onmouseover="showLink(2)">Asset</a><br>
    <a href="#" onmouseover="showLink(3)">Arena</a><br>
</body>
</html>
```

Khi di chuyển chuột qua **Aptech**, kết quả được hiển thị như sau.



Hình 7.4: Kết quả của ví dụ 4

➤ OnLoad

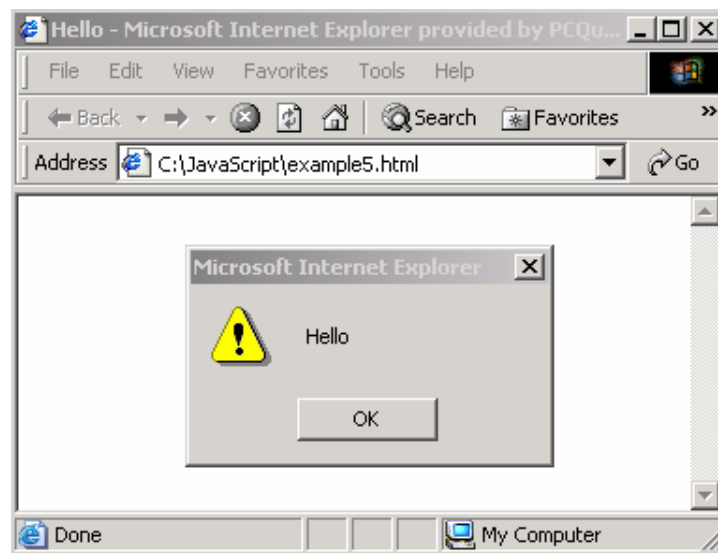
Sự kiện onLoad (áp dụng cho đối tượng body) được phát sinh khi đã tải xong tài liệu. Nó cũng được phát sinh khi một ảnh đã tải xong.

Ví dụ 5:

<HTML>

```
<HEAD>
  <TITLE>Hello </TITLE>
</HEAD>
<BODY onLoad="alert('Hello')">
</BODY>
</HTML>
```

Kết quả:



Hình 7.5: Kết quả của ví dụ 5

➤ **onSubmit**

Sự kiện onSubmit được tạo ra bất cứ khi nào người dùng truyền dữ liệu từ form đi (thường sử dụng nút Submit). Sự kiện xảy ra trước khi form thật sự được gửi đi. Thật ra, trình xử lý sự kiện tương ứng với sự kiện này có thể ngăn chặn form không được gửi đi bằng cách trả về giá trị false. Cách này dùng để kiểm tra sự hợp lệ của dữ liệu nhập vào.

➤ **onMouseDown**

Sự kiện này được kích hoạt khi hành động nhấp chuột xảy ra. Nghĩa là, khi người dùng nhấp chuột. Đây là trình xử lý sự kiện cho các đối tượng button, document, và link.

➤ **onMouseUp**

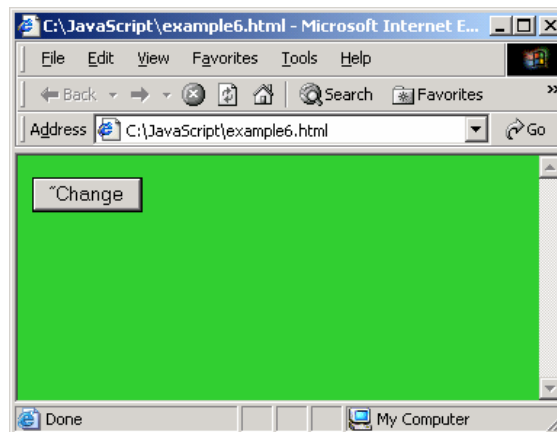
Sự kiện này được kích hoạt khi hành động thả chuột xảy ra. Nghĩa là, khi người dùng thả chuột. Đây là trình xử lý sự kiện cho đối tượng button, document, và link.

Ví dụ 6:

```
<HTML>
<BODY BGCOLOR="lavender">
  <FORM>
    <INPUT type = button name = text1 value="Change Color"
```

```
onmousedown=" (document.bgColor='aqua') "  
onmouseup=" (document.bgColor='limegreen') "  
</FORM>  
</BODY>  
</HTML>
```

Hình dưới đây hiển thị kết quả của ví dụ 6



Hình 7.6: Kết quả của ví dụ 6

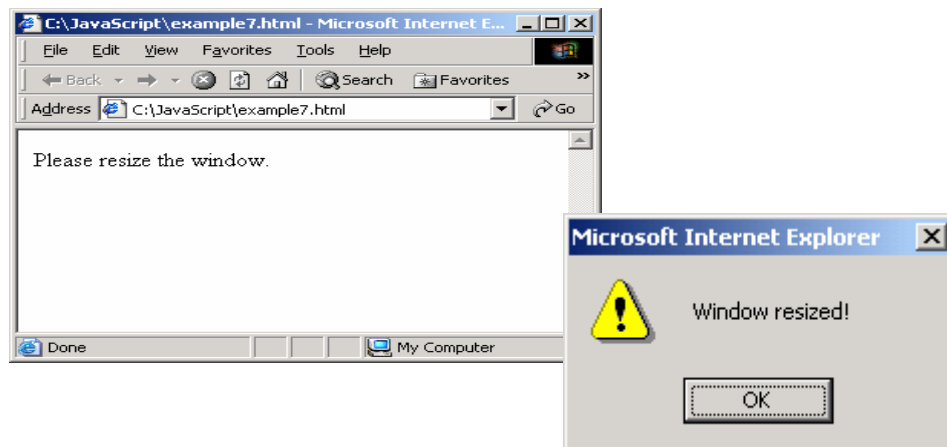
➤ onResize

Sự kiện này được kích hoạt khi hành động thay đổi kích thước cửa sổ xảy ra. Nghĩa là, khi người dùng hoặc một script làm thay đổi kích thước một cửa sổ hay frame. Đây là trình xử lý sự kiện cho các đối tượng Window.

Ví dụ 7

```
<html>  
<head>  
  <script language="JavaScript">  
    window.onresize= notify;  
    function notify()  
    {  
      alert("Window resized!");  
    }  
  </script>  
</head>  
<body>  
  Please resize the window.  
</body>  
</html>
```

Khi thay đổi kích thước cửa sổ, kết quả xuất hiện như sau:



Hình 7.7: Kết quả của ví dụ 7

7.5 TRÌNH XỬ LÝ SỰ KIỆN

Khi một sự kiện được khởi tạo, chúng ta có thể tạo một đoạn mã JavaScript để đáp ứng lại sự kiện. Đoạn mã này được gọi là trình xử lý sự kiện. Trình xử lý sự kiện có thể là một câu lệnh đơn, một tập hợp các câu lệnh hoặc một hàm

```
<INPUT TYPE="button"
      NAME="docode"
      onClick="DoOnClick();">
```

Khi nhấp chuột vào một button, sự kiện onClick được khởi tạo. Sự kiện onClick gọi hàm DoOnClick và thực thi những câu lệnh bên trong hàm.

Trình xử lý sự kiện cho các thẻ HTML

Để khởi tạo trình xử lý sự kiện cho thẻ HTML, chúng ta phải chỉ định thẻ và thuộc tính trình xử lý sự kiện. Sau đó chúng ta gán mã JavaScript. Đoạn mã phải được đặt trong cặp dấu nháy kép.

```
<TAG eventHandler="JavaScript Code">
```

Các đối số chuỗi phải được đặt trong cặp dấu nháy đơn.

```
<INPUT TYPE="button" NAME="Button1" VALUE="Open Sesame!"
      onClick="window.open('mydoc.html', 'newWin')">
```

Thay vì sử dụng nhiều câu lệnh JavaScript, hàm sẽ giúp cho việc thiết kế chương trình tốt hơn. Chúng ta sẽ gọi hàm khi cần thiết. Hơn nữa các hàm đó có thể được dùng bởi các phần tử khác.

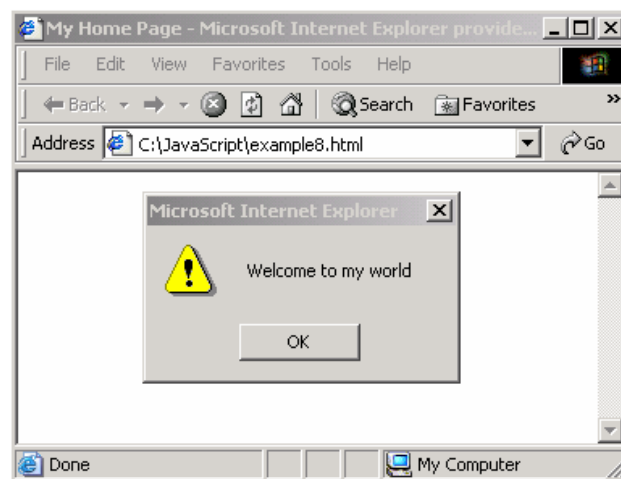
Câu lệnh này gán hàm greeting() cho trình xử lý sự kiện onLoad của window. Thuộc tính trình xử lý sự kiện được tham chiếu đến hàm greeting chứ không phải lời gọi đến hàm greeting()

Ví dụ 8

```
<HTML>
<HEAD>
  <TITLE>My Home Page</TITLE>
  <SCRIPT LANGUAGE="JavaScript">
```

```
function greeting() {  
    alert("Welcome to my world");  
}  
</SCRIPT>  
</HEAD>  
<BODY onLoad="greeting()">  
</BODY>  
</HTML>
```

Kết quả:



Hình 7.8: Kết quả ví dụ 8

Trình xử lý sự kiện như là những thuộc tính

Chúng ta cũng có thể gán một hàm cho một trình xử lý sự kiện của một đối tượng. Cú pháp như sau:

```
object.eventhandler = function;
```

Ví dụ,

```
window.onload = greeting;
```

Chúng ta xem lại ví dụ trên và sử dụng trình xử lý sự kiện như những thuộc tính:

Ví dụ 9

```
<HTML>  
<HEAD>  
  <TITLE>My Home Page</TITLE>  
  <SCRIPT LANGUAGE="JavaScript">  
    <!--  
      function greeting() {  
        alert("Welcome to my world");  
      }  
      window.onload = greeting;
```

```
// -->  
</SCRIPT>  
</HEAD>  
</HTML>
```

Kết quả sẽ tương tự như Hình 7.8. Điểm mạnh của kỹ thuật này là tính linh hoạt. Chúng ta có thể thay đổi nhanh chóng các trình xử lý sự kiện khi được yêu cầu.

TÀI LIỆU THAM KHẢO

1. Giáo trình Lập trình Web, Trung tâm Aptech
2. Tự học HTML, Trung tâm Điện toán Trường Đại học Bách Khoa Tp HCM.