

## MỤC LỤC

CHƯƠNG 1.....	1
GIỚI THIỆU CHUNG.....	1
Các ngôn ngữ đánh dấu dùng cho thiết bị điện tử.....	1
Ngôn ngữ đánh dấu thiết bị cầm tay.....	1
Ngôn ngữ đánh dấu vô tuyến.....	1
Ngôn ngữ đánh dấu siêu văn bản thu nhỏ.....	1
Các ngôn ngữ đánh dấu văn bản ứng dụng Web.....	2
Ngôn ngữ đánh dấu siêu văn bản.....	2
Ngôn ngữ đánh dấu mở rộng.....	2
Ngôn ngữ XHTML.....	2
Ngôn ngữ DHTML.....	3
Ngôn ngữ đánh dấu siêu văn bản (HTML).....	3
Giới thiệu HTML.....	3
Đặc điểm của HTML.....	4
Cấu trúc của một tài liệu HTML.....	5
Qui trình tạo một tài liệu HTML.....	6
CHƯƠNG 2.....	8
CÁC THẺ CƠ BẢN TRONG HTML.....	8
Thẻ và các thuộc tính của thẻ.....	8
Các thẻ cơ bản trong HTML.....	9
Thẻ xác định văn bản HTML.....	9
Thẻ xác định phần đầu cho trang web.....	9
Thẻ xác định tiêu đề cho văn bản HTML.....	9
Thẻ xác định các danh mục trang web.....	10
Thẻ tạo danh sách.....	11
Thẻ tạo danh sách không thứ tự.....	12
Thẻ tạo danh sách có thứ tự.....	14
Thẻ xác định văn bản trang web.....	19
Thẻ tạo đường thẳng.....	19
Thẻ xác định dòng chú thích.....	21
Các thẻ vận dụng với văn bản.....	21
Thẻ vận dụng cho kiểu chữ.....	21
2.3.1.1 Làm chữ đậm.....	21
2.3.1.2 Làm chữ in nghiêng.....	22
2.3.1.3 Thay đổi kích thước chữ.....	23
2.3.1.4 Tạo dòng chữ thấp.....	23
2.3.1.5 Tạo dòng chữ cao.....	24
2.3.1.6 Gạch ngang và gạch dưới chữ.....	25
2.3.1.7 Tạo chữ dạng riêng.....	25
2.3.1.8 Tạo dạng chữ bị xoá.....	25
2.3.1.9 Tạo dạng chữ chèn vào.....	25
Thẻ vận dụng cho hiệu ứng font chữ.....	25
2.3.2.1 Chọn font chữ cho văn bản.....	25
2.3.2.2 Đổi cỡ chữ văn bản.....	26
2.3.2.3 Chọn cỡ chữ mặc định.....	27
2.3.2.4 Đổi màu chữ.....	27
2.3.2.5 Làm chữ nhấp nháy.....	27
Thẻ vận dụng trình bày trang Web.....	27
Lựa chọn màu nền.....	27

Lựa chọn hình ảnh làm nền.....	28
Chỉnh lề cho trang Web.....	29
Tạo đoạn văn bản.....	29
Ngắt đoạn.....	30
Một số thẻ đặc biệt khác.....	31
Thẻ làm việc với siêu liên kết.....	31
Giới thiệu siêu liên kết và URL.....	31
Sử dụng siêu liên kết.....	32
Thẻ Meta.....	38
Các thẻ DIV và SPAN.....	39
Các thẻ múa đoạn.....	40
2.5.3.1 Thẻ <ADDRESS>.....	40
2.5.3.2 Thẻ <BLOCKQUOTE>.....	41
2.5.3.3 Thẻ <PRE>.....	42
Sử dụng ký tự đặc biệt trong tài liệu HTML.....	43
CHƯƠNG.....	48
LÀM VIỆC VỚI BẢNG - BIỂU MẪU – KHUNG VÀ ĐA PHƯƠNG TIỆN.....	48
Làm việc với bảng.....	48
Cách tạo bảng.....	48
Các thuộc tính của bảng.....	50
Thuộc tính của thẻ <TABLE>.....	50
Thuộc tính của thẻ <TR>.....	50
Thuộc tính của thẻ <TD>.....	51
Hiệu chỉnh bảng.....	52
Tạo khung viền cho bảng.....	52
Thay đổi kích thước bảng.....	53
Bổ sung cạnh và đường kẻ lưới.....	53
Trang trí văn bản chung quanh bảng.....	54
Kết hợp các cột và các dòng.....	55
Canh lề nội dung trong ô.....	57
Sử dụng hình ảnh làm nền cho bảng.....	57
Làm việc với biểu mẫu.....	59
Sử dụng biểu mẫu.....	59
Phần tử FORM.....	59
Các phần tử nhập của HTML.....	60
Phần tử INPUT.....	60
Button.....	61
Textbox.....	61
Checkbox.....	61
Radio.....	62
Submit.....	62
Ảnh.....	63
Reset.....	63
Phần tử TextArea.....	64
Phần tử BUTTON.....	65
Phần tử Select.....	67
Phần tử LABEL.....	70
Tạo biểu mẫu.....	71
Thiết lập tiêu điểm (Focus).....	73
Thứ tự tab.....	74
Phím truy cập (Access Keys).....	74

Phân tử vô hiệu hoá.....	74
Làm việc với khung.....	74
dụng.....	75
Tại sao sử dụng khung?.....	75
Làm việc với khung.....	75
3.3.2.1 Sử dụng khung.....	75
3.3.2.2 Liên kết các khung.....	80
3.3.2.3 Phân tử NOFRAMES.....	81
3.3.2.4 Phân tử IFRAMES (inline frame).....	82
Làm việc với đa phương tiện.....	83
Sử dụng hình ảnh trong tài liệu HTML.....	83
Chèn ảnh tĩnh.....	84
Chèn ảnh động (.GIF) vào tài liệu HTML.....	86
Chèn âm thanh vào tài liệu HTML.....	87
Chèn video vào tài liệu HTML.....	88
CHƯƠNG 4.....	89
STYLE SHEET.....	89
DHTML.....	89
Giới thiệu DHTML.....	89
Các đặc điểm của DHTML.....	90
Style sheet.....	91
4.2.1 Khái niệm, chức năng và các lợi ích của style sheet.....	91
4.2.2 Quy tắc stylesheet.....	94
4.2.3 Các Selector trong style sheet.....	96
4.2.4 Kết hợp và chèn một style sheet vào tài liệu HTML.....	103
4.2.5 Thiết lập thuộc tính trong style sheet.....	105
CHƯƠNG 5.....	106
TỔNG QUAN VỀ JAVASCRIPT.....	106
5.1 Giới thiệu về Javascript.....	106
Javascript.....	106
Tim hiểu lịch sử của JavaScript.....	106
Nguồn gốc của JavaScript.....	107
JavaScript đến với Internet Explorer.....	107
JavaScript trở thành chuẩn chính thức.....	107
JavaScript hiện nay đã phát triển đến đâu?.....	108
Nhúng Javascript vào file HTML.....	108
Dùng thẻ <SCRIPT>.....	109
Dùng file bên ngoài.....	110
Dùng JavaScript trong trình xử lý sự kiện.....	111
Thẻ <NOSCRIPT> và </NOSCRIPT>.....	112
Biến, hằng và các kiểu dữ liệu trong JavaScript.....	113
5.2.1 Biến và phân loại biến.....	113
5.2.2 Hằng.....	114
5.2.3 Các kiểu dữ liệu trong JavaScript.....	115
5.2.3.1 Kiểu số nguyên.....	116
5.2.3.2 Kiểu số thực (kiểu số dấu chấm động).....	116
5.2.3.3 Kiểu Logical (hay Boolean).....	116
5.2.3.5 Kiểu null.....	117
Câu hỏi và bài tập.....	118
CHƯƠNG 6.....	119
TOÁN TỬ VÀ BIỂU THỨC TRONG JAVASCRIPT.....	119

6.1 Các toán tử trong JavaScript.....	119
6.1.1 Các toán tử thông dụng.....	119
6.1.1.1 Toán tử gán.....	119
6.1.1.2 Toán tử số học.....	120
6.1.1.3 Toán tử so sánh.....	121
6.1.1.4 Toán tử logic.....	122
6.1.1.5 Toán tử thao tác trên bit.....	123
6.1.1.6 Toán tử chuỗi.....	125
6.1.2 Một số toán tử khác.....	126
6.1.2.1 Toán tử điều kiện.....	126
6.1.2.2 Toán tử dấu phẩy.....	126
6.1.2.3 Toán tử new.....	126
6.1.2.4 Toán tử typeof.....	126
6.1.2.5 Toán tử this.....	128
6.1.3 Thứ tự ưu tiên của các toán tử.....	129
6.2 Các biểu thức trong JavaScript.....	129
6.2.1 Biểu thức regular.....	130
6.2.2 Tạo ra một biểu thức regular.....	132
6.2.2.1 Khởi tạo đối tượng (Object initializer).....	133
6.2.2.2 Gọi hàm khởi tạo của đối tượng RegExp.....	133
6.2.3 Sử dụng biểu thức regular.....	134
6.3 Câu hỏi và bài tập.....	136
CHƯƠNG 7.....	139
CÂU LỆNH ĐIỀU KIỆN.....	139
7.1 Lệnh và khối lệnh.....	139
7.1.1 Lệnh và quy ước lệnh trong JavaScript.....	139
7.1.2 Khối lệnh.....	139
7.2 Các câu lệnh điều kiện.....	139
7.2.1 Câu lệnh if...else.....	139
7.2.2 Câu lệnh switch.....	143
7.3 Câu hỏi và bài tập.....	146
CHƯƠNG 8.....	149
CÂU LỆNH VÒNG LẶP.....	149
8.1 Các lệnh vòng lặp trong JavaScript.....	149
8.1.1 Câu lệnh for.....	149
8.1.2 Câu lệnh do..while.....	151
8.1.3 Câu lệnh while.....	152
8.2 Các lệnh chuyển điều khiển trong vòng lặp.....	153
8.2.1 Câu lệnh label.....	153
8.2.2 Câu lệnh break.....	153
8.2.3 Câu lệnh continue.....	154
8.3 Các lệnh thao tác trên đối tượng.....	156
8.3.1 Câu lệnh for...in.....	156
8.3.2 Câu lệnh with.....	157
8.4 Câu hỏi và bài tập.....	158
CHƯƠNG 9.....	160
HÀM.....	160
9.1 Khái niệm và các thao tác trên hàm.....	160
9.1.1 Khái niệm về hàm.....	160
9.1.2 Tạo hàm.....	160
9.1.3 Gọi hàm.....	162

9.1.4 Câu lệnh return.....	162
9.2 Một số hàm thông dụng được hỗ trợ bởi JavaScript.....	164
9.2.1 Hàm eval.....	164
9.2.2 Hàm isFinite.....	164
9.2.3 Hàm isNaN.....	165
9.2.4 Các hàm parseInt và parseFloat.....	165
9.2.5 Các hàm Number và String.....	166
9.3 Câu hỏi và bài tập.....	166
CHƯƠNG 10.....	169
MẢNG.....	169
10.1 Khái niệm về mảng và các thao tác trên mảng trong JavaScript.....	169
10.1.1 Khái niệm về mảng.....	169
10.1.2 Tạo mảng.....	169
10.1.3 Gán giá trị cho các phần tử mảng.....	169
10.1.4 Truy cập đến các phần tử mảng.....	171
10.2 Các phương thức của mảng.....	171
10.2.1 Phương thức concat.....	172
10.2.2 Phương thức join.....	172
10.2.3 Phương thức pop.....	174
10.2.4 Phương thức push.....	174
10.2.5 Phương thức reverse.....	174
10.2.6 Phương thức sort.....	175
10.3 Mảng hai chiều.....	176
10.4 Câu hỏi và bài tập.....	178
CHƯƠNG 11.....	180
CÁC ĐỐI TƯỢNG CƠ BẢN CỦA JAVASCRIPT.....	180
11.1 Đối tượng Math.....	181
11.1.1 Mô tả.....	181
11.1.2 Các thuộc tính của đối tượng Math.....	182
11.1.3 Các phương thức của đối tượng Math.....	183
11.2 Đối tượng String.....	185
11.2.1 Mô tả.....	185
11.2.2 Các thuộc tính của đối tượng String.....	186
11.2.3 Các phương thức của đối tượng String.....	186
11.2.4 Tìm kiếm trong một chuỗi.....	188
11.2.5 Định vị các ký tự trong một chuỗi.....	190
11.3 Đối tượng Date.....	192
11.3.1 Mô tả.....	192
11.3.2 Các nhóm phương thức của đối tượng Date.....	193
11.3.3 Các phương thức của đối tượng Date.....	193
11.3.3.1 Nhóm phương thức get.....	193
11.3.3.2 Nhóm phương thức set.....	194
11.3.3.3 Nhóm phương thức to.....	194
11.3.3.4 Nhóm phương thức parse và UTC.....	194
11.4 Câu hỏi và bài tập.....	195
CHƯƠNG 12.....	199
XỬ LÝ FORM VÀ CÁC SỰ KIỆN CHO CÁC PHẦN TỬ TRÊN FORM.....	199
12.1 Giới thiệu về đối tượng form.....	199
12.1.1 Mô tả đối tượng.....	199
12.1.2 Các thuộc tính và phương thức của đối tượng form.....	199
12.2 Xử lý sự kiện trong JavaScript.....	201

12.2.1 Khái niệm về sự kiện và trình xử lý sự kiện.....	201
12.2.2 Các sự kiện JavaScript phổ biến.....	202
12.2.3 Làm việc với trình xử lý sự kiện.....	212
12.2.3.1 Trình xử lý sự kiện cho các thẻ HTML.....	212
12.2.3.2 Trình xử lý sự kiện như là những thuộc tính.....	213
12.3 Sử dụng sự kiện cho các thành phần trên form.....	214
12.3.1 Đối tượng Textfield (Trường văn bản).....	214
12.3.2 Đối tượng Command Button.....	216
12.3.3 Đối tượng Checkbox.....	217
12.3.4 Đối tượng radio.....	219
12.3.5 Đối tượng ComboBox (lựa chọn).....	222
12.3.6 Kiểm tra tính hợp lệ của nội dung các trường trên form.....	222
12.4 Câu hỏi và bài tập.....	227

# CHƯƠNG 1

## GIỚI THIỆU CHUNG

Trong ngành in ấn trước đây, để chỉ thị cho thợ in sắp chữ trong văn bản, tác giả hay chủ bút thường vẽ các vòng tròn trong bản thảo và chú thích bằng một ngôn ngữ, tương tự tốc kí. Ngôn ngữ đó được gọi là ngôn ngữ đánh dấu (Markup Language).

Do những nhu cầu phát triển của khoa học công nghệ mà người ta đã xây dựng ra một ngôn ngữ có tên: Ngôn ngữ đánh dấu tổng quát (SGML - Standard Generalized Markup Language). SGML được phát triển bởi Ed Mosher, Ray Lorie và Charles F. Goldfarb của nhóm IBM research vào năm 1969. Ban đầu nó có tên là Generalized Markup Language (GML), và được thiết kế để diễn tả các ngôn ngữ khác bao gồm văn phạm, từ vựng của chúng. Năm 1986, SGML được tổ chức ISO (International Standard Organisation) thu nhận làm tiêu chuẩn để lưu trữ và trao đổi dữ liệu. Và sau này các ngôn ngữ đánh dấu thiết bị điện tử, thiết kế Web được phát triển dựa vào cơ sở của ngôn ngữ đánh dấu tổng quát SGML.

### 1.1 Các ngôn ngữ đánh dấu dùng cho thiết bị điện tử

Một số ngôn ngữ đánh dấu được sử dụng để cung cấp các dịch vụ và nội dung đến các thiết bị điện tử như máy nhắn tin, điện thoại di động, thiết bị vô tuyến. Các ngôn ngữ đó bao gồm: Ngôn ngữ đánh dấu thiết bị cầm tay, ngôn ngữ đánh dấu vô tuyến và ngôn ngữ đánh dấu siêu văn bản thu nhỏ.

#### 1.1.1 Ngôn ngữ đánh dấu thiết bị cầm tay

Ngôn ngữ đánh dấu thiết bị cầm tay (HDML - Handheld Device Markup Language) được thiết kế cho máy nhắn tin vô tuyến, điện thoại, tết bào điện tử và các thiết bị cầm tay để lấy các thông tin từ các trang Web. HDML là một tập hợp con của WAP, được Openwave Systems xây dựng trước khi chuẩn WAP ra đời. Công ty AT&T Wireless mở dịch vụ dựa trên HDML vào năm 1996.

HDML trước tiên được tạo ra để xây dựng nội dung dựa trên Web cho máy điện thoại di động và các thiết bị cầm tay. Vào năm 1997, HDML 2.0 được tung ra cho phép người sử dụng nhận các thông số chứng khoán, các đầu đề tin tức, các cảnh báo thư điện tử trên máy điện thoại di động.

#### 1.1.2 Ngôn ngữ đánh dấu vô tuyến

Ngôn ngữ đánh dấu vô tuyến (WML - Wireless Markup Language) là một ngôn ngữ dựa trên thẻ được sử dụng trong giao thức ứng dụng vô tuyến. WML là một loại văn bản XML cho phép các công cụ XML và HTML sử dụng để phát triển các ứng dụng WML. WML được phát triển từ HDML của Openwave nhưng nó không phải là một siêu tập hợp của HDML, các đặc trưng HDML không xuất hiện trong WML.

Tiêu chuẩn WML chính thống được phát triển và được diễn đàn WAP duy trì. WML có bốn lĩnh vực chức năng quan trọng: mẫu và hiển thị ký tự; tổ chức, định vị thẻ và tập; kết nối và định vị liên thẻ; thẻ hiện thông số chuỗi và quản lý trình trạng thái.

#### 1.1.3 Ngôn ngữ đánh dấu siêu văn bản thu nhỏ

Ngôn ngữ đánh dấu siêu văn bản thu nhỏ (cHTML - Compact HTML) là một tập hợp con của HTML cho điện thoại di động và PDA, được công ty NTT Docomo phát triển cho hệ thống vô tuyến i-Mode ở Nhật Bản. cHTML được thiết kế cho việc hiển thị màn hình và hỗ trợ một số chức năng của các thiết bị cầm tay.

Ví dụ, cHTML hỗ trợ các nút bấm di chuyển khi con chuột không được sử dụng.

## 1.2 Các ngôn ngữ đánh dấu văn bản ứng dụng Web

### 1.1 *Ngôn ngữ đánh dấu siêu văn bản*

Ngôn ngữ đánh dấu siêu văn bản (HTML - Hyper Text Markup Language) là một ngôn ngữ đánh dấu được thiết kế để tạo ra các trang Web, trong đó các thông tin được trình bày trên World Wide Web. HTML là một ứng dụng đơn giản của SGML, được sử dụng trong các tổ chức công nghệ truyền thông. HTML giờ đây đã trở thành một chuẩn Internet do tổ chức World Wide Web Consortium (W3C) duy trì. Phiên bản mới nhất của nó hiện là HTML 4.01. Tuy nhiên, HTML hiện không còn được phát triển tiếp, người ta đã thay thế nó bằng [XHTML](#).

HTML tồn tại như là các tập tin văn bản chứa trên các máy tính nối vào mạng [Internet](#). Các file này chứa [thẻ đánh dấu](#), là các chỉ thị cho chương trình về cách hiển thị, xử lý văn bản ở dạng thuận tự. Các file này thường được truyền đi trên mạng internet thông qua giao thức mạng [HTTP](#), sau đó thì phần HTML sẽ được hiển thị thông qua một [trình duyệt web](#), các trình duyệt đảm nhiệm công việc đọc văn bản của trang cho người sử dụng, [phần mềm đọc email](#), hay một thiết bị [không dây](#) như [điện thoại di động](#).

### 1.2 *Ngôn ngữ đánh dấu mở rộng*

Ngôn ngữ đánh dấu mở rộng (XML – eXtend Markup Language) khá giống với HTML, hai ngôn ngữ này có cùng luật cú pháp. Tuy nhiên, tính linh hoạt của XML cho phép bạn tạo và sử dụng tập thẻ và tập thuộc tính riêng để nhận biết các phần tử cấu trúc và nội dung tài liệu.

XML không chỉ là ngôn ngữ đánh dấu, nó còn có phương pháp định ra nội dung tài liệu, tương tự như HTML định hình thức tài liệu trên Web. Với HTML, người thiết kế đánh dấu văn bản, hình ảnh cùng các thành phần khác của trang Web bằng tập thẻ mà không liên quan tới ý nghĩa tài liệu, XML không chỉ chỉ định hình thức mà còn cả nội dung tài liệu.

XML được xem là công cụ mạnh hơn HTML do nó mang lại thông tin đầy đủ về dữ liệu. Một số tổ chức chuyên môn đã xây dựng ngôn ngữ XML riêng, bao gồm các thẻ nhận diện đặc tả công nghiệp. Ví dụ: Ngành công nghiệp hóa học đã phát triển Chemical Markup Language (CML).

XML giúp bạn tạo tài liệu độc lập với server. Tài liệu được nằm ngay trên máy khi tài liệu được tải về tiếp tục sử dụng không phụ thuộc vào Server. Mặt khác XML mang tính chặt chẽ theo tiêu chuẩn của ngôn ngữ đánh dấu văn bản.

### 1.3 *Ngôn ngữ XHTML*

XHTML là sự kết hợp giữa HTML 4.0 và XML 1.0 thành một định dạng riêng cho Web. XHTML cho phép HTML mở rộng bằng các thẻ sở hữu. XHTML được mã hóa chặt chẽ hơn HTML và phải tuân thủ nhiều quy tắc cấu trúc hơn.

XHTML 1.0 được thiết kế nhằm mục đích tạo thói quen tốt cho người xây dựng trang Web. Bởi vì có rất nhiều người trình bày trang Web theo cách thức của một trình duyệt thể hiện mà không quan tâm tới sử dụng các HTML chuẩn, điều này sẽ gây ra hai tác hại: Thứ nhất là kết quả hiển thị sẽ phụ thuộc vào trình duyệt của người sử dụng, thứ hai là tạo ra thói quen không tốt khi thiết kế, đó là chỉ quan tâm tới trình duyệt thể hiện mục đích của mình mà không quan tâm chuẩn của nó.

Sử dụng XHTML chuẩn là những bước đầu tiên để sẵn sàng xây dựng và triển khai XML vì việc xây dựng XML đòi hỏi phải chặt chẽ hơn HTML và XML không chấp nhận một lỗi cú pháp trong tài liệu.

Có hai lí do để sử dụng XHTML cho Website:

- Xây dựng các trang web động một cách tin cậy, dựa vào cú pháp chặt chẽ. Dữ liệu cho các trang Web động thường được khai thác từ cơ sở dữ liệu, các file hoặc các nguồn khác và được hiển thị theo những template phụ thuộc vào yêu cầu của người sử dụng. Việc xây dựng một cách cầu thủ sẽ không chỉ gây ra những lỗi trong việc chèn dữ liệu vào những vị trí trong trang Web mà có thể gây ra những lỗi trả về phía người dùng.
- Việc xây dựng trang Web bằng XHTML sẽ nhanh hơn bởi trình duyệt sẽ không mất nhiều thời gian để dịch, và sửa lỗi .

#### 1.4 Ngôn ngữ DHTML

Khi Microsoft và Netscape đưa ra Version 4 của các trình duyệt, thì những nhà phát triển Web có một lựa chọn mới: Dynamic HTML (DHTML). Trong thực tế nó là một tập hợp gồm HTML, [Cascading Style Sheets](#) (CSS), và JavaScript. Tập hợp các công nghệ trên cho phép các nhà phát triển sửa đổi nội dung và cấu trúc của một trang Web một cách nhanh chóng.

DHTML yêu cầu sự hỗ trợ từ các trình duyệt. Mặc dù cả Internet Explorer và Netscape đều hỗ trợ DHTML, nhưng cách thể hiện của chúng là khác nhau, vì vậy các nhà phát triển cần phải biết được loại trình duyệt nào mà phía client dùng. DHTML thật sự là một bước tiến mới. Hiện nay DHTML vẫn đang trên con đường phát triển mạnh.

DHTML giúp tăng cường tính tương tác của các đối tượng điều khiển trong trang HTML tĩnh bằng cách cho phép người dùng VBscript hoặc Javascript điều khiển chúng. Ví dụ một thẻ image để nhúng ảnh vào trang web có thể nhận biết khi người dùng di chuyển chuột trên nó bằng cách cài đặt hàm xử lý sự kiện OnMouseOver, khi đó thông qua những xử lý thích hợp sẽ làm đối tượng hình ảnh trở nên sống động hơn.

Nhìn chung, bên cạnh những mở rộng như tạo những hiệu ứng MouseOver, chuỗi chữ di chuyển động, thay đổi màu sắc,... Các khía cạnh bảo mật của DHTML tương tự như HTML vì nó dựa trên nền tảng HTML.

#### 1.3 Ngôn ngữ đánh dấu siêu văn bản (HTML)

##### 1.1 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.

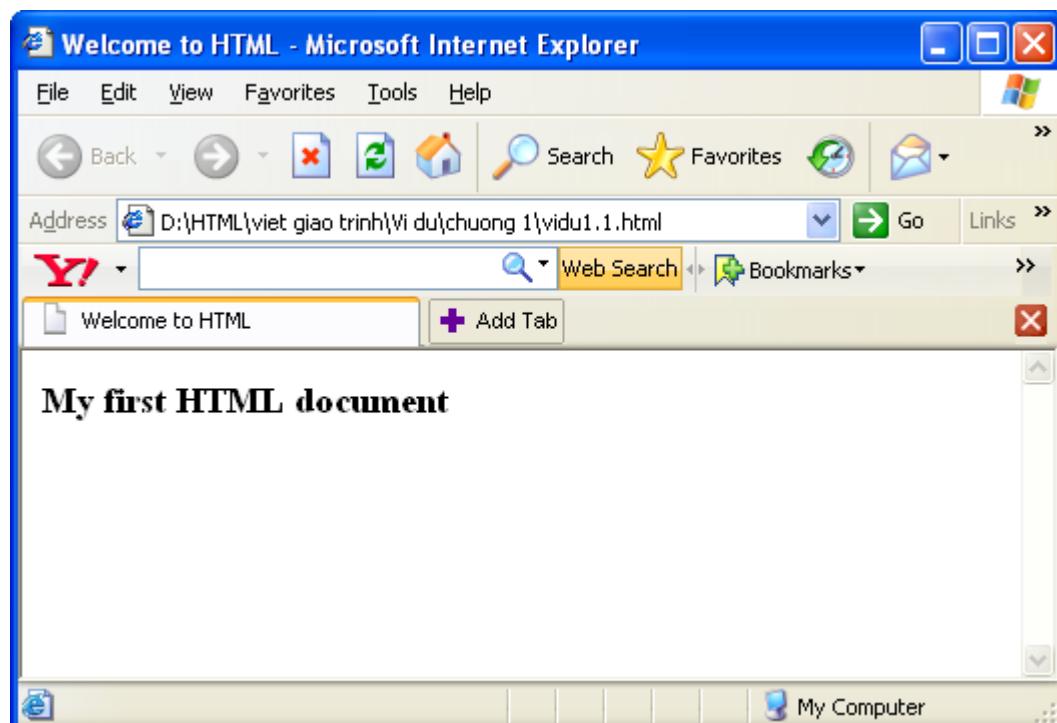
- Xuất bản các tài liệu trực tuyến và truy xuất thông tin trực tuyến bằng cách sử dụng các liên kết được chèn vào tài liệu HTML.
- Tạo các biểu mẫu trực tuyến để thu thập thông tin về người dùng, quản lý các giao dịch...
- Chèn các đối tượng như audio clip, video clip, các thành phần ActiveX và các Java 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, trình duyệt sẽ hiển thị thông điệp “My first HTML document”

### Ví dụ 1.1:

```
<HTML>
  <HEAD>
    <TITLE>Welcome to HTML </TITLE>
  </HEAD>
  <BODY>
    <H3>My first HTML document </H3>
  </BODY>
</HTML>
```

### Kết quả:



**Hình 1.1: Kết quả ví dụ 1.1**

## 1.2 Đặc điểm của HTML

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 Navigator và Microsoft 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 (“<>”) và thẻ đóng (“</>”), 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 ko phân biệt chữ hoa và chữ thường, vì thế bạn có thể sd <html> thay cho <HTML>

Thẻ HTML bao gồm:

<ELEMENT ATTRIBUTE = value>

ELEMENT: nhận dạng thẻ

ATTRIBUTE: 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 là thuộc tính màu nền 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 Cấu trúc của một tài liệu HTML

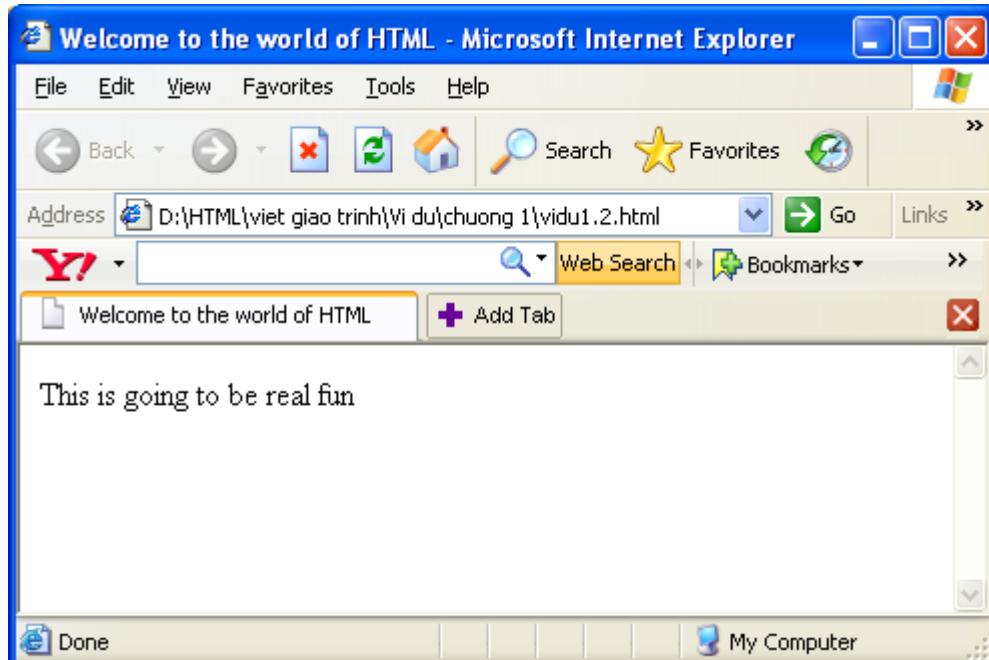
Một tài liệu HTML gồm 3 phần cơ bản:

- Phần HTML: Mọi tài liệu HTML phải bắt đầu bằng thẻ mở <HTML> và kết thúc bằng thẻ đóng </HTML>. Cặp thẻ này báo cho trình duyệt biết nội dung giữa chúng là một tài liệu HTML
- Phần đầu: Phần đầu bắt đầu bằng thẻ <HEAD> và kết thúc bởi thẻ </HEAD>. Phần này chứa tiêu đề hiển thị trên thanh điều hướng của trang Web. Tiêu đề là phần khá quan trọng. Các mốc được dùng để đánh dấu một Website, 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 khoá 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 1.2:

```
<HTML>
  <HEAD>
    <TITLE>Welcome to the world of HTML </TITLE>
  </HEAD>
  <BODY>
    <P> This is going to be real fun </P>
  </BODY>
</HTML>
```

### Kết quả:



**Hình 1.2: Kết quả ví dụ 1.2**

#### 1.4 Qui trình tạo một tài liệu HTML

- *Định hình trang Web*

Để thiết kế một trang Web, trước tiên chúng ta cần phân tích và định hướng mục đích của trang Web. Điều này giúp ta có cái nhìn tổng quát về trang Web và sẽ thuận lợi cho việc tổ chức hay nâng cấp trang Web sau này.

Những yêu cầu cần phải nghiên cứu.

- Hình dung nội dung trang Web bạn cần tạo, hướng tới một đích chung cho trang Web với những chức năng và nhiệm vụ gì?
- Đặt mình vào vị trí người xem, khách hàng. Làm thế nào để nội dung trình bày thể hiện tốt nhất. Ví dụ bạn có thể thêm vào âm thanh, hình ảnh minh họa cho sinh động, bố cục nội dung, trình bày sao cho hợp lý nhất.

- *Tổ chức tập tin*

Các yếu tố làm nên trang Web đó là các tập tin, do vậy việc tổ chức tập tin là rất quan trọng, nó giúp ta thuận lợi trong việc lưu trữ tìm kiếm các đoạn mã hay cơ sở dữ liệu của trang Web.

Chia các thư mục trung tâm theo cấu trúc của trang Web, bạn có thể tạo một thư mục riêng rẽ cho tài liệu HTML: các hình ảnh, cơ sở dữ liệu, các tập tin bên ngoài,... Trong trường hợp trang Web lớn với nhiều trang, bạn có thể chia thành nhiều mục hay chương, chuyển các hình ảnh đến thư mục độc lập.

- *Tạo trang Web*

Để tạo một trang Web HTML chúng ta không cần một công cụ đặc biệt nào, chỉ cần sử dụng bất kỳ bộ soạn thảo văn bản nào như Wordpad hay Notepad, được cung cấp kèm theo hệ phần mềm Windows.

Dựa trên qui định về cấu trúc của một trang Web, kết hợp với các thẻ cần thiết để viết ra trang Web của mình.

- *Lưu trang Web*

Nếu ta sử dụng một trình xử lý văn bản đơn giản để tạo trang Web bạn sẽ không có vấn đề gì khi lưu trang Web. Nhưng khi ta dùng một trình xử lý văn bản phức tạp thì bạn phải lưu ý những thông tin bên ngoài mà chương trình sẽ đính kèm vào tập tin của bạn. Để đảm bảo mọi trình duyệt sẽ nhận diện được tập tin đó, bạn phải đặt phải đặt đuôi của tập tin đúng.

- Mỗi tập tin được lưu phần đuôi của nó có dạng .htm hay .html
- Chọn thư mục thích hợp để lưu trang Web.
- Xem trang Web qua trình duyệt

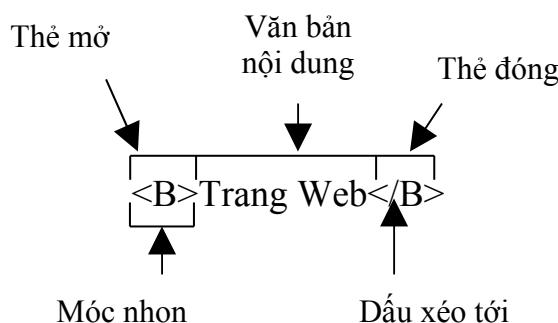
Khi đã tạo ra trang Web chúng ta cần xem nó như thế nào qua một trình duyệt thông thường là Internet Explorer.

## CHƯƠNG 2

# CÁC THẺ CƠ BẢN TRONG HTML

### 2.1 Thẻ và các thuộc tính của thẻ

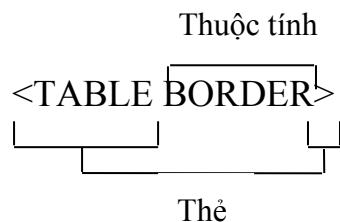
Thẻ là những câu lệnh được viết giữa dấu nhỏ hơn (<) và dấu lớn hơn (>) hay còn gọi là dấu móc nhọn, quy định cách hiển thị văn bản. Có 2 loại thẻ: Thẻ **mở** và thẻ **đóng**, đoạn văn bản hiển thị nằm giữa hai thẻ này, cả thẻ mở và thẻ đóng đều được viết như nhau nhưng thẻ đóng có thêm một dấu / (dấu xéo tới) phía trước.



**Hình 2.1: Phân tích một thẻ HTML**

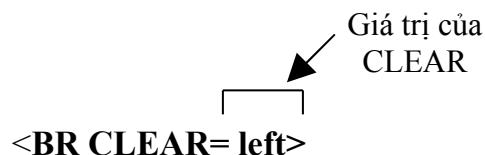
#### ❖ Thuộc tính của thẻ

Thuộc tính tác động lên nội dung văn bản. Thuộc tính được nhập vào giữa thẻ và dấu lớn hơn cuối cùng. Thường thì chúng ta dùng nhiều thuộc tính trong một thẻ. Các thuộc tính được viết không cần thứ tự và cách nhau một khoảng trống.



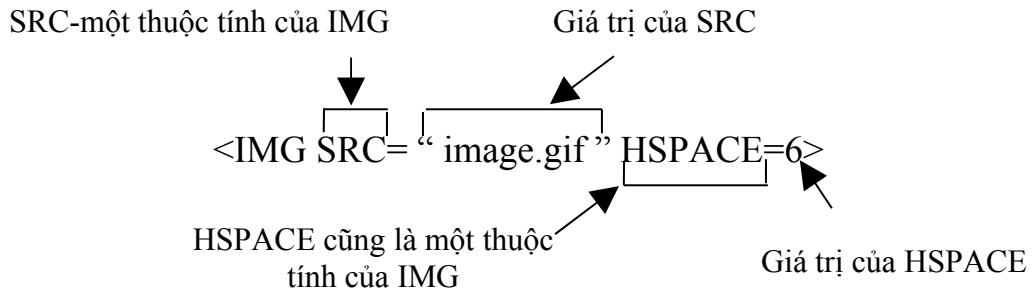
**Hình 2.2: Thẻ có thể thêm vào các thuộc tính theo định dạng của người viết**

Thuộc tính thường đi kèm với các giá trị. Trong một số trường hợp, có thể lựa chọn giữa các giá trị. Ví dụ: thuộc tính CLEAR trong thẻ Br có thể nhận được các giá trị *left* (trái), *right* (phải), hay *all* (cả hai bên). Tất cả các giá trị khác được đưa vào sẽ không được chấp nhận.



**Hình 2.3 : Thẻ BR chỉ chấp nhận những thuộc tính với một giá trị định sẵn**

Các thuộc tính khác còn xét đến dạng giá trị mà chúng có thể chấp nhận. Ví dụ: thuộc tính HSPACE trong thẻ IMG chỉ chấp nhận các số nguyên làm giá trị.



**Hình 2.4: Thẻ có thể thêm nhiều thuộc tính khác nhau**

Giá trị được đặt giữa hai dấu nháy (""). Có thể bỏ qua dấu nháy nếu giá trị chỉ chứa chữ (A-Z, a-z), số (0-9), dấu gạch nối () hoặc dấu chấm (.)

#### ❖ Thẻ lồng nhau

Thẻ lồng nhau dùng để chỉnh sửa cách trình bày nội dung trang, ví dụ định dạng chữ nghiêng cho một vài chữ quan trọng trong đề mục.

#### *Lưu ý:*

Trật tự các thẻ lồng nhau: Thẻ được mở đầu tiên sẽ là thẻ đóng cuối cùng.

Ví dụ: <B> Phần 1:<I> Nội dung</I></B>

Kết quả là: **Phần 1: Nội dung**

## 2.2 Các thẻ cơ bản trong HTML

### 2.2.1 Thẻ xác định văn bản HTML

#### *Cú pháp:*

`<html>` Các nội dung của văn bản HTML `</html>`

### 2.2.2 Thẻ xác định phần đầu cho trang HTML

#### *Cú pháp:*

`<head>` Phần đầu văn bản HTML `</head>`

Thẻ xác định phần đầu của văn bản HTML, thông thường thì thẻ để tạo tiêu đề trang được đặt lồng vào trong thẻ này.

### 2.2.3 Thẻ xác định tiêu đề cho văn bản HTML

#### *Cú pháp:*

`<title>` Nội dung tiêu đề của văn bản HTML `</title>`

### 2.2.4 Thẻ xác định phần thân cho trang HTML

#### *Cú pháp:*

<body> Phần thân văn bản HTML </body>

Tất cả nội dung của trang web được nằm giữa hai thẻ này.

### Ví dụ 2.1:

<HTML>

<HEAD>

<TITLE> tiêu đề </TITLE>

</HEAD>

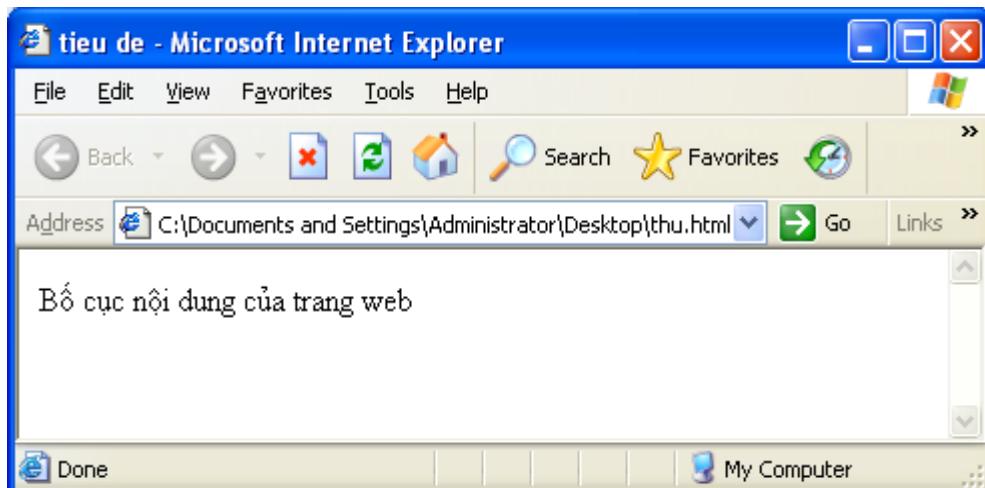
<BODY>

Bộ cục nội dung của trang web

</BODY>

</HTML>

### Kết quả:



**Hình 2.5: Thẻ Body xác định phần thân của trang web**

#### 2.2.5 Thẻ xác định các danh mục trang web

Các danh mục thường được hiển thị to và đậm hơn để phân biệt chúng với các phần còn lại của văn bản. Chúng ta có thể hiển thị các danh mục này theo một trong sáu kích thước từ H1 đến H6 như trong ví dụ sau:

### Ví dụ 2.2:

<HTML>

<HEAD>

<TITLE> Introduction to HTML </TITLE>

</HEAD>

<H1> Introduction to HTML </H1>

<H2> Introduction to HTML </H2>

<H3> Introduction to HTML </H3>

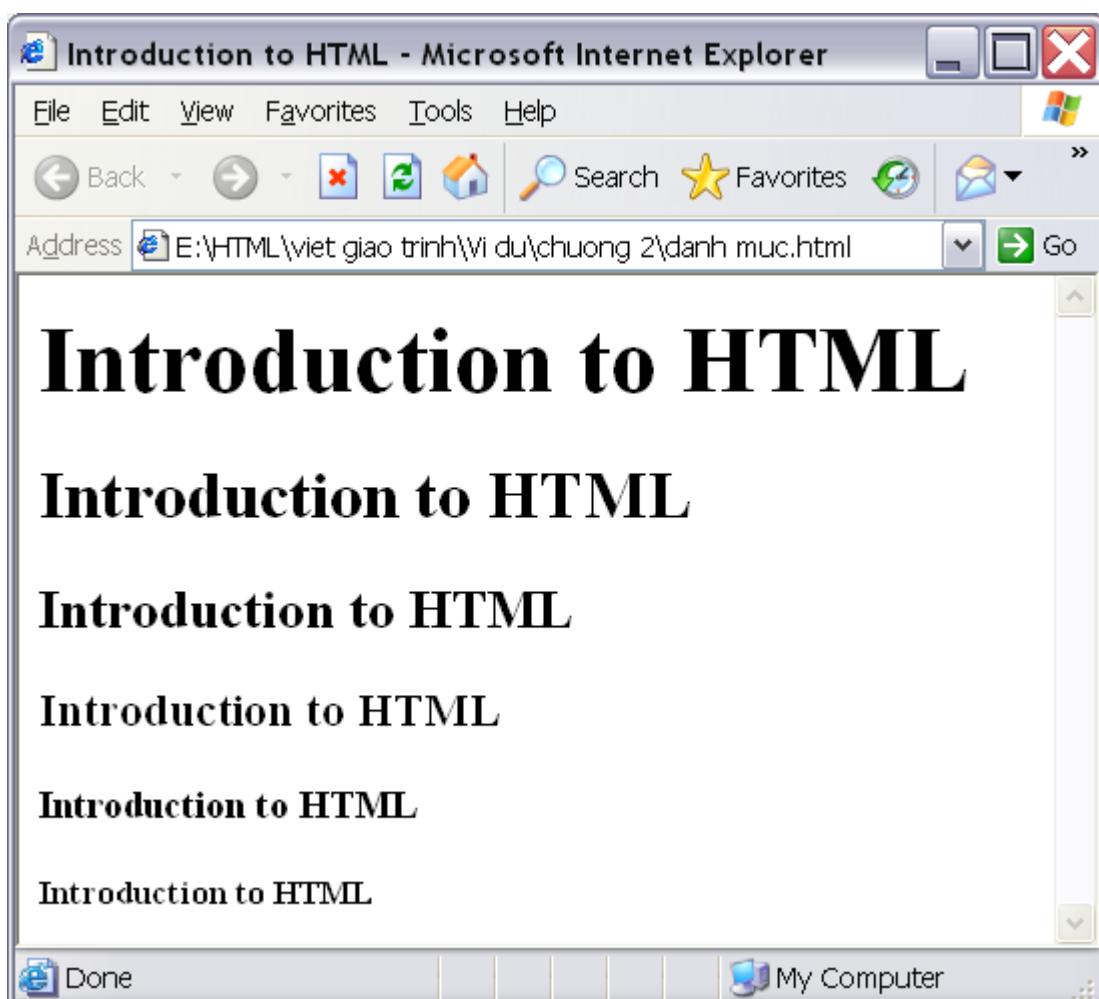
<H4> Introduction to HTML </H4>

```

<H5> Introduction to HTML </H5>
<H6> Introduction to HTML </H6>
</HTML>

```

**Kết quả:**



**Hình 2.6: Minh họa thẻ xác định các danh mục cho trang web**

#### 2.2.6 Thẻ tạo 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
- Oranges
- Apples
- Orchids
- Mangoes

Có thể được nhóm thành:

Fruits  
    Apples  
    Oranges  
    Mangoes  
  
Flowers  
    Roses  
    Sunflowers  
    Orchids

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

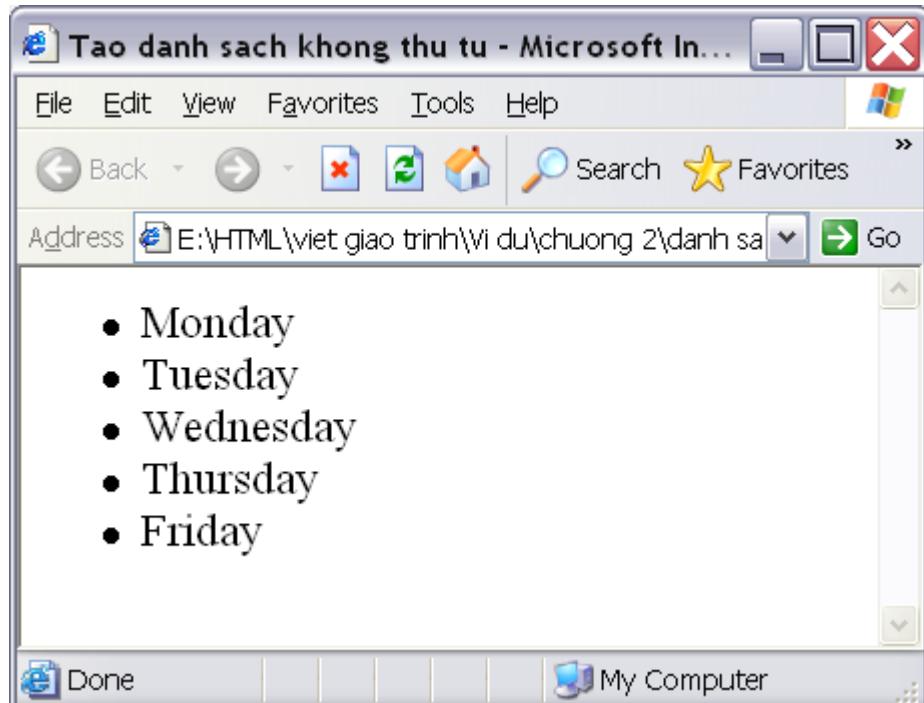
#### 2.2.6.1 *Thẻ tạo danh sách không thứ tự*

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 `<UL>... </UL>`. Mỗi mục trong danh sách được đánh dấu bằng thẻ `<LI>`. LI được viết tắt của từ List Item. Thẻ kết thúc `</LI>` là tùy chọn (không bắt buộc).

#### Ví dụ 2.3:

```
<HTML>
    <HEAD>
        <TITLE> Tao danh sach khong thu tu </TITLE>
    </HEAD>
    <BODY>
        <UL>
            <LI> Monday
            <LI> Tuesday
            <LI> Wednesday
            <LI> Thursday
            <LI> Friday
        </UL>
    </BODY>
</HTML>
```

**Kết quả:**



**Hình 2.7: Minh họa thẻ tạo danh sách không thứ tự trong trang web**

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

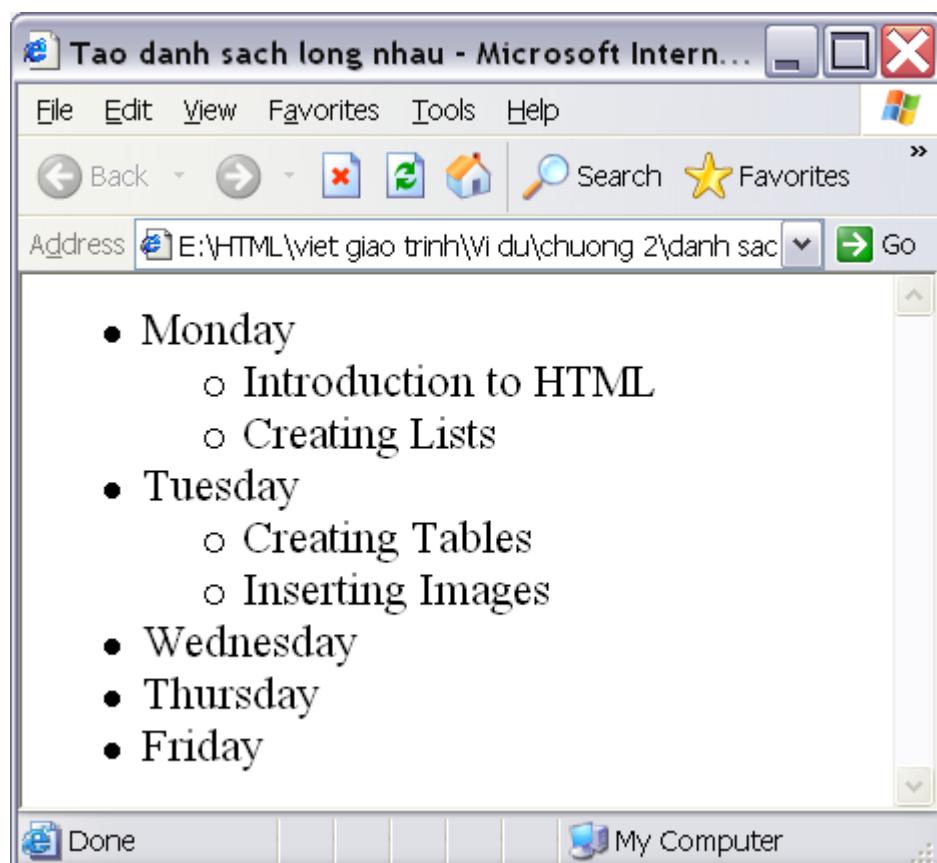
**Ví dụ 2.4:**

```
<HTML>
  <HEAD>
    <TITLE> Tao danh sach long nhau </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>
    </UL>
```

```

<LI> Wednesday
<LI> Thursday
<LI> Friday
</UL>
</BODY>
</HTML>

```

**Kết quả:****Hình 2.8: Minh họa cách tạo danh sách lồng nhau****2.2.6.2 Thẻ tạo danh sách có thứ tự**

Danh sách có thứ tự nằm trong cặp thẻ `<OL>... </OL>`. 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 2.5:**

```

<HTML>
  <HEAD>
    <TITLE> Tao danh sach co thu tu </TITLE>
  </HEAD>
  <BODY>
    <OL>

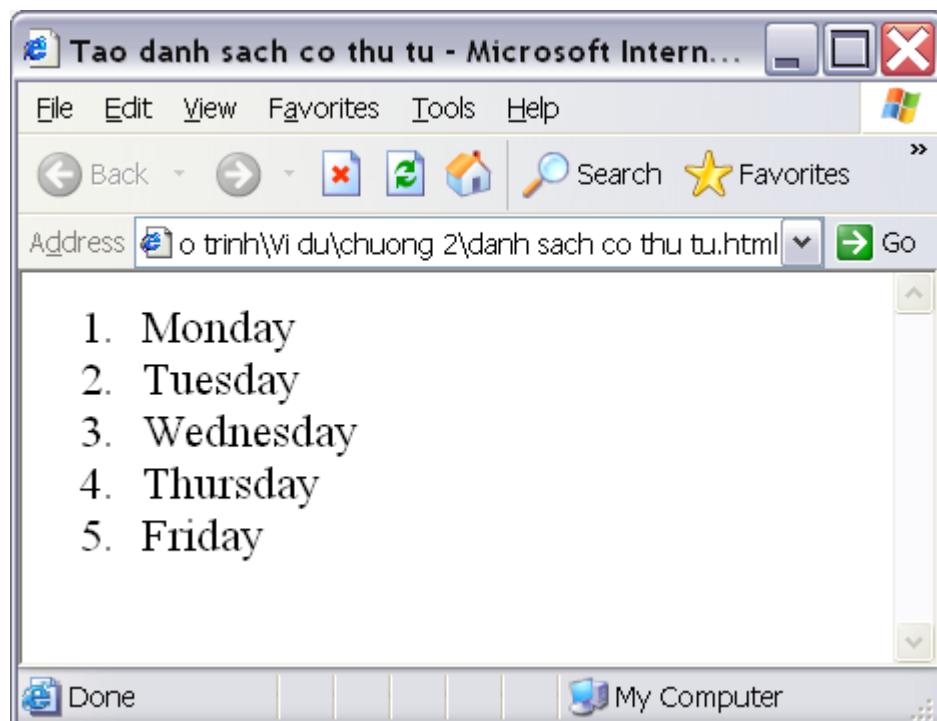
```

```

<LI> Monday
<LI> Tuesday
<LI> Wednesday
<LI> Thursday
<LI> Friday
</OL>
</BODY>
</HTML>

```

**Kết quả:**



**Hình 2.9: Minh họa cách tạo danh sách có thứ tự**

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

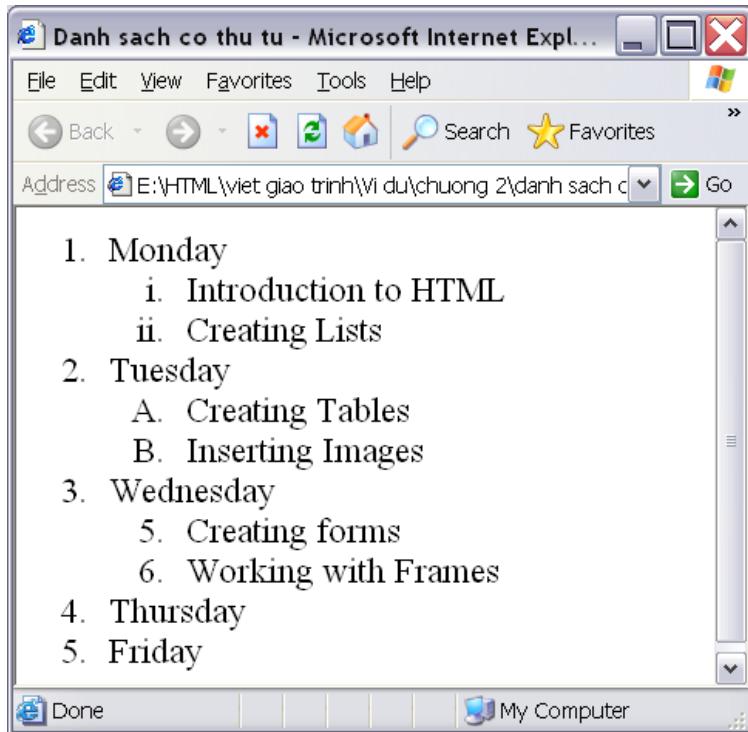
**Bảng 2.1: Các thuộc tính dùng để định nghĩa hệ thống số**

Thuộc tính	Thẻ
Upper Roman	<LI TYPE = I>
Lower Roman	<LI TYPE = i>
Uppercase	<LI TYPE = A>
Lowercase	<LI TYPE = a>
Bắt đầu với một số khác lớn hơn 1	<OL START = n>

Trong đó thuộc tính START xác định số khởi tạo ban đầu của danh sách.

**Ví dụ 2.6:**

```
<HTML>
    <HEAD>
        <TITLE> Danh sach co thu tu </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
        </UL>
    </BODY>
</HTML>
```

**Kết quả:****Hình 2.10: Minh họa cách tạo danh sách có thứ tự**

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.

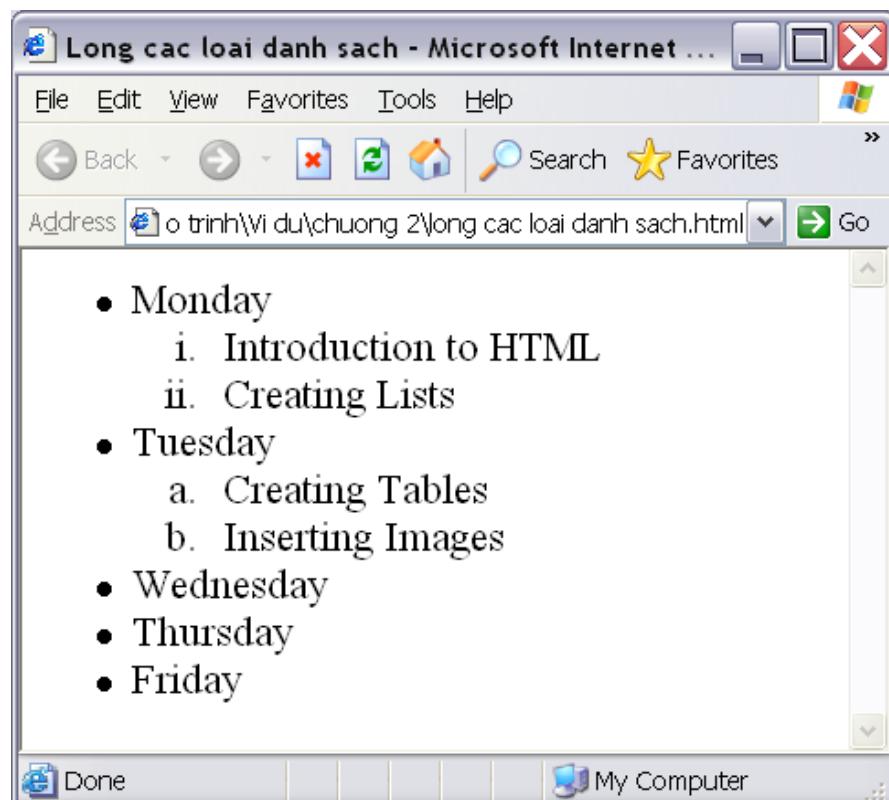
**Ví dụ 2.7:**

```
<HTML>
  <HEAD>
    <TITLE> Long cac loai danh sach </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>

```

**Kết quả:****Hình 2.11: Lồng các loại danh sách****2.2.6.3 Thẻ tạo danh sách định nghĩa**

Danh sách định nghĩa được dùng để tạo ra một danh sách các điều khoản và các định nghĩa của chúng. Danh sách định nghĩa nằm trong cặp thẻ `<DL>... </DL>`. Thẻ `<DT>` được dùng để chỉ ra điều khoản còn thẻ `<DD>` được dùng để chỉ ra định nghĩa cho điều khoản đó.

**Ví dụ 2.8:**

```

<HTML>
  <HEAD>
    <TITLE> Danh sach dinh nghia </TITLE>
  </HEAD>
  <BODY>
    <DL>
      <DT> Sunday

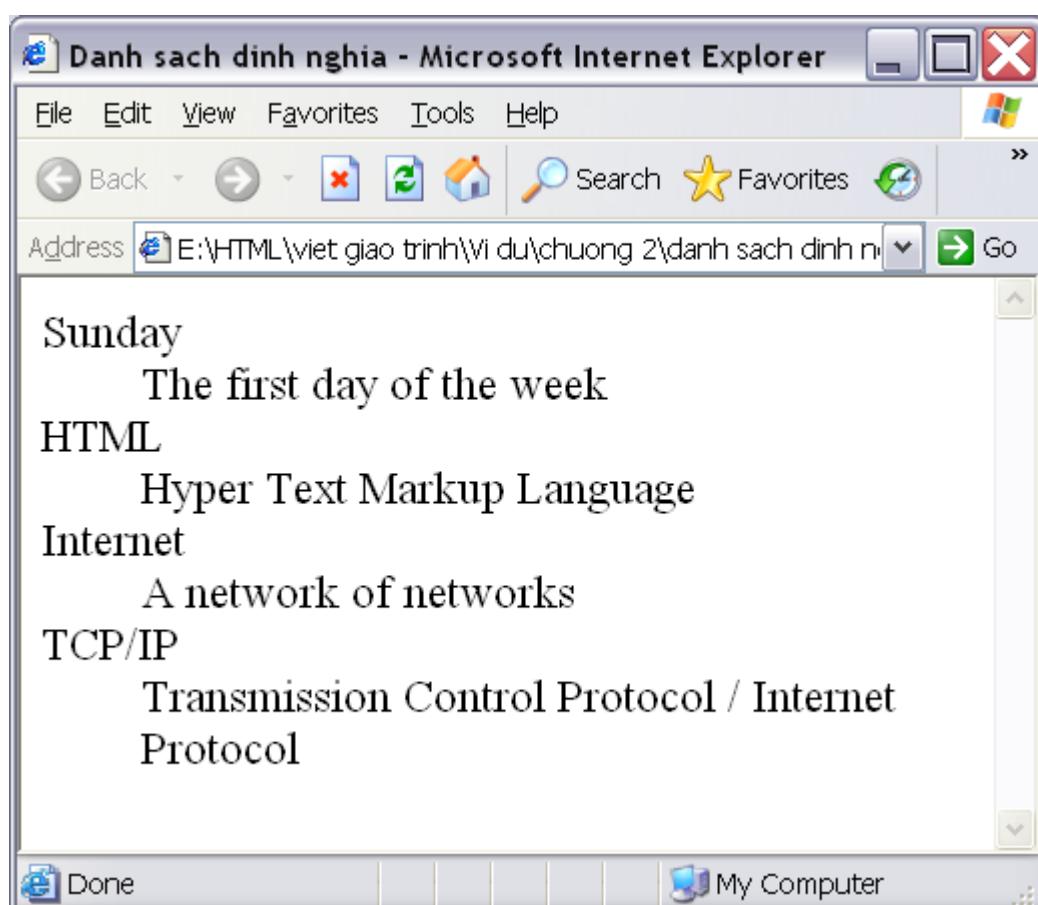
```

```

<DD> The first day of the week
<DT> HTML
<DD> Hyper Text Markup Language
<DT> Internet
<DD> A network of networks
<DT> TCP/IP
<DD> Transmission Control Protocol / Internet Protocol
</DL>
</BODY>
</HTML>

```

**Kết quả:**



**Hình 2.12: Danh sách định nghĩa**

### 2.2.7 Thẻ xác định văn bản trang web

**Cú pháp:**

`<p> Nội dung của văn bản </p>`

Khi cần trình bày một nội dung văn bản nào đó, chúng ta đặt các văn bản nằm trong thẻ p.

### 2.2.8 Thẻ tạo đường thẳng

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

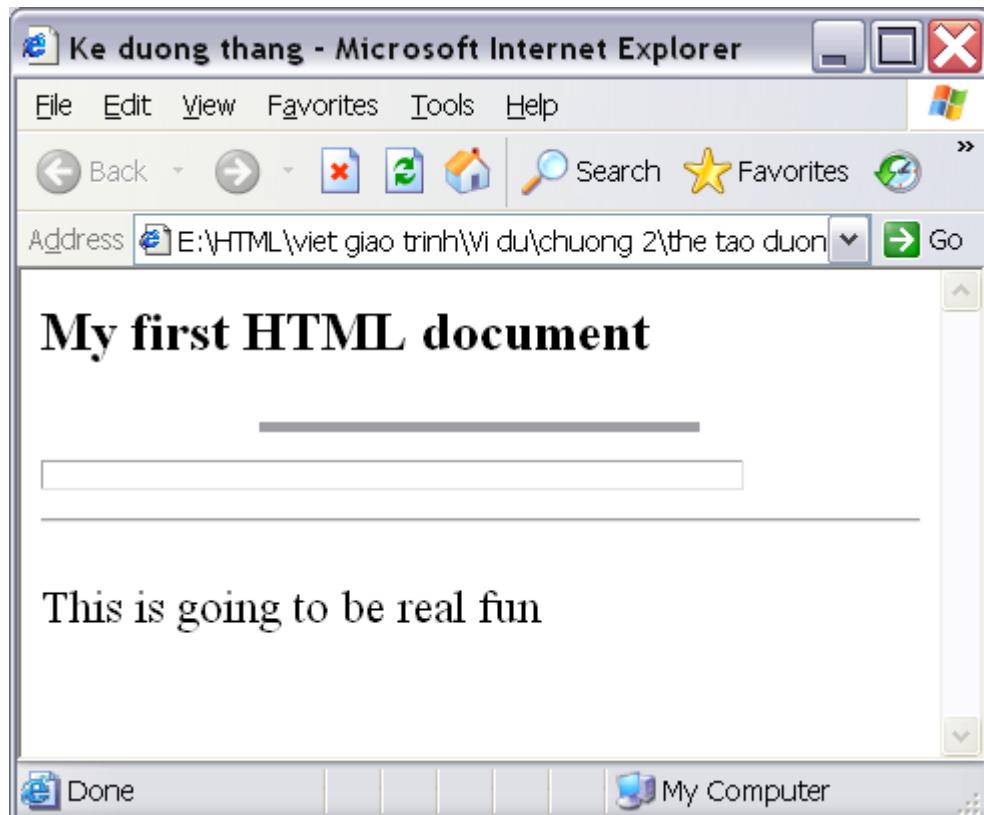
**Bảng 2.2: Các thuộc tính của thẻ HR**

Thuộc tính	Mô tả
align	Chỉ định vị trí của đường thẳng. Chúng ta có thể canh lề center (giữa) hoặc right (phải) hoặc 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 2.9:

```
<HTML>
  <HEAD>
    <TITLE> Ke duong thang </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>
```

**Kết quả:**



**Hình 2.13: Minh họa thẻ tạo đường thẳng**

### 2.2.9 Thẻ xác định dòng chú thích

**Cú pháp:**

<!--> : Xác định một vùng **chú thích**

Khi chú thích cho phần nguồn của trang web, chúng ta sử dụng thẻ này để tạo ra vùng chú thích mà không cho phép hiển thị trên nội dung của trang web.

## 2.3 Các thẻ vận dụng với văn bản

### 2.3.1 Thẻ vận dụng cho kiểu chữ

#### 2.3.1.1 Làm chữ đậm

Để in đậm một đoạn văn bản chúng ta có các thẻ sau để thực hiện.

**Cú pháp:**

<b> Nhập văn bản vào bạn muốn làm đậm</b>

hoặc

<Strong> Nhập đoạn văn bản bạn muốn làm đậm</Strong>

**Ví dụ 2.10:**

<HTML>

<HEAD>

<TITLE> Chu dam </TITLE>

```

</HEAD>
<BODY>
    <B> This is a bold line text
</BODY>
</HTML>

```

**Kết quả:**



**Hình 2.14: Hiển thị nội dung dòng chữ đậm trên trình duyệt**

### 2.3.1.2 Làm chữ in nghiêng

Để in nghiêng một đoạn văn bản ta dùng thẻ *<i>* như sau

**Cú pháp:**

*<i>* Nhập văn bản vào bạn muốn in nghiêng *</i>*

**Ví dụ 2.11:**

<HTML>

```

<HEAD>
    <TITLE> Chu nghieng </TITLE>
</HEAD>
<BODY>
    <i> This is an italic line text
</BODY>
</HTML>

```

**Kết quả:**



**Hình 2.15: Hiển thị nội dung dòng chữ nghiêng trên trình duyệt**

#### 2.3.1.3 Thay đổi kích thước chữ

Để thay đổi cỡ chữ tương đối của một từ hay một nhóm từ so với các văn bản xung quanh, ta dùng cú pháp sau:

**Cú pháp:**

<big> Nhập văn bản vào bạn muốn tăng cỡ chữ lớn hơn </big>  
 <small> Nhập văn bản bạn muốn giảm cỡ chữ bé hơn <small>

#### 2.3.1.4 Tạo dòng chữ thấp

Để tạo các dòng chữ thấp tương ứng với chỉ số dưới ta dùng thẻ Sub sau đây.

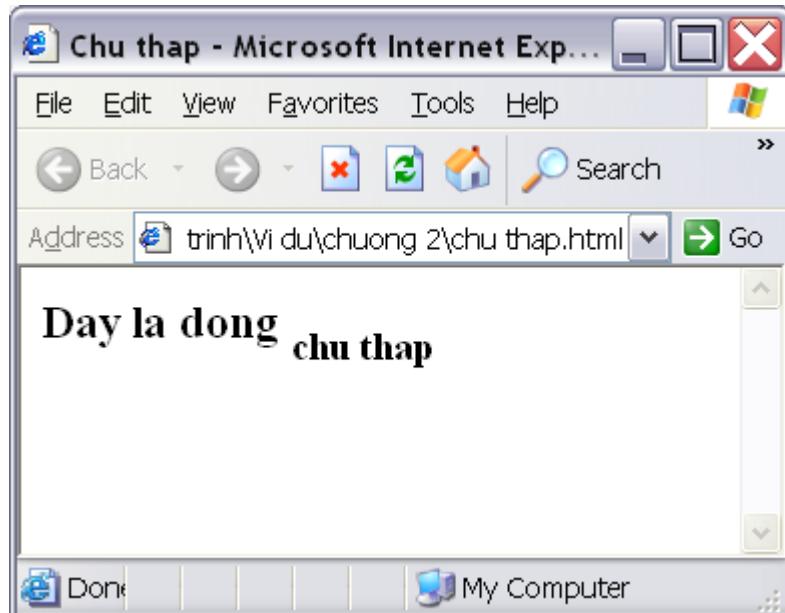
**Cú pháp:**

<sub> Nhập chữ hay kí hiệu bạn muốn chỉnh thấp </sub>

#### Ví dụ 2.12:

```
<HTML>
  <HEAD>
    <TITLE> Chu thap </TITLE>
  </HEAD>
  <BODY>
    <B> Day la dong <sub> chu thap </sub>
  </BODY>
</HTML>
```

Kết quả:



**Hình 2.16: Minh họa thẻ sub để tạo dòng chữ thấp**

#### 2.3.1.5 Tạo dòng chữ cao

Để tạo dòng chữ cao tương ứng với chỉ số trên ta dùng thẻ sup như sau:

**Cú pháp:**

<sup> Nhập chữ hay kí hiệu bạn muốn chỉnh cao </sup>

**Ví dụ 2.13:**

```
<HTML>
  <HEAD>
    <TITLE> Chu thap </TITLE>
  </HEAD>
  <BODY>
    <B> Day la dong <sup> chu cao </sup>
  </BODY>
</HTML>
```

**Kết quả:**



**Hình 2.17: Minh họa thẻ sup để tạo dòng chữ cao**

#### 2.3.1.6 Gạch ngang và gạch dưới chữ

**Cú pháp:**

<Strike> Nhập văn bản cần gạch bỏ </Strike>  
<u> Nhập đoạn văn bản cần gạch dưới </u>

#### 2.3.1.7 Tạo chữ dạng riêng

Để nhấn mạnh hay làm nổi bật đoạn văn bản ta dùng thẻ sau.

**Cú pháp:**

<em> Đoạn văn bản cần nhấn mạnh </em>

#### 2.3.1.8 Tạo dạng chữ bị xoá

**Cú pháp:**

<del> Nhập vào đoạn văn bản cần xoá </del>  
<s> Nhập đoạn văn bản cần xoá </s>

#### 2.3.1.9 Tạo dạng chữ chèn vào

**Cú pháp:**

<ins> định dạng chữ mới chèn thêm </ins>

### 2.3.2 Thẻ vận dụng cho hiệu ứng font chữ

#### 2.3.2.1 Chọn font chữ cho văn bản

Để chọn font chữ cho đoạn văn bản cần trình bày ta dùng thẻ font như sau

**Cú pháp:**

<font face =“fontname1, fontname2”> nhập văn bản cần hiển thị văn bản đã chọn</font>

**fontname2** là kiểu chữ ưu tiên thứ 2 nếu như người truy cập không cài đặt kiểu chữ thứ nhất. Mỗi tên kiểu chữ phải được ngăn cách với tên đứng trước bằng dấu phẩy. Có thể thêm **fontname3, fontname4...** cho các kiểu chữ ưu tiên tiếp theo.

Ví dụ: <Font face =“Times new roman, Arial”> Văn bản cần định dạng bởi font chữ</font>.

**fontname1** là kiểu chữ được chọn đầu tiên. Gõ tên đầy đủ của kiểu chữ mà ta muốn.

### 2.3.2.2 *Đổi cỡ chữ văn bản*

Khi cần thay đổi cỡ một số chữ, chúng ta có thể làm theo hai cách: chọn cỡ chữ ngay hay điều chỉnh cho vùng chữ được to hơn hay nhỏ hơn các chữ xung quanh.

#### Cú pháp:

<font size= “n”> nhập văn bản mà bạn cần điều chỉnh cỡ chữ </font>

Giá trị của n nhận từ 1 đến 7.

#### Ví dụ 2.14:

<HTML>

<HEAD>

<TITLE> Chon font chu </TITLE>

</HEAD>

<BODY>

<font face = Arial, size = 3> This is text set by Arial font </font>

<p>

<font face = Times new roman, size = 3> This is text set by Times new roman font </font>

</BODY>

</HTML>

#### Kết quả:



### Hình 2.18: Minh họa thẻ font

#### 2.3.2.3 Chọn cỡ chữ mặc định.

Khi muốn thống nhất một cỡ chữ nhất định trên trang Web, ta thường sử dụng thẻ sau đây.

##### Cú pháp:

```
<basefont size=“n”>
```

n nhận giá trị từ 1 đến 7, cỡ chữ mặc định là 3.

#### 2.3.2.4 Đổi màu chữ

Một trong những cách làm nổi bật văn bản là đổi màu chúng. Bạn có thể đổi một phần văn bản thành màu khác và phần còn lại là màu đen.

##### Cú pháp:

```
<font color = “#rrggbb”> Nhập văn bản bạn muốn đổi màu </font>
```

Trong đó: **rrggbb** là số thập lục phân biểu hiện màu mong muốn, **rr** là giá trị thập lục phân giành cho màu đỏ, **gg** cho xanh lá cây, **bb** cho xanh dương.

Ví dụ:

`<FONT COLOR = “308F9E”> Tương ứng R là 48 (hệ 16=30), G là 148 (hệ 16=8F), và B là 158 (hệ 16=9E) do đó giá trị trong hệ thập lục phân tương ứng là 308F9E. Tuy nhiên ta có thể thay đổi các giá trị R, G, B trong bảng màu để được các giá trị màu khác nhau, hoặc là:`

```
<font color= “color”> Nhập văn bản bạn muốn đổi màu </font>
```

color là 1 trong 16 màu định sẵn.

Ví dụ:

```
<FONT COLOR= “Red”> , đoạn văn bản được tác động bởi thẻ sẽ có màu đỏ.
```

#### 2.3.2.5 Làm chữ nhấp nháy

Để tạo ra các hiệu ứng về chữ như nhấp nháy ta có thể dùng với thẻ sau

##### Cú pháp:

```
<blink> Nhập đoạn văn bản cần nhấp nháy </blink>
```

### 2.4 Thẻ vận dụng trình bày trang Web

#### 2.4.1 Lựa chọn màu nền

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

##### Ví dụ 2.15:

```
<HTML>
```

```
    <HEAD>
```

```
        <TITLE> Sử dụng màu nền </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>

```

**Kết quả:**



**Hình 2.19: Sử dụng màu nền**

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.

**Bảng 2.3: Bảng mã một số màu**

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

#### 2.4.2 Lựa chọn hình ảnh làm nền

Chúng ta có thể dùng hình ảnh làm nền cho toàn bộ trang. Hình ảnh phải phù hợp với nội dung trang và làm cho trang thêm hấp dẫn.

**Cú pháp:**

```
<body background="bgimage.jpg" bgproperties=fixed >
```

**bgimage.gif** là địa chỉ hình ảnh .

**bgproperties=fixed** để chỉnh hình ảnh thành mờ bất động, chỉ dùng trong Internet Explorer.

**Ví dụ 2.16:**

```
<HTML>
```

```
    <HEAD>
```

```
        <TITLE>nen </TITLE>
```

```
    </HEAD>
```

```
    <BODY BGCOLOR = "#808080" background = "nen.jpg" bgproperties = "fixed">
```

```
    </BODY>
```

```
</HTML>
```

**Kết quả:**

**Hình 2.20: Sử dụng hình nền**

**2.4.3 Chỉnh lề cho trang Web**

Để căn lề cho nội dung trang, ta có thể thay đổi, điều chỉnh khoảng cách đó cho phù hợp với yêu cầu thực tế bằng cách dùng các thuộc tính của thẻ BODY như sau:

**Cú pháp:**

```
<BODY LEFTMARGIN=x TOPMARGIN=y các thuộc tính khác>
```

Với x là độ rộng đơn vị **pixel** của lề trái, y là khoảng cách giữa đầu nội dung và đỉnh cửa sổ.

**2.4.4 Tạo đoạn văn bản**

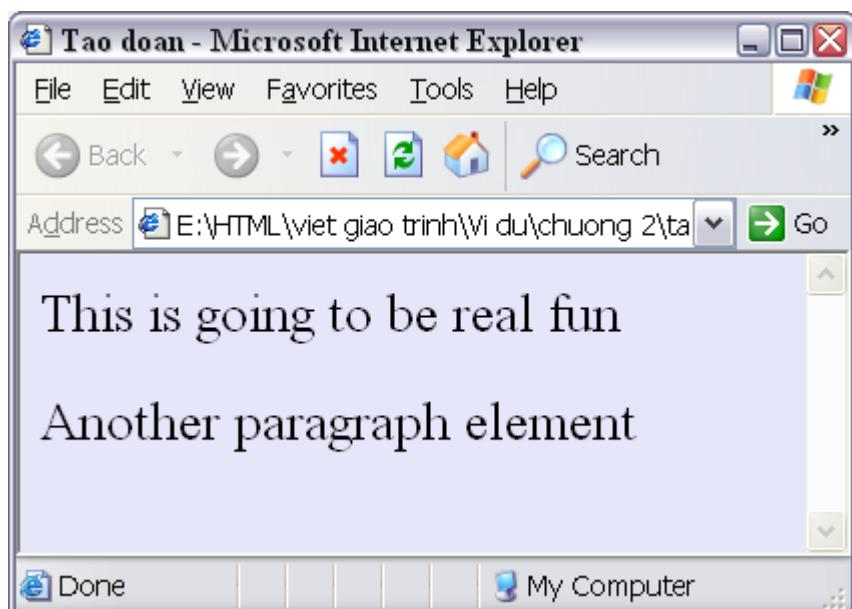
Khi viết một bài báo hay một bài luận, bạn nhóm nội dung thành một loạt các đoạn. Mục đích là nhóm các ý tưởng logic lại với nhau và áp dụng một số định dạng cho nội dung. Trong một tài liệu HTML, nội dung có thể được nhóm thành các đoạn. Thẻ đoạn <P> được sử dụng để đánh dấu sự bắt đầu của một đoạn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

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

### Ví dụ 2.17:

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

### Kết quả:



**Hình 2.21: Tạo các đoạn với thẻ <P>**

#### 2.4.5 Ngắt đoạn

Khi tạo một đoạn mới với thẻ P, hầu hết trình duyệt chèn thêm một khoảng trống lớn giữa chúng. Để bắt đầu đoạn mới mà không có khoảng trống, hãy sử dụng thẻ BR để ngắt hàng.

### Cú pháp:

<BR> Xác định vị trí cần xuống dòng, không cần thẻ BR đóng.

Tuy nhiên bạn có thể dùng nhiều thẻ **BR** để tạo ra khoảng cách giữa các dòng hay các đoạn.

## 2.5 Một số thẻ đặc biệt khác

### 2.5.1 *Thẻ làm việc với 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.

#### 2.5.1.1 *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.

Như vậy, với siêu liên kết, chúng ta có thể liên kết đến:

- 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

Các liên kết có thể là liên kết trong hoặc liên kết ngoài. Liên kết trong là liên kết nối đến các phần khác trong cùng tài liệu hoặc cùng một website. Liên kết ngoài là liên kết kết nối đến các trang trên các website khác hoặc máy chủ khác.

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 website của Microsoft. 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 website hay tài nguyên trên World Wide Web.

Định dạng đặc trưng là [www.nameofsite.typeofsite.contrycode](http://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](http://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.

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.

#### 2.5.1.2 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:

- Protocol: Giao thức.

Một số giao thức thường dùng là:

- o http - giao thức truyền siêu văn bản
- o telnet - mở một phiên telnet
- o gopher - tìm kiếm file
- o ftp - giao thức truyền file
- o mailto - gửi thư điện tử
- Host.domain: Địa chỉ Internet của máy chủ
- Port: Cổng phục vụ của máy chủ đích
- Hypertext: Văn bản hay hình ảnh mà người dùng cần nhấp vào để kích hoạt liên kết.
  - a. Liên kết đến những tài liệu khác

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

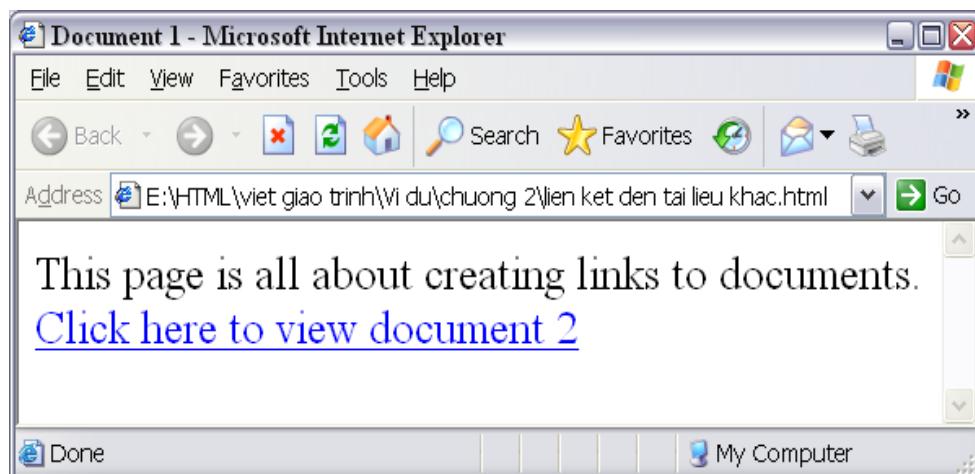
**Ví dụ 2.18:**

<HTML>

```

<HEAD>
    <TITLE> Document 1 </TITLE>
</HEAD>
<BODY>
    <P> This page is all about creating links to documents.
    <A HREF = Doc2.html> Click here to view document 2 </A>
</BODY>
</HTML>

```

**Kết quả:****Hình 2.22: Liên kết đến tài liệu khác**

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.

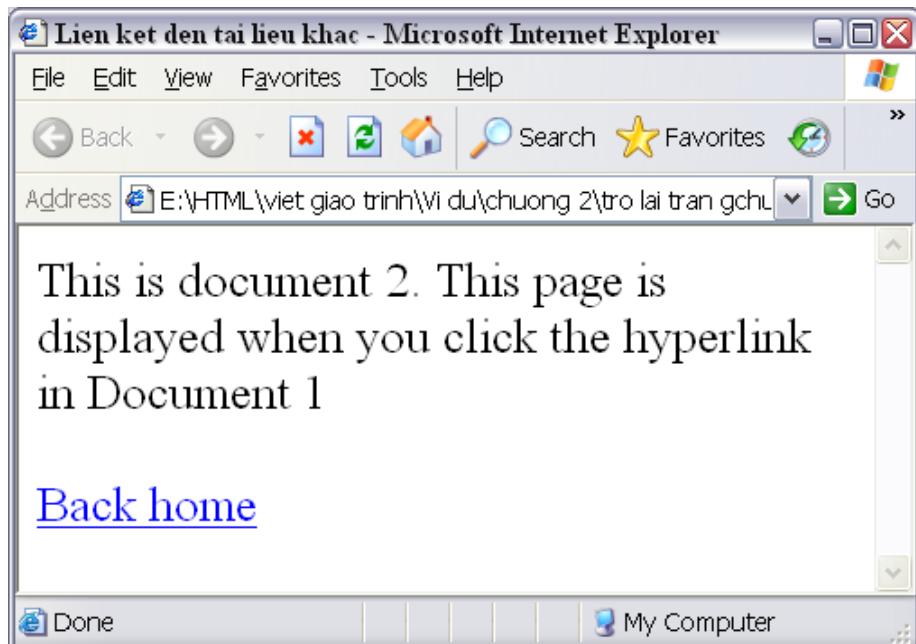
**Ví dụ 2.19:**

```

<HTML>
    <HEAD>
        <TITLE> Lien ket den tai lieu khac </TITLE>
    </HEAD>
    <BODY>
        <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>

```

**Kết quả:**



**Hình 2.23: Trở lại trang trước**

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\Doc2.html

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.html”> 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 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.

#### b. 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 được 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.

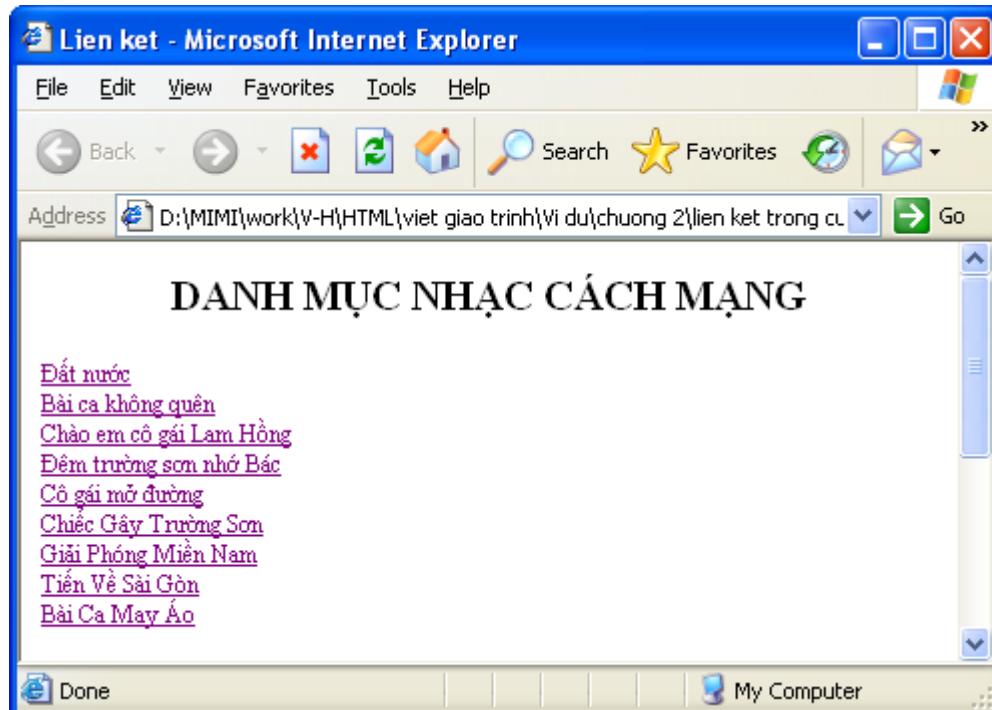
### Ví dụ 2.20:

<HTML>

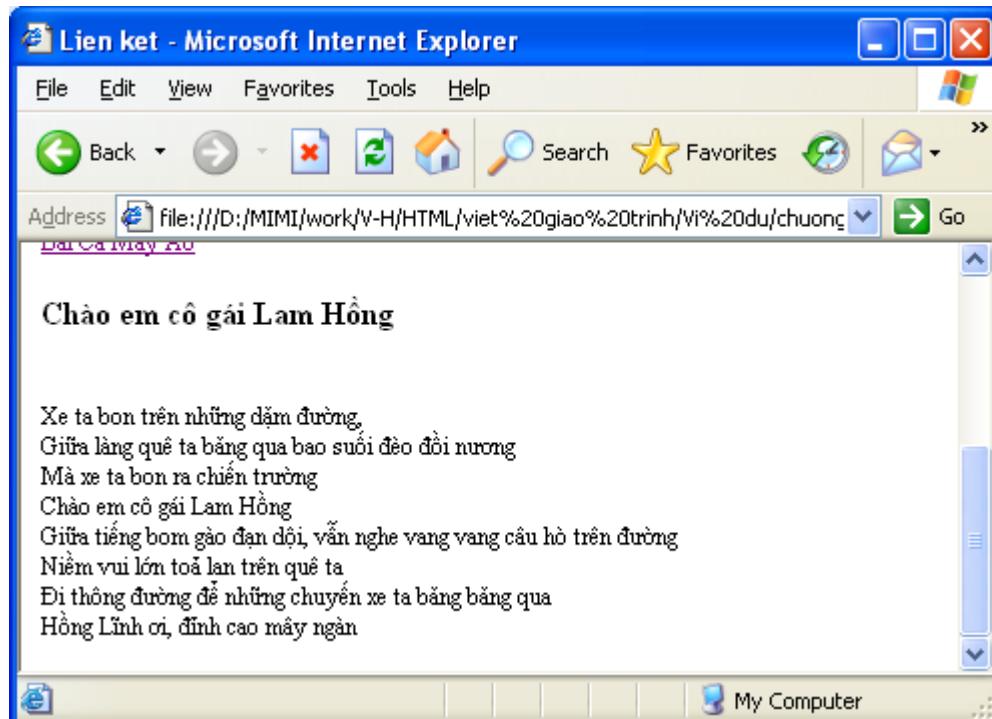
```

<HEAD>
    <TITLE> Lien ket</TITLE>
</HEAD>
<BODY>
    <H1><center>DANH MỤC NHẠC CÁCH MẠNG </center></H1>
    <A HREF = "#bh1"> Đất nước</A> <BR>
    <A HREF = "#bh2"> Bài ca không quên </A> <BR>
    <A HREF = "#bh3"> Chào em cô gái Lam Hồng </A> <BR>
    <A HREF = "#bh4"> Đêm trường sơn nhớ Bác </A> <BR>
    <A HREF = "#bh5"> Cô gái mở đường </A> <BR>
    <A HREF = "#bh6"> Chiếc Gậy Trường Sơn </A> <BR>
    <A HREF = "#bh7"> Giải Phóng Miền Nam </A> <BR>
    <A HREF = "#bh8"> Tiền Vè Sài Gòn </A> <BR>
    <A HREF = "#bh9"> Bài Ca May Áo </A> <BR>
    <H2> <A name="bh3">Chào em cô gái Lam Hồng </H2> <BR>
    <P>Xe ta bon trên những dặm đường,<BR>
    Giữa làng quê ta băng qua bao suối đèo đồi nương <BR>
    Mà xe ta bon ra chiến trường <BR>
    Chào em cô gái Lam Hồng <BR>
    Giữa tiếng bom gào đạn dội, vẫn nghe vang vang câu hò trên đường <BR>
    Niềm vui lớn toả lan trên quê ta <BR>
    Đi thông đường để những chuyên xe ta băng băng qua <BR>
    Hồng Lĩnh ơi, đỉnh cao mây ngàn
    // Tương tự với lời các bài hát khác
</BODY>
</HTML>
```

### Kết quả:



Hình 2.24.1: Liên kết đến các phần trong cùng một tài liệu



Hình 2.24.2: Liên kết đến các phần trong cùng một tài liệu

c. 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 2.21:**

&lt;HTML&gt;

```

<HEAD>
    <TITLE> Lien ket</TITLE>
</HEAD>
<BODY>
    <H1><center>DANH MỤC NHẠC CÁCH MẠNG </center></H1>
    <A HREF = "baihat#bh1"> Đất nước</A> <BR>
    <A HREF = "baihat#bh2">Bài ca không quên </A> <BR>
    <A HREF = "baihat#bh3">Chào em cô gái Lam Hồng </A> <BR>
    <A HREF="baihat#bh4">Đêm trường sơn nhớ Bác</A> <BR>
    <A HREF = "baihat#bh5">Cô gái mờ đường </A> <BR>
    <A HREF="baihat#bh6">Chiếc Gậy Trường Sơn </A> <BR>
    <A HREF="baihat#bh7">Giải Phóng Miền Nam </A> <BR>
    <A HREF = "baihat#bh8">Tiến Về Sài Gòn </A> <BR>
    <A HREF = "baihat#bh9">Bài Ca May Áo </A> <BR>
</BODY>
</HTML>

```

**Kết quả:****Hình 2.25.1: Liên kết đến một điểm xác định ở một tài liệu khác**



**Hình 2.25.2: Liên kết đến một điểm xác định ở một tài liệu khác**

d. Sử dụng email

Nếu muốn người sử dụng gửi được email, chúng ta có thể đưa một đặc tính vào trong trang web và cho phép họ gửi email 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](mailto>thisperson@mymail.com)”>

### 2.5.2 Thẻ Meta

Phần tiêu đề cũng có thể 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 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.

### 2.5.3 Các thẻ DIV và SPAN

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 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 2.22:

<HTML>

<HEAD>

<TITLE> DIV va SPAN </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>

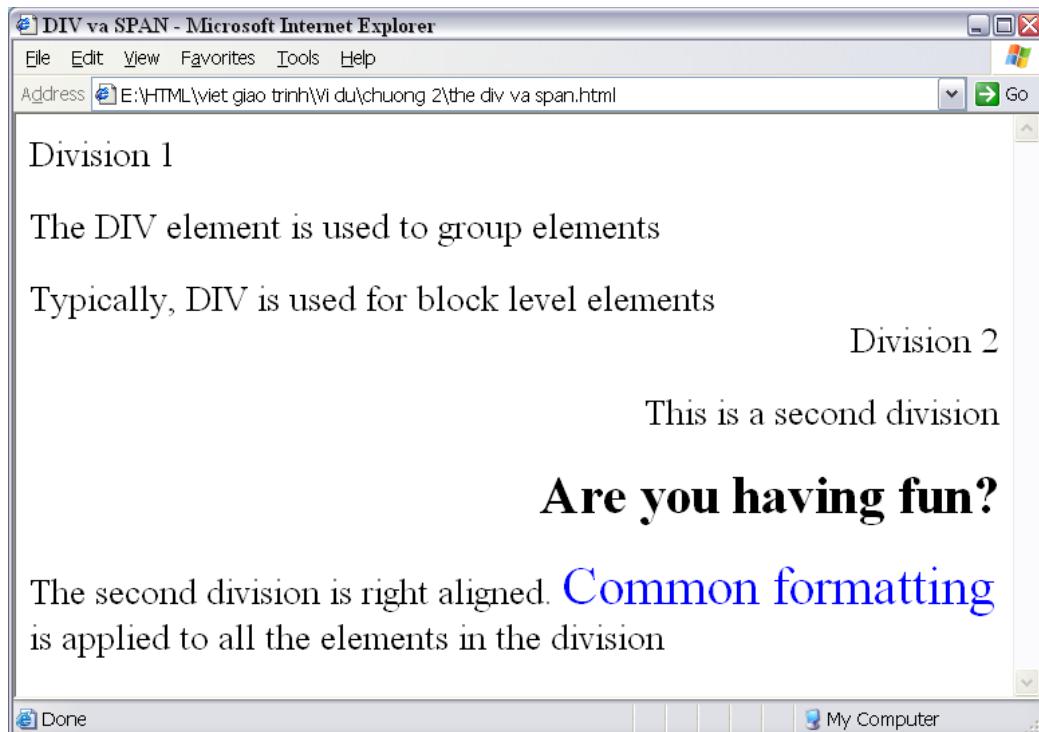
<P> The second division is right aligned.

<SPAN style = “font-size:25; color: blue”> Common formatting  
</SPAN> is applied to all the elements in the division

</BODY>

</HTML>

#### Kết quả:



**Hình 2.26: Các thẻ DIV và SPAN**

#### 2.5.4 Các thẻ mức độ

##### 2.5.4.1 Thẻ <ADDRESS>

Phần tử <ADDRESS> được dùng để hiển thị các thông tin như tác giả, địa chỉ, chữ ký trong tài liệu HTML. Phần tử này được hiển thị ở cuối trang và có thể chứa một hoặc một số phần sau:

- Liên kết đến trang chủ
- Đặc tính chuỗi tìm kiếm
- Thông tin bản quyền

##### Ví dụ 2.23:

```

<HTML>
    <HEAD>
        <TITLE> The ADDRESS </TITLE>
    </HEAD>
    <BODY>
        <H3> My first HTML document </H3>
        <P> This is going to be real fun
        <H2> Using another heading </H2>
        <P> Another paragraph element
        <ADDRESS>
            <P><A HREF = “http://www.viethanit.edu.vn/”> Click here to visit
            Viet-Han’s homepage </A>
        </ADDRESS>
    </BODY>
</HTML>

```

```

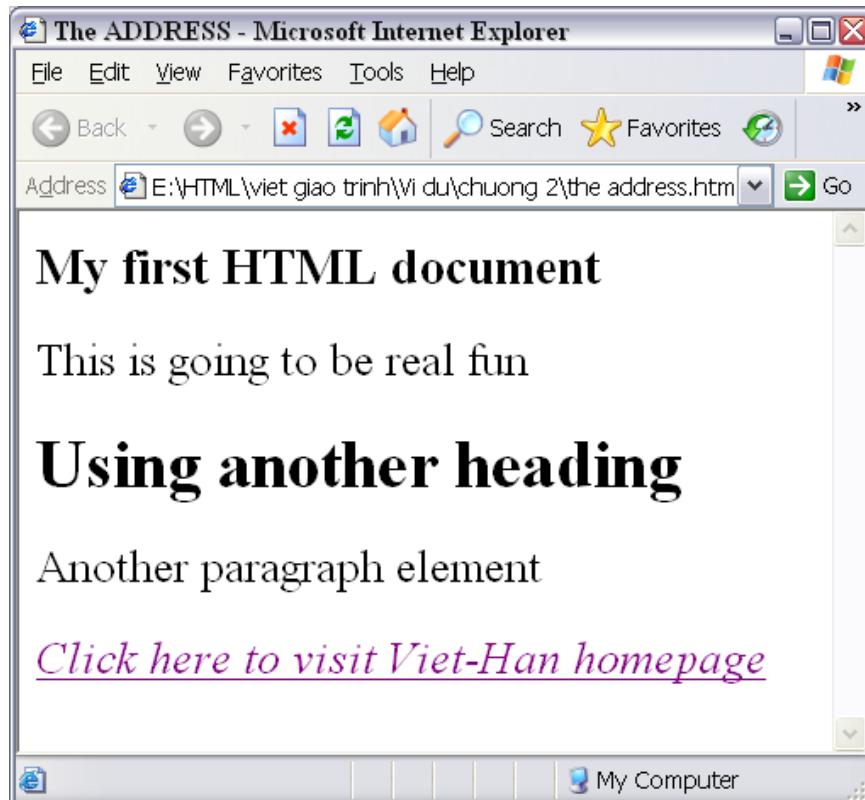
</ADDRESS>

</BODY>

</HTML>

```

**Kết quả:**



**Hình 2.27: Minh họa thẻ ADDRESS**

#### 2.5.4.2 Thẻ <BLOCKQUOTE>

Chúng ta có thể chỉ định một trích dẫn văn bản bên trong tài liệu bằng cách sử dụng phần tử BLOCKQUOTE và Q. BLOCKQUOTE được sử dụng cho những phần trích dẫn dài và được hiển thị như một đoạn văn bản thụt vào đầu dòng. Nếu phần trích dẫn ngắn và không cần ngắt đoạn thì sử dụng phần tử Q tốt hơn. Khi sử dụng phần tử Q, bạn phải xác định dấu ngoặc kép.

**Ví dụ 2.24:**

```

<HTML>

<HEAD>
    <TITLE> The BLOCKQUOTE</TITLE>
</HEAD>

<BODY>
    <P> The blockquote element is used to format the content in blocks of text.
    <BLOCKQUOTE>
        Humpty Dumpty sat on a wall

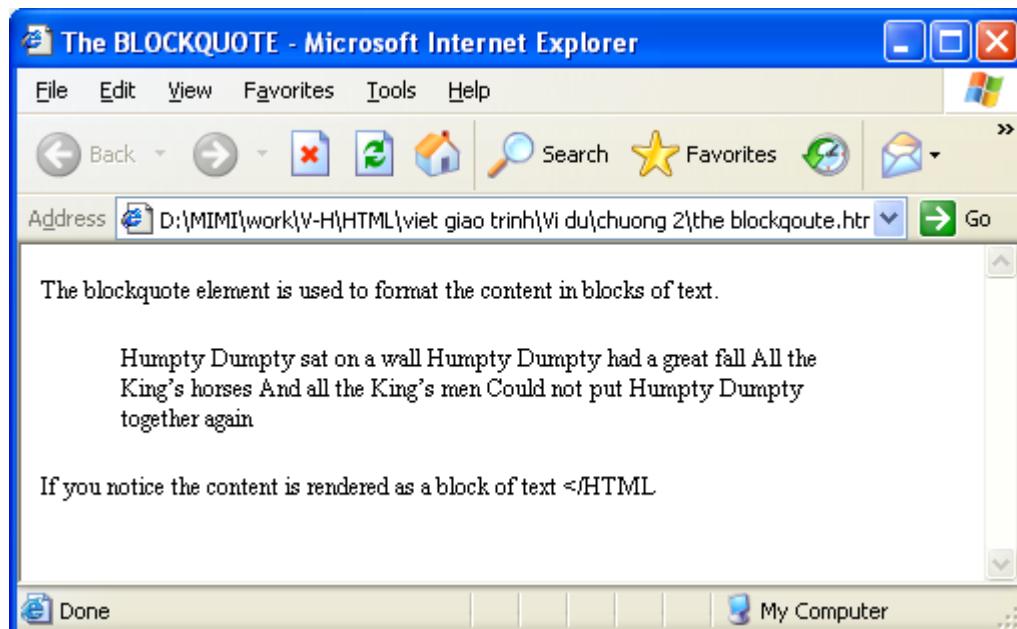
```

```

        Humpty Dumpty had a great fall
        All the King's horses
        And all the King's men
        Could not put Humpty Dumpty together again
    </BLOCKQUOTE>
    <P> If you notice the content is rendered as a block of text
</BODY>
</HTML>

```

**Kết quả:**



**Hình 2.28: Minh họa thẻ BLOCKQUOTE**

#### 2.5.4.3 Thẻ <PRE>

Nếu chúng ta muốn văn bản được hiển thị với định dạng đã được xác định trước, chúng ta sử dụng phần tử PRE. Phần tử này dùng để xác định định dạng cho văn bản. Khi văn bản được hiển thị trong trình duyệt, nó sẽ tuân theo tất cả các định dạng đã được xác định trước trong mã nguồn tài liệu HTML.

**Ví dụ 2.25:**

```

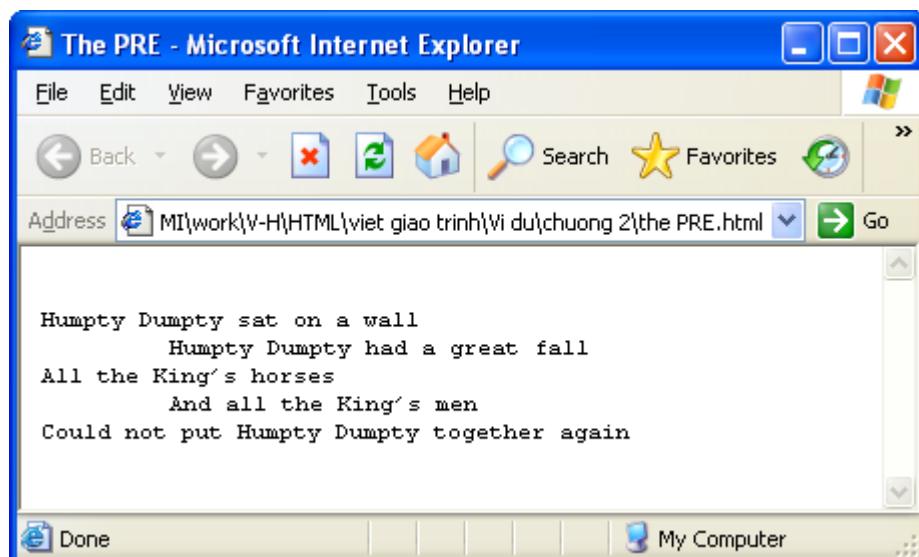
<HTML>
    <HEAD>
        <TITLE> The PRE </TITLE>
    </HEAD>
    <BODY>
        <PRE>
            Humpty Dumpty sat on a wall
            Humpty Dumpty had a great fall

```

```

All the King's horses
And all the King's men
Could not put Humpty Dumpty together again
</PRE>
</BODY>
</HTML>
```

**Kết quả:**



Hình 2.29: Minh họa thẻ PRE

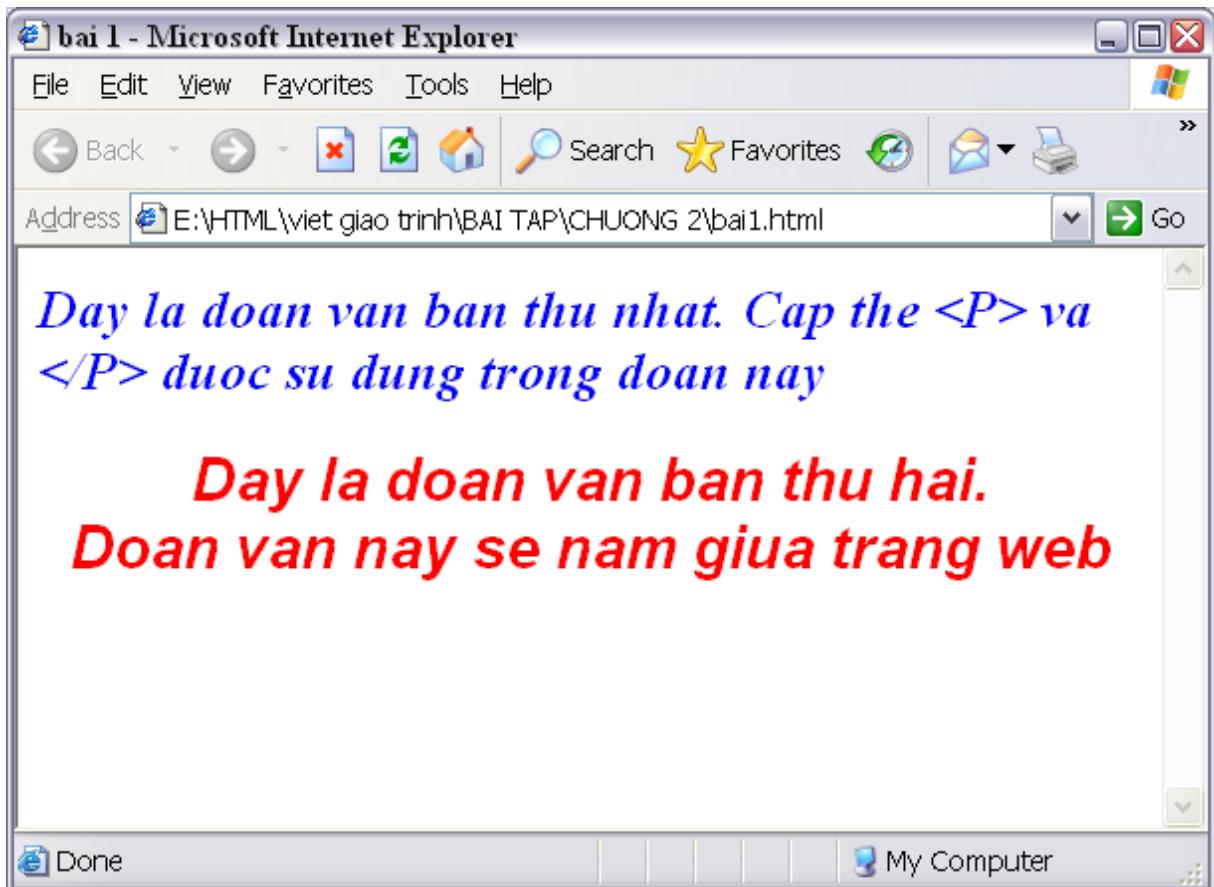
## 2.6 Sử dụng ký tự đặc biệt trong tài liệu HTML

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 (>)	&gt;	
Nhỏ hơn (<)	&lt;	
Trích dẫn ("")	&quot;	
Ký tự "&"	&amp;	

## BÀI TẬP THỰC HÀNH

1. Hãy viết đoạn mã HTML kết hợp thẻ font, thẻ ngắt, thẻ đoạn và các thuộc tính của chúng để có kết quả là trang như sau:



Trong đó: đoạn thứ nhất viết bằng font chữ Times new roman, đoạn thứ hai viết bằng font chữ Arial

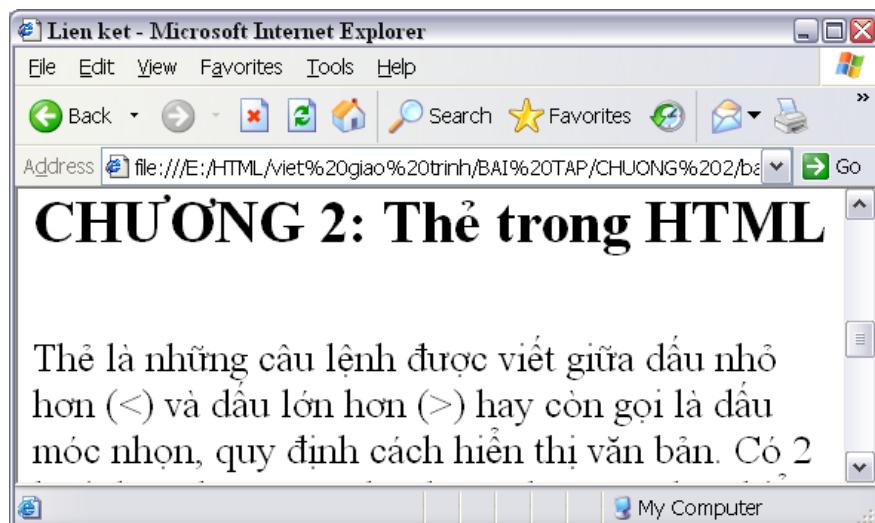
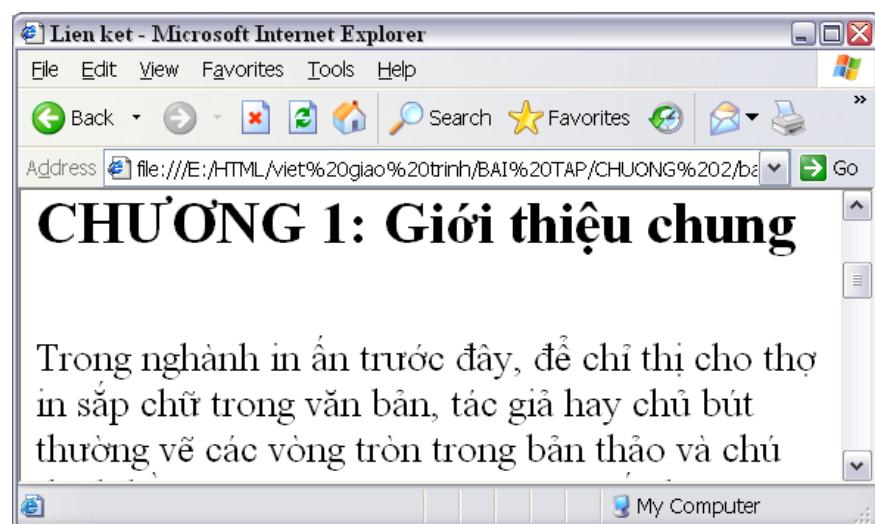
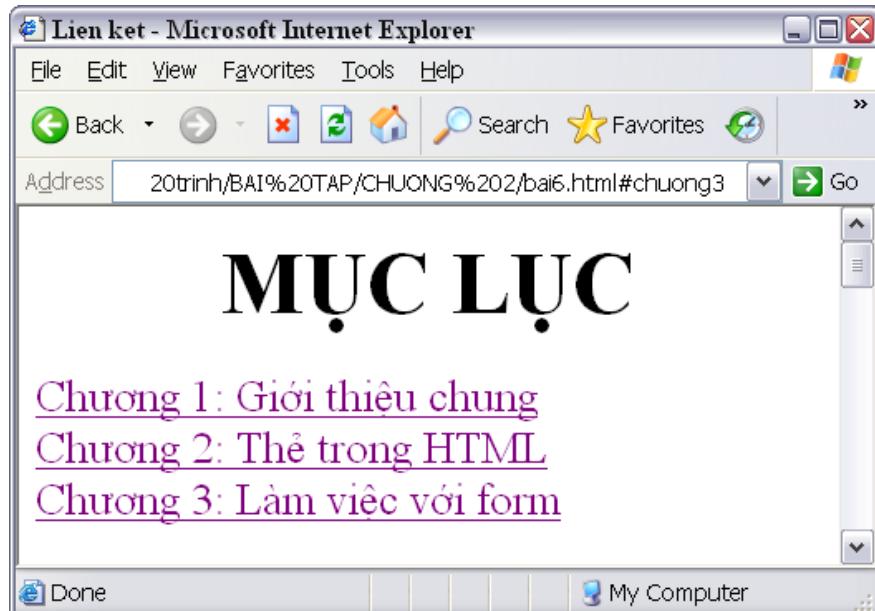
2. Viết đoạn mã HTML để canh lề phải một khối văn bản gồm một tiêu đề và hai đoạn văn bất kỳ bằng cách sử dụng thẻ DIV
3. Viết đoạn mã HTML hiển thị nội dung như sau:

```
* ***** * *****
*      *      *
*      *      *
* ***** * *****
*      *      *
*      *      *
* ***** * *****
```

4. Viết đoạn mã HTML trong đó có chèn một hình ảnh làm nền cho trang web ở bài 3
5. Hãy viết đoạn mã HTML minh họa liên kết giữa hai trang web như hình sau:



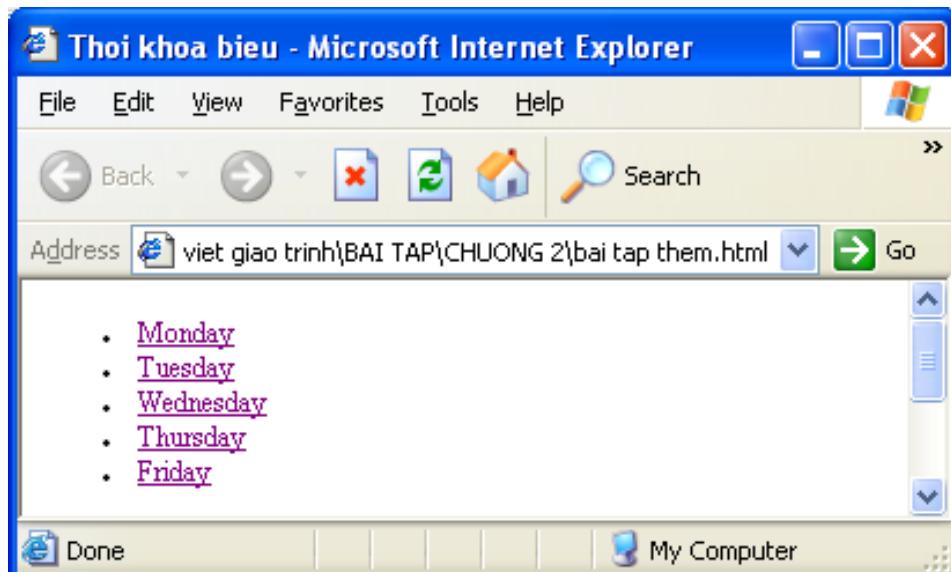
6. Viết đoạn mã HTML minh họa việc liên kết đến một phần trong cùng tài liệu như hình minh họa sau:





Thuộc tính tác động lên nội dung văn bản. Thuộc tính được nhập vào giữa thẻ và dấu lớn hơn cuối cùng. Thường thì chúng ta dùng nhiều thuộc tính

7. Viết đoạn mã HTML tạo ra một thời khoá biểu với danh sách như hình sau:



Danh sách trên là một thời khoá biểu. Trong đó mỗi mục của danh sách là một liên kết đến một vị trí xác định trong một tài liệu khác. Tài liệu này liệt kê các môn học của từng ngày trong tuần. Tuỳ ý trình bày bên trong tài liệu bằng cách vận dụng các thẻ đã học.

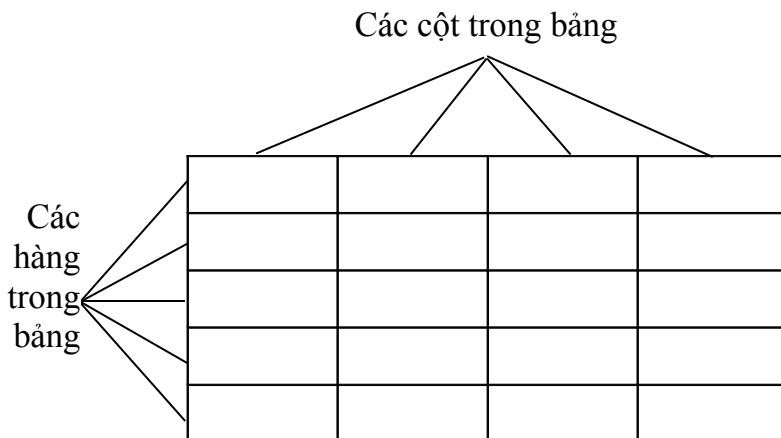
## CHƯƠNG 3

# LÀM VIỆC VỚI BẢNG - BIỂU MẪU – KHUNG VÀ ĐA PHƯƠNG TIỆN

### 3.1 Làm việc với bảng

Bảng là bổ sung quan trọng của HTML, được bắt nguồn từ phòng phát triển ở Nescape Communications Corporation. Tuy nhiên kết quả không được như các chương trình xử lý văn bản phổ biến.

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**

#### 3.1.1 *Cách tạo bảng*

Để tạo bảng chúng ta phải phác họa nội dung cho bảng: bảng có chức năng nhiệm vụ gì? Có bao nhiêu cột? Bao nhiêu dòng? Chiều dài, chiều rộng, nội dung của mỗi ô là gì? Trong bảng, chiều rộng hay chiều dài đều được tính bằng đơn vị pixel.

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 3.1:

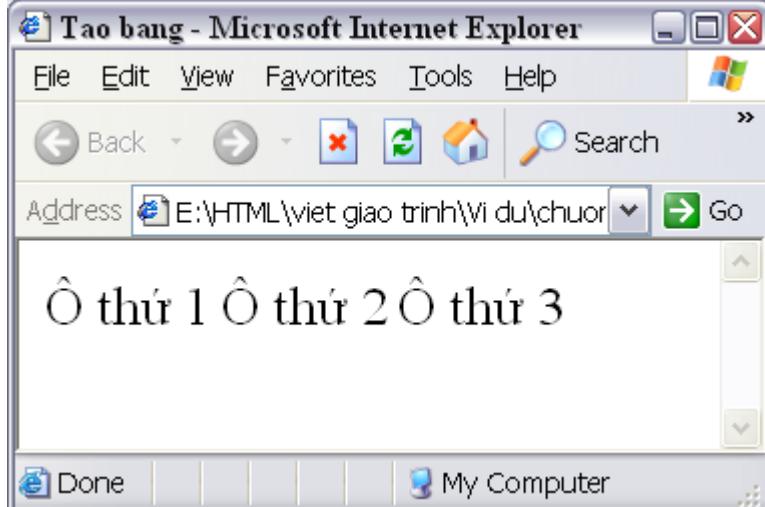
```
<HTML>
  <HEAD>
    <TITLE> Tao bang </TITLE>
  </HEAD>
  <BODY>
    <TABLE>
```

```

<TD> Ô thứ 1 </TD>
<TD> Ô thứ 2 </TD>
<TD> Ô thứ 3 </TD>
</BODY>
</HTML>

```

**Kết quả:**



► **Hình 3.2: Minh họa tạo bảng**

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

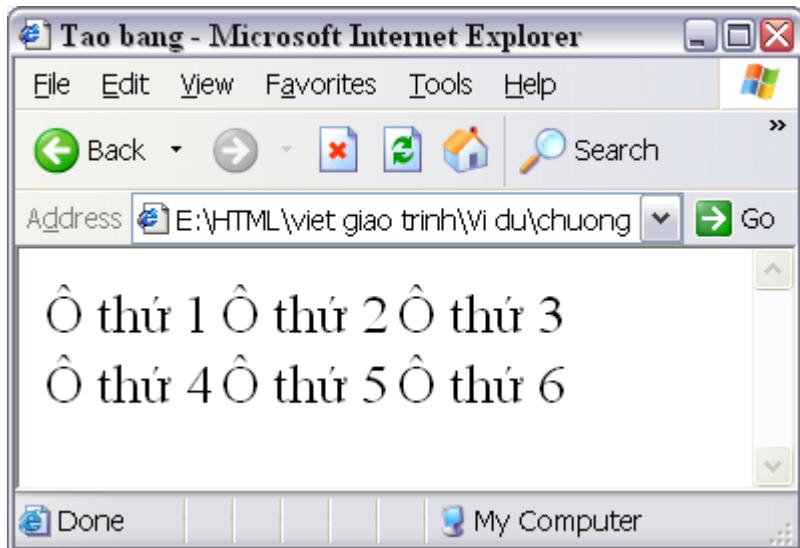
**Ví dụ 3.2:**

```

<HTML>
  <HEAD>
    <TITLE> Tao bang </TITLE>
  </HEAD>
  <BODY>
    <TABLE>
      <TR>
        <TD> Ô thứ 1 </TD>
        <TD> Ô thứ 2 </TD>
        <TD> Ô thứ 3 </TD>
      <TR>
        <TD> Ô thứ 4 </TD>
        <TD> Ô thứ 5 </TD>
        <TD> Ô thứ 6 </TD>
      </TABLE>
    </BODY>

```

&lt;/HTML&gt;

**Kết quả:****Hình 3.3: Minh họa tạo hàng trong bảng**

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.

### 3.1.2 Các thuộc tính của bảng

#### 3.1.2.1 Thuộc tính của thẻ <TABLE>

**Bảng 3.1: Các thuộc tính của thẻ <TABLE>**

Thuộc tính	Mô tả
Bgcolor	Định rõ màu nền cho bảng
Border	Định màu cho đường viền
Bordercolordark	Định màu sẫm cho phần bóng của đường viền
Bordercolorlight	Định màu nhạt cho phần sáng hơn của đường viền.
Cellpadding	Định rõ khoảng cách giữa nội dung và đường viền
Frame	Hiển thị đường viền ngoài
Height	Định rõ chiều cao bảng
Rules	Hiển thị đường viền trong
Width	Định rõ chiều rộng của bảng

#### 3.1.2.2 Thuộc tính của thẻ <TR>

**Bảng 3.2: Các thuộc tính của thẻ <TR>**

Thuộc tính	Mô tả
Align, Vlign	Canh chỉnh nội dung hàng theo phương ngang, phương dọc
Bgcolor	Định rõ màu nền của hàng

## 3.1.2.3 Thuộc tính của thẻ &lt;TD&gt;

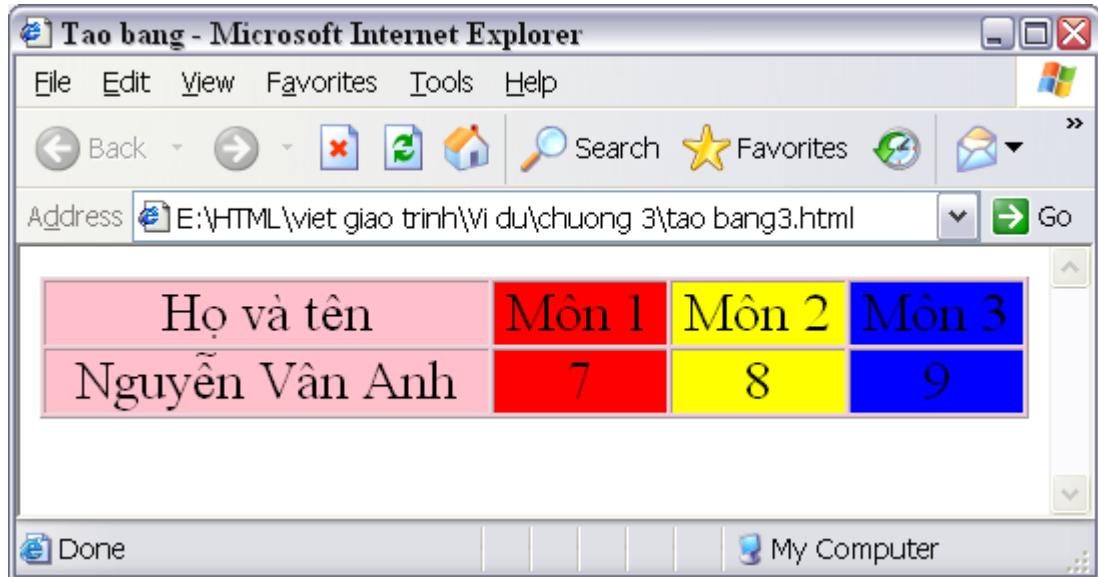
**Bảng 3.3: Các thuộc tính của thẻ <TR>**

Thuộc tính	Mô tả
Align, valign	Canh chỉnh nội dung ô theo phương ngang, phương dọc
Border	Mở rộng ô qua nhiều cột
Colspan	Định màu sẫm cho phần bóng của đường viền
Nowrap	Giữ cho nội dung ô nằm trên một dòng.
Rowspan	Kéo dài ô xuống nhiều hàng
Width, height	Định rõ kích thước ô

**Ví dụ 3.3:**

```
<HTML>
    <HEAD>
        <TITLE> Tao bang </TITLE>
    </HEAD>
    <BODY>
        <TABLE bgcolor = pink width = "100%" border = "1" cellspacing = 1
cellpadding = "0" >
            <TR>
                <TD align = "center" valign = "top"> Họ và tên </TD>
                <TD align = "center" valign = "middle" bgcolor = red> Môn 1
                </TD>
                <TD align = "center" valign = "middle" bgcolor = yellow> Môn
2 </TD>
                <TD align="center" valign="bottom" bgcolor = blue> Môn 3
                </TD>
            </TR>
            <TR>
                <TD align = "center" valign = "top"> Nguyễn Văn Anh </TD>
                <TD align = "center" valign = "middle" bgcolor = red> 7 </TD>
                <TD align="center" valign="middle" bgcolor = yellow> 8
                </TD>
                <TD align="center" valign="middle" bgcolor = blue> 9 </TD>
            </TR>
        </TABLE>
    </BODY>
```

&lt;/HTML&gt;

**Kết quả:****Hình 3.4: Sử dụng các thuộc tính của bảng**

### 3.1.3 **Hiệu chỉnh bảng**

#### 3.1.3.1 *Tạo khung viền cho bảng*

Thuộc tính Border trong thẻ TABLE, với Border =n, trong đó n là độ dày đường viền tính bằng pixel sẽ giúp tạo khung viền cho bảng. HTML mặc định n=2 pixel. Thuộc tính Border ảnh hưởng đến toàn bộ khung viền, kể cả các đường phân chia trong bảng.

Thực chất là khung viền luôn luôn tồn tại, thuộc tính Border chỉ quyết định việc hiển thị hay không hiển thị khung viền. Để xoá thuộc tính khung viền, ta đặt giá trị cho thuộc tính border=0.

Ví dụ:

**<TABLE BORDER=2>**

Ngoài ra chúng ta cũng có thể thay đổi màu cho khung viền. Thông thường, màu của khung viền của một bảng giống với màu nền, nhưng ta có thể thay đổi màu của khung viền để làm ra sự khác biệt đó.

**Cú pháp:**

BORDERCOLOR = "#rrggbb" hoặc BORDERCOLOR= "tên màu"

Ví dụ: Nếu muốn tô màu đỏ cho khung viền với bảng có Border = 2 ta sẽ viết như sau:

**<TABLE BORDER=2 BORDERCOLOR= "Red">**

Bên cạnh đó ta còn có thể thiết đặt một màu nền cho toàn bảng. Màu nền sẽ giúp làm nổi bật bảng, tạo cho người duyệt phải chú ý đến nội dung của bảng. Có thể định màu nền cho cả bảng hoặc chỉ một vài ô trong bảng đều được.

Ta sẽ dùng thuộc tính BGCOLOR cho toàn bảng

Ví dụ: Nếu muốn tô màu xanh dương cho bảng với Border = 2 ta sẽ viết như sau:

```
<TABLE BORDER=2 BGCOLOR= "Green">
```

Chúng ta cũng có thể tô màu nền cho một ô hay một nhóm ô nào đó bằng cách thêm các thuộc tính BGCOLOR vào thẻ của hàng hay ô đó.

### 3.1.3.2 Thay đổi kích thước bảng

Trình duyệt sẽ tự động định dạng chiều rộng bảng bằng cách tính chiều rộng của văn bản chứa bên trong. Tuy nhiên người thiết kế vẫn có thể định dạng kích cỡ để ước lượng được các khoảng trống.

**Cú pháp:**

```
<TABLE WIDTH= "x" HEIGHT= "y">
```

Tương tự như trên ta có thể định dạng kích thước ô. Việc thay đổi kích thước của ô dẫn đến thay đổi kích thước của cả các ô khác trong hàng.

**Cú pháp:**

```
<TH WIDTH= "x" HEIGHT= "y">
```

```
<TD WIDTH= "x" HEIGHT= "y">
```

Ngoài ra ta còn có thể canh bảng trên trang bằng thuộc tính quen thuộc ALIGN

**Cú pháp:**

```
<TABLE ALIGN= ">
```

### 3.1.3.3 Bổ sung cạnh và đường kẻ luar

Có hai đặc tính mới của bảng trong đặc tả kỹ thuật HTML. Cả hai đều cung cấp kiểm soát chi tiết về đường biên xung quanh bảng và giữa những ô dữ liệu riêng lẻ. Đó là frame và rules.

**Cú pháp:**

```
<TABLE FRAME= "VALUE" RULES= "VALUE" WIDTH= "50%">
```

Chẳng hạn:

```
<table align = "center" border = "10" frame = "vsible" cellspacing= "0" rules= "rows" width= "50%">
```

Giá trị của chúng sẽ được định nghĩa trong các bảng sau.

**Bảng 3.4: Bảng giá trị của thuộc tính frame**

Giá trị	Ý nghĩa
Void	Loại bỏ toàn bộ đường biên bảng bên ngoài
Above	Hiển thị đường biên ở phía trên khung bảng
Below	Hiển thị đường biên phía dưới khung bảng
Hsides	Hiển thị đường biên trên và dưới khung bảng
Lsh	Hiển thị đường biên phía trái khung bảng
Rsh	Hiển thị đường biên phía phải khung bảng

Vside	Hiển thị đường biên phía trái và phải khung bảng
Box	Hiển thị đường biên tất cả các phía của khung bảng

**Bảng 3.5: Bảng giá trị của thuộc tính Rows**

Giá trị	Ý nghĩa
None	Loại bỏ toàn bộ đường biên bảng bên trong
Group	Hiển thị đường biên ngang giữa tất cả các nhóm bảng. Nhóm được xác định bằng những phần tử <code>thead</code> , <code>tbody</code> , <code>tfoot</code> và <code>colgroup</code> .
Rows	Hiển thị đường biên chiều ngang giữa tất cả các hàng của bảng.
Cols	Hiển thị đường biên đứng giữa tất cả các cột bảng

### 3.1.3.4 Trang trí văn bản chung quanh bảng

Kỹ thuật chèn văn bản chung quanh bảng cũng tương tự như chèn văn bản chung quanh hình ảnh. Văn bản được xếp chung quanh bảng được nhập vào sau bảng.

Nếu bảng được canh theo lề trái thì văn bản nằm phía bên phải, còn văn bản được căn theo lề phải thì bảng nằm bên trái.

#### **Cú pháp:**

```
<TABLE ALIGN = left> </TABLE>
<TABLE ALIGN = right> </TABLE>
```

#### **Ví dụ 3.4 :**

```
<HTML>
```

```
<HEAD>
    <TITLE> Hieu chinh bang </TITLE>
</HEAD>
<BODY>
    <TABLE bordercolor = red width = "50%" border = "1" cellspacing = 1
cellpadding = "0" align = left >
        <TR>
            <TD align = "center" valign = "top"> Họ và tên </TD>
            <TD align = "center" valign = "middle" bgcolor = red> Môn 1
</TD>
            <TD align = "center" valign = "middle" bgcolor = yellow> Môn
2 </TD>
            <TD align="center" valign="bottom" bgcolor = blue> Môn 3
</TD>
        </TR>
    </TABLE>
</BODY>
</HTML>
```

```

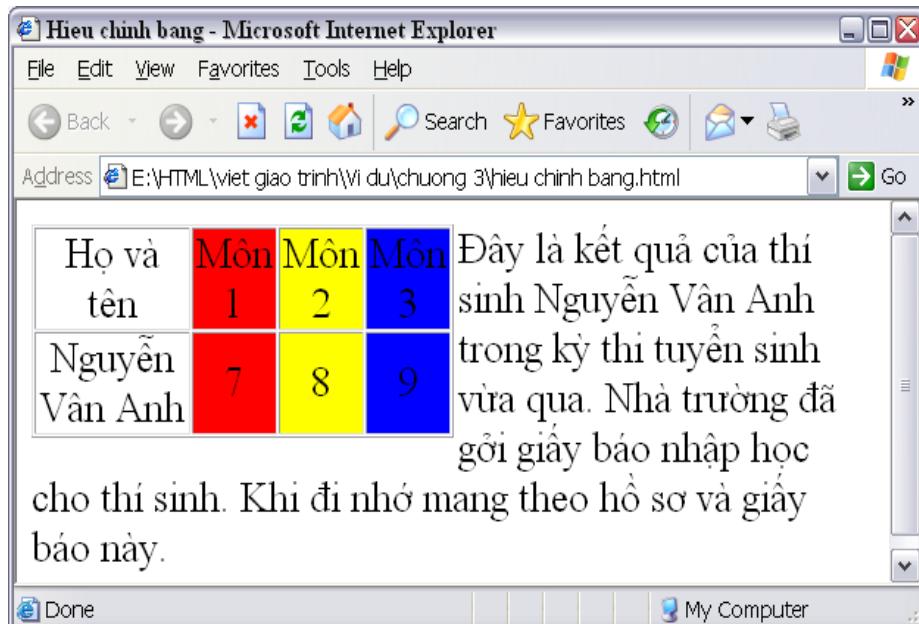
<TR>
    <TD align = "center" valign = "top"> Nguyễn Văn Anh </TD>
    <TD align = "center" valign = "middle" bgcolor = red> 7 </TD>
    <TD align="center" valign="middle" bgcolor = yellow> 8
    </TD>
    <TD align="center" valign="middle" bgcolor = blue> 9 </TD>
</TR>
</TABLE>

Đây là kết quả của thí sinh Nguyễn Văn Anh trong kỳ thi tuyển sinh vừa qua. Nhà trường đã gửi giấy báo nhập học cho thí sinh. Khi đi nhớ mang theo hồ sơ và giấy báo này.

</BODY>
</HTML>

```

### Kết quả:



► **Hình 3.5: Trang trí văn bản xung quanh bảng**

#### 3.1.3.5 Kết hợp các cột và các 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>.

#### Cú pháp:

<TD COLSPAN = n </TD> với n là số cột mà ô đó trải qua

<TD ROWSPAN = n </TD> với n là số hàng mà ô đó trải qua.

**Ví dụ 3.5:**

&lt;HTML&gt;

```

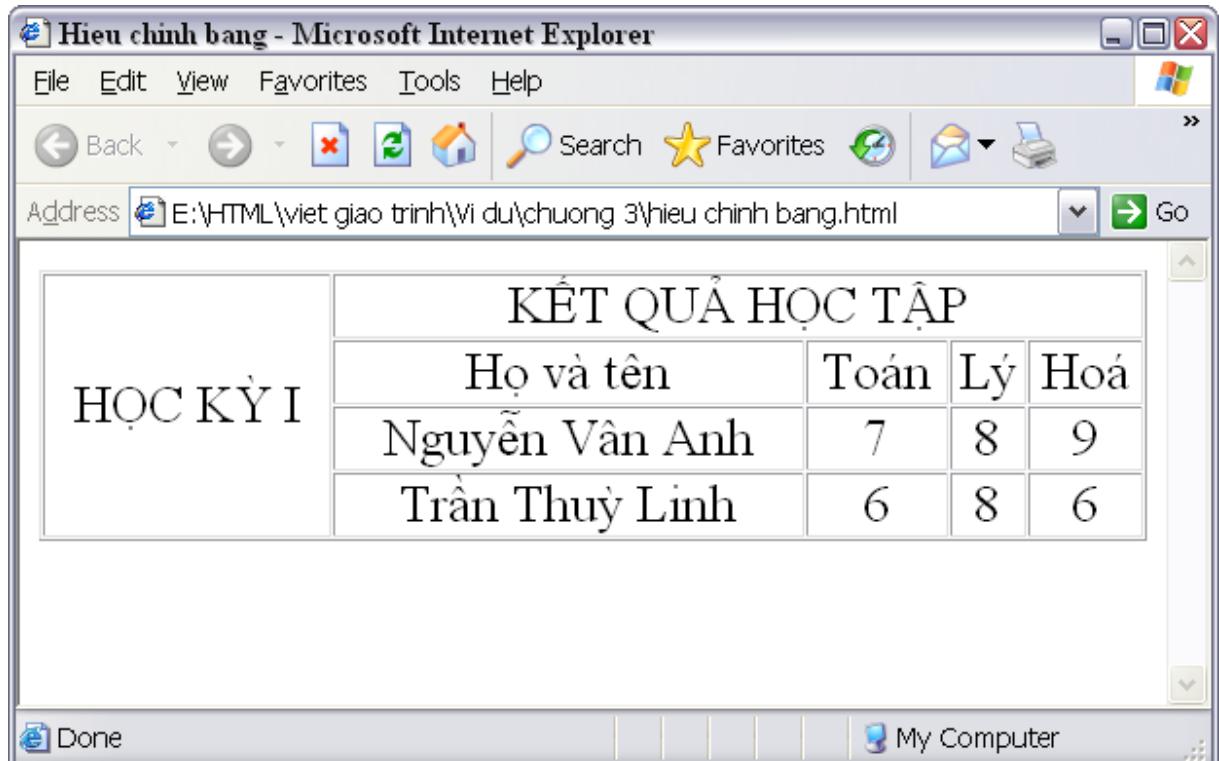
<HEAD>
    <TITLE> Hieu chinh bang </TITLE>
</HEAD>
<BODY>

    <TABLE bordercolor = red width = "100%" border = "1" cellspacing = 1
cellpadding = "0" align = left >

        <TR>
            <TD rowspan = 4 align = center> HỌC KỲ I </TD>
            <TD colspan = 4 align = center> KẾT QUẢ HỌC TẬP </TD>
        </TR>
        <TR>
            <TD align = "center" valign = "top"> Họ và tên </TD>
            <TD align = "center" valign = "middle"> Toán </TD>
            <TD align = "center" valign = "middle"> Lý </TD>
            <TD align="center" valign=""middle"> Hoá </TD>
        </TR>
        <TR>
            <TD align = "center" valign = "top"> Nguyễn Văn Anh </TD>
            <TD align = "center" valign = "middle"> 7 </TD>
            <TD align="center" valign="middle"> 8 </TD>
            <TD align="center" valign="middle"> 9 </TD>
        </TR>
        <TR>
            <TD align = "center" valign = "top"> Trần Thuỷ Linh </TD>
            <TD align = "center" valign = "middle"> 6 </TD>
            <TD align="center" valign="middle"> 8 </TD>
            <TD align="center" valign="middle"> 6 </TD>
        </TR>
    </TABLE>
</BODY>
</HTML>

```

**Kết quả:**



► **Hình 3.6: Kết hợp các cột và các dòng**

### 3.1.3.6 *Canh lề nội dung trong ô*

Thường nội dung bên trong ô được mặc định canh lề bên trái và ở giữa ô (theo chiều dọc), nhưng ta cũng có thể canh lề theo nội dung bên trong ô theo ý mình.

Trong thẻ của mỗi ô, mỗi hàng hay mỗi section gõ :

ALIGN=direction, VALIGN=direction

Đối với **ALIGN** thì **direction** nhận các giá trị là **left, center, right**

Đối với **VALIGN** thì **direction** nhận các giá trị là **top, middle, bottom** .

Chúng ta có thể canh lề cho tất cả các ô trong một hoặc nhiều cột hay hàng bằng cách thêm thuộc tính **ALIGN** hay **VALIGN** vào các thẻ như TR, THEAD, TBODY... tạo khoảng cách bên trong và xung quanh ô.

Khoảng cách giữa các ô làm cho bảng to ra mà vẫn không thay đổi gì kích thước của ô cả. Khoảng trống xung quanh nội dung trong ô sẽ đẩy đường viền ra ngoài.

Trong thẻ TABLE gõ

CELLSPACING = n (đối với khoảng cách bên ngoài nó)

CELLPADDING = n (đối với khoảng cách bên trong nó)

với **n** khoảng cách tính bằng Pixel.

### 3.1.3.7 *Sử dụng hình ảnh làm nền cho bảng*

Để sử dụng hình ảnh làm nền bạn cần chú ý độ tương phản của hình nền và nội dung bảng.

**Cú pháp:**

```
<TABLE Background = "image.gif">
```

```
<TD Background = "image.gif">
```

**Ví dụ 3.6 :**

```
<HTML>
```

```
    <HEAD>
```

```
        <TITLE> Hieu chinh bang </TITLE>
```

```
    </HEAD>
```

```
    <BODY>
```

```
        <TABLE bordercolor = red width = "100%" border = "1" cellspacing = 1  
cellpadding = "0" background = "c711.jpg" >
```

```
            <TR>
```

```
                <TD rowspan = 4 align = center> HỌC KỲ I </TD>
```

```
                <TD colspan = 4 align = center> KẾT QUẢ HỌC TẬP </TD>
```

```
            </TR>
```

```
            <TR>
```

```
                <TD align = "center" valign = "top"> Họ và tên </TD>
```

```
                <TD align = "center" valign = "middle"> Toán </TD>
```

```
                <TD align = "center" valign = "middle"> Lý </TD>
```

```
                <TD align="center" valign="middle"> Hoá </TD>
```

```
            </TR>
```

```
            <TR>
```

```
                <TD align = "center" valign = "top"> Nguyễn Văn Anh </TD>
```

```
                <TD align = "center" valign = "middle"> 7 </TD>
```

```
                <TD align="center" valign="middle"> 8 </TD>
```

```
                <TD align="center" valign="middle"> 9 </TD>
```

```
            </TR>
```

```
            <TR>
```

```
                <TD align = "center" valign = "top"> Trần Thuỳ Linh </TD>
```

```
                <TD align = "center" valign = "middle"> 6 </TD>
```

```
                <TD align="center" valign="middle"> 8 </TD>
```

```
                <TD align="center" valign="middle"> 6 </TD>
```

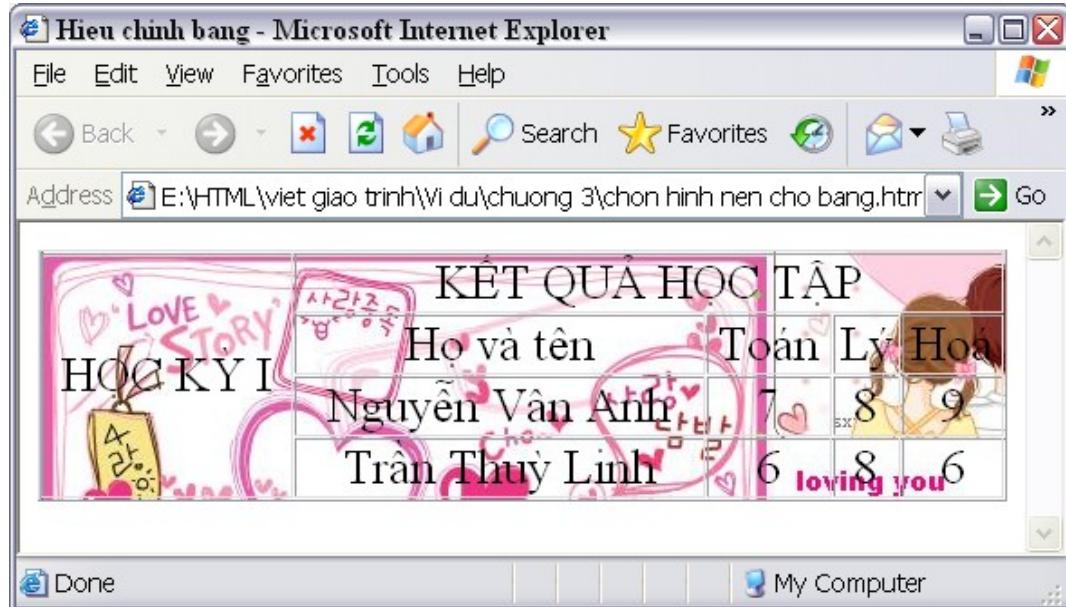
```
            </TR>
```

```
        </TABLE>
```

```
    </BODY>
```

```
</HTML>
```

**Kết quả:**



► **Hình 3.7: Sử dụng hình ảnh làm nền cho bảng**

### 3.2 Làm việc với 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.

#### 3.2.1 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 email 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 website. Hầu hết các site cung cấp một dịch vụ nào đó 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 website. 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:

#### 3.2.2 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 của thẻ này bao gồm:

**Bảng 3.6: Các thuộc tính của thẻ <FORM>**

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: 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: 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: METHOD = (GET   POST)

Ví dụ: Để đưa một biểu mẫu đến chương trình “xử lý biểu mẫu” sử dụng theo phương thức POST ta viết như sau:

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

### 3.2.3 Các phần tử nhập của HTML

Khi tạo ra 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.

#### 3.2.3.1 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:

**Bảng 3.7: Các thuộc tính của thẻ <INPUT>**

Thuộc tính	Mô tả
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 thuộc tính tuỳ 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 gt.
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.

### 3.2.3.2 Button

Phần tử này tạo ra một điều khiển nút (button)

**Bảng 3.7: Các thuộc tính của đối tượng INPUT cho phần tử Button**

Thuộc tính	Mô tả
NAME	Thiết lập hoặc truy xuất tên của điều khiển
SIZE	Thiết lập hoặc truy xuất kích thước của điều khiển
TYPE	Được biểu diễn bởi <INPUT type = button>
VALUE	Thiết lập hoặc truy xuất giá trị của nút

### 3.2.3.3 Textbox

Phần tử này tạo ra một điều khiển nhập văn bản trên 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.

Chẳng hạn:

<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.

### 3.2.3.4 Checkbox

Phần tử 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.

**Bảng 3.7: Các thuộc tính của đối tượng INPUT cho phần tử Checkbox**

Thuộc tính	Mô tả
NAME	Thiết lập hoặc truy xuất tên của điều khiển
SIZE	Thiết lập hoặc truy xuất kích thước của điều khiển
TYPE	Được biểu diễn bởi <INPUT type = checkbox>
VALUE	Thiết lập hoặc truy xuất giá trị của checkbox
STATUS	Thiết lập hoặc truy xuất trạng thái xem checkbox có được chọn hay không.
CHECKED	Thiết lập hoặc truy xuất trạng thái của checkbox

### 3.2.3.5 Radio

Phần tử này tạo ra điều khiển 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 nên cặp NAME/VALUE trong dữ liệu được nhận. Các nút radio nên đặt thuộc tính VALUE.

**Bảng 3.8: Các thuộc tính của đối tượng INPUT cho phần tử Radio**

Thuộc tính	Mô tả
NAME	Thiết lập hoặc truy xuất tên của điều khiển
SIZE	Thiết lập hoặc truy xuất kích thước của điều khiển
TYPE	Được biểu diễn bởi <INPUT type = radio>
VALUE	Thiết lập hoặc truy xuất giá trị của radio
STATUS	Thiết lập hoặc truy xuất trạng thái xem nút radio có được chọn hay không.
CHECKED	Thiết lập hoặc truy xuất trạng thái của nút radio.

Chẳng hạn:

<INPUT TYPE = radio NAME = “sex” VALUE = “male”> Male

### 3.2.3.6 Submit

Phần tử này tạo ra một nút Submit. Khi người dùng nhấp vào nút này, 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.

Chẳng hạn:

<INPUT TYPE = submit NAME = “b1” VALUE = “Click”>

### 3.2.3.7 *Ảnh*

Phần tử 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. Toạ 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.

Chẳng hạn:

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

### 3.2.3.8 *Reset*

Phần tử 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.

Chẳng hạn:

```
<INPUT TYPE = reset VALUE = “Reset” NAME = “B2”>
```

**Ví dụ 3.7:** Chương trình sau thể hiện việc sử dụng nhiều kiểu nhập khác nhau

<HTML>

<HEAD>

<TITLE> Sample form </TITLE>

</HEAD>

<BODY>

<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 = “Radio\_example” VALUE = “Radio-0”> Beginner

<INPUT TYPE = radio NAME = “Radio\_example” VALUE = “Radio-1”> Intermediate

<INPUT TYPE = radio NAME = “Radio\_example” VALUE = “Radio-2”> Expert

</P>

<P><B> Types of Investments you make </B></P>

<P>

```

<INPUT TYPE = checkbox NAME = "Checkbox_example"
       VALUE = "Checkbox-0"> Individual Stocks

<INPUT TYPE = checkbox NAME = "Checkbox_example"
       VALUE = "Checkbox-1"> Options

<INPUT TYPE = checkbox NAME = "Checkbox_example"
       VALUE = "Checkbox-2"> Mutual Funds <BR> </P>

<P><B> What is your stock pick for this year? </B></P>

<P><INPUT TYPE = text name = "Textfield" size = "30"
        MAXLENGTH = "30"></P>

<P>

    <INPUT TYPE = submit NAME = "Submit" VALUE =
           "Submit">

    <INPUT TYPE = reset NAME = "Reset" VALUE = "Reset">
</P>

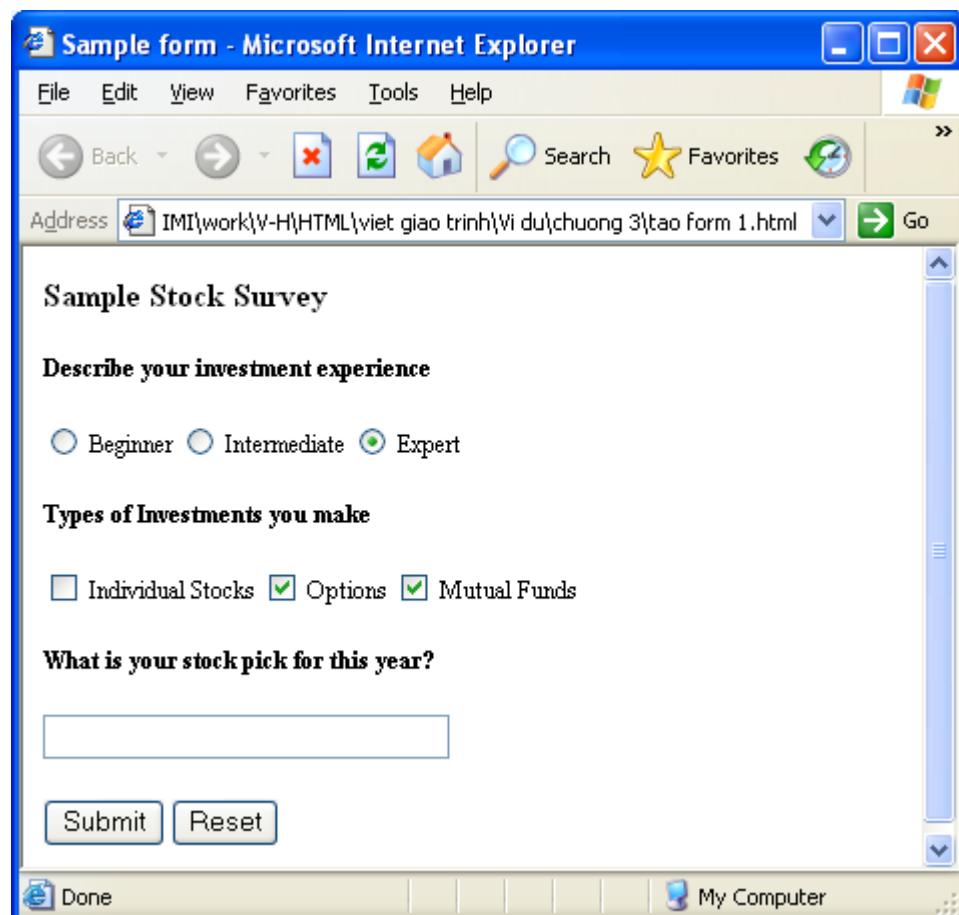
</FORM>

</BODY>

</HTML>

```

### Kết quả:



► **Hình 3.8: Minh họa một số phần tử trong biểu mẫu**

#### 3.2.3.9 Phản hồi TextArea

Phần tử 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>

**Bảng 3.9: Các thuộc tính của phần tử TextArea**

Thuộc tính	Mô tả
COLS	Truy xuất độ rộng của TextArea
ROWS	Thiết lập hoặc truy xuất số hàng ngang trong <TEXTAREA>
SIZE	Thiết lập hoặc truy xuất kích thước của điều khiển
TYPE	Truy xuất loại điều khiển, sử dụng giá trị <TEXTAREA>
VALUE	Thiết lập hoặc truy xuất giá trị của TEXTAREA

Chẳng hạn:

<TEXTAREA NAME = “Text1” COLS = 20 ROWS = 5></TEXTAREA>

### 3.2.3.10 Phân tử BUTTON

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

**Bảng 3.10: Các thuộc tính của phân tử Button**

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 nhập 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.

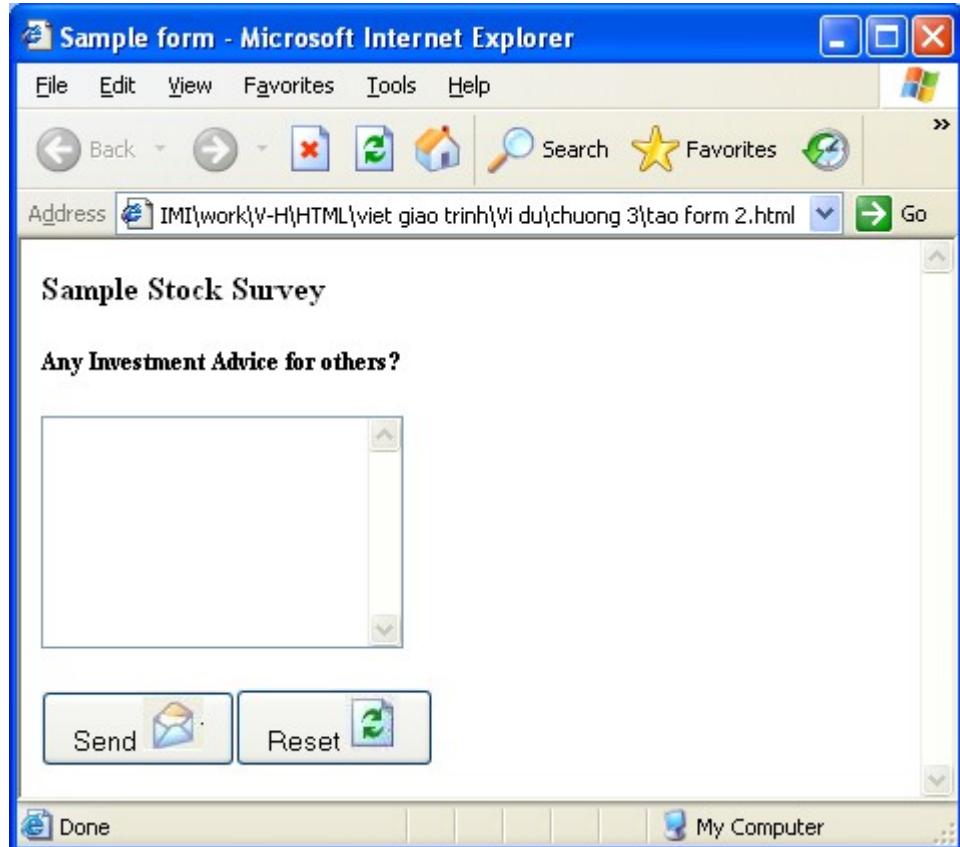
Một nút (BUTTON) có hai loại là submit (type = submit) giống như một phân tử INPUT của loại nút. Sự khác nhau ở chỗ khi phân tử BUTTON được nhập vào thì cặp tên/giá trị được nhận cùng biểu mẫu. Một nút có loại là submit cũng chứa một ảnh và giống một phân tử INPUT có loại là ảnh. Sự khác nhau ở chỗ phân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

**Ví dụ 3.8:** Chương trình sau minh họa cho việc sử dụng phân tử TEXTAREA và BUTTON

<HTML>

```
<HEAD>
    <TITLE> Sample form </TITLE>
</HEAD>
<BODY>
    <FORM ACTION = “http://www.mysite.com/FormSite” METHOD =
POST>
        <B><CENTER><H2 align = left> Sample Stock Survey
        </H2></CENTER></B>
        <P><B> Any Investment Advice for others? </B></P>
        <TEXTAREA NAME = “TextArea” ROWS = 7 COLS = 20>
        </TEXTAREA>
        <P>
            <BUTTON TYPE = “submit” NAME = “Submit” VALUE =
“Submit”> Send <IMG src = “send.jpg”> </BUTTON>
            <BUTTON TYPE = “reset” NAME = “Reset” VALUE = “Reset”>
            Reset <IMG src=“reset.jpg”> </BUTTON>
        </P>
    </FORM>
</BODY>
</HTML>
```

**Kết quả:**



► **Hình 3.9: Minh họa phần tử TEXTAREA và BUTTON**

#### 3.2.3.11 Phân tử Select

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

**Bảng 3.11: Các thuộc tính của phân tử SELECT**

Thuộc tính	Mô tả
NAME	Gán tên cho phân 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 chức năng 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 văn bản mô tả của nó, xem nó có được chọn hay không.

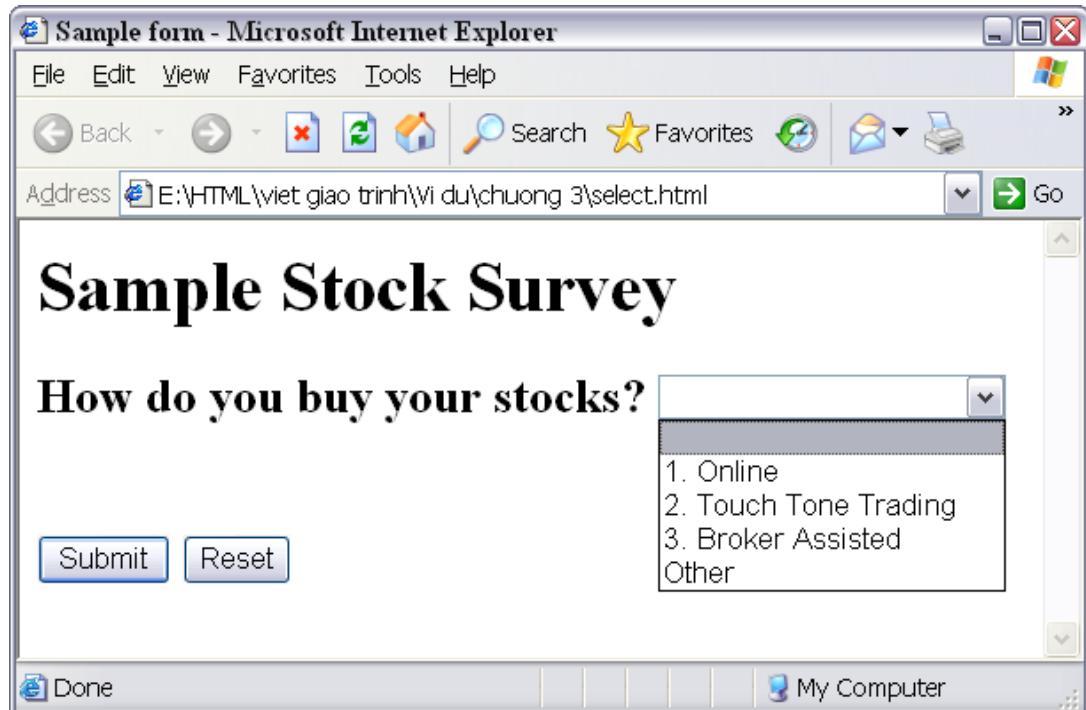
Mảng OPTION được tạo ra theo danh sách lựa chọn trong phân tử SELECT. Mỗi lựa chọn có thuộc tính Text và Selected cho phép chúng ta kiểm tra tùy chọn đó có được chọn hay không và truy xuất văn bản của lựa chọn theo thứ tự. Bây giờ ta có thể kiểm tra mỗi phân tử trong mảng và xác nhận nó.

**Ví dụ 3.9:**

&lt;HTML&gt;

```
<HEAD>
    <TITLE> Sample form </TITLE>
</HEAD>
<BODY>
    <FORM ACTION = “http://www.mysite.com/FormSite” METHOD =
POST>
        <B><CENTER><H2 align = left> Sample Stock Survey
        </H2></CENTER></B>
        <P><B> How do you buy your stocks? </B>
        <SELECT NAME = “Select example”>
            <OPTION></OPTION>
            <OPTION>1. Online</OPTION>
            <OPTION>2. Touch Tone Trading</OPTION>
            <OPTION>3. Broker Assisted</OPTION>
            <OPTION>Other</OPTION>
        </SELECT>
        <BR><BR><BR>
        <INPUT TYPE = “submit” NAME = “Submit” VALUE = “Submit”>
        <INPUT TYPE = “reset” NAME = “Reset” VALUE = “Reset”>
        </P>
    </FORM>
</BODY>
</HTML>
```

**Kết quả:**



► **Hình 3.10: Minh họa phần tử SELECT**

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.

**Bảng 3.12: Các thuộc tính của phần tử OPTGROUP**

Thuộc tính	Mô tả
SELECTED	Đây là thuộc tính logic sử dụng để chọn trước một tuỳ chọn.
VALUE	Xác định giá trị được nhận vào cho tuỳ 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 tuỳ chọn.

#### Ví dụ 3.10:

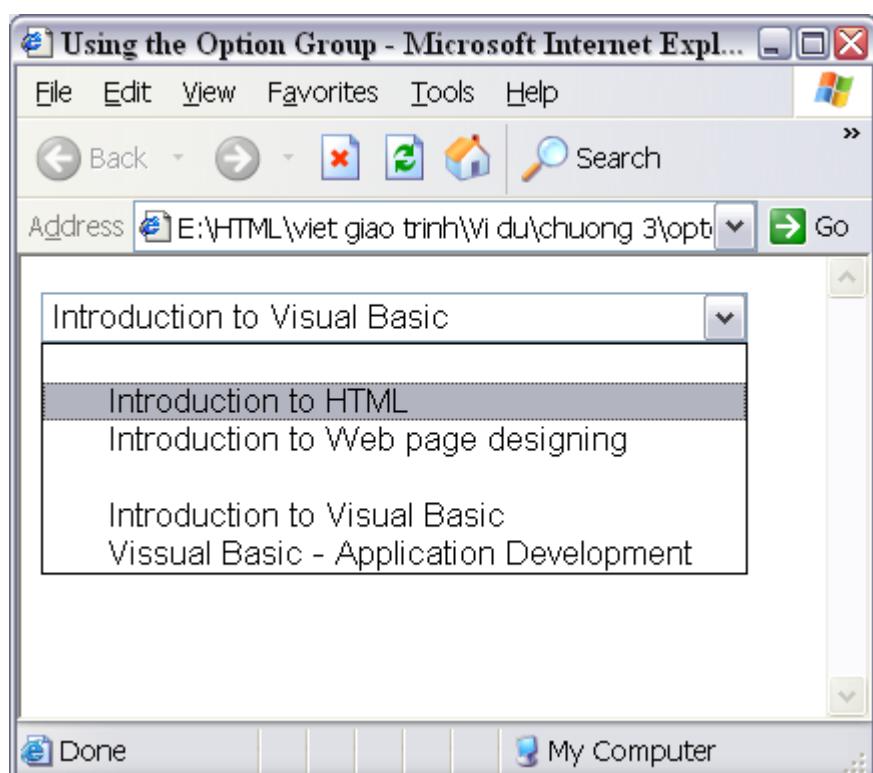
```
<HTML>
  <HEAD>
    <TITLE> Using the Option Group </TITLE>
  </HEAD>
  <BODY>
    <FORM ACTION = "http://www.mysite.com/FormSite" METHOD =
    POST>
      <P> <SELECT name = "course">
        <OPTGROUP>
          <OPTION value="InternetIntro"> Introduction to HTML
```

```

<OPTION value="IntroWeb"> Introduction to Web page
designing
</OPTGROUP>
<OPTGROUP>
<OPTION value="VBIntro"> Introduction to Visual Basic
<OPTION value="VBDev"> Vissual Basic – Application
Development
</OPTGROUP>
</SELECT>
</FORM>
</BODY>
</HTML>

```

### Kết quả:



► **Hình 3.11: Minh họa phần tử OPTGROUP**

#### 3.2.3.12 Phản tử LABEL

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

### Ví dụ 3.11:

```

<HTML>
<HEAD>

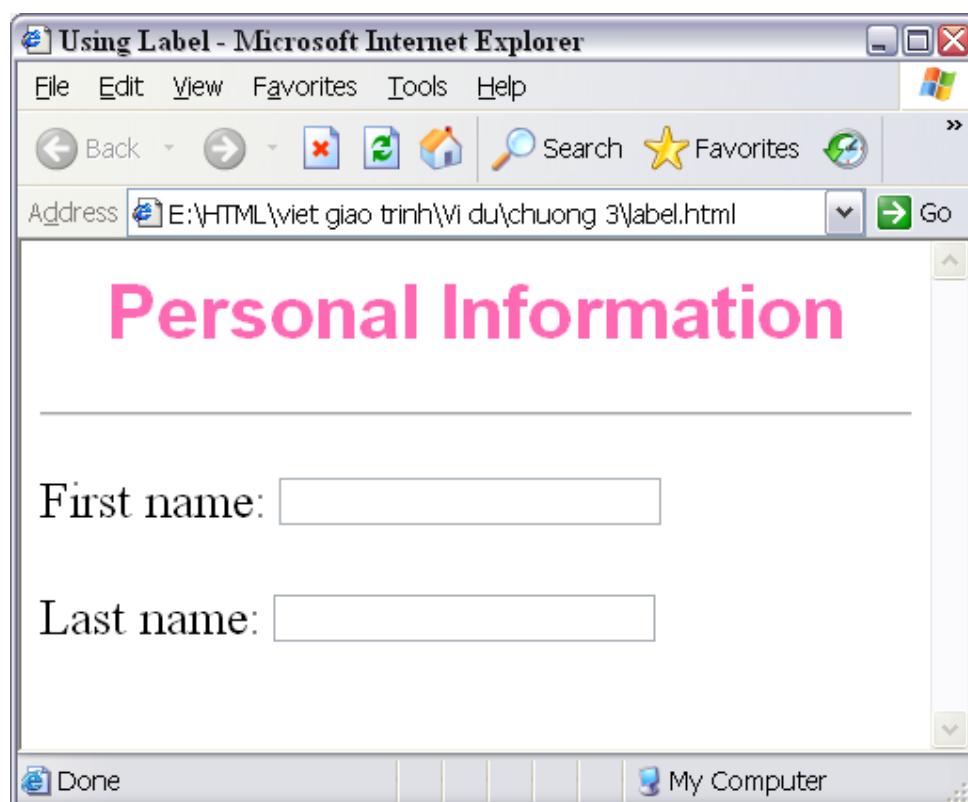
```

```

<TITLE> Using Label </TITLE>
</HEAD>
<BODY>
<H2><CENTER><FONT size = 5 color = hotpink face = arial> Personal
Information </FONT></CENTER></H2>
<HR align = center>
<FORM ACTION = "http://www.mysite.com/FormSite" 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>

```

**Kết quả:**



► **Hình 3.12: Minh họa phần tử LABEL**

#### 3.2.4 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 3.12:**

&lt;HTML&gt;

```
<HEAD>
    <TITLE> Job application</TITLE>
</HEAD>
<BODY>
    <H1><CENTER><FONT size = 4 color = hotpink> Application Form
    </FONT></CENTER></H1>
    <HR><BR>
    <FORM ACTION = "http://www.mysite.com/FormSite" METHOD =
    POST>
        <P>
            <LABEL for = "name"> Name: </LABEL>
            <INPUT type = "text" id = "name"> <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> 2 years </OPTION>
            <OPTION> 3 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>

```

**Kết quả:**

The screenshot shows a Microsoft Internet Explorer window with the title "Job application - Microsoft Internet Explorer". The address bar displays the path "E:\HTML\viet giao trinh\vi du\chuong 3\tonghop.html". The main content area is titled "Application Form". It contains the following elements:

- A text input field labeled "Name:".
- A section labeled "Area of Interest" with three radio buttons: "Web Designer" (selected), "Web Administrator", and "Web Developer".
- A dropdown menu labeled "Experience" set to "None".
- A text area labeled "Comments" with the placeholder "Type your comments here".
- A checked checkbox labeled "Send acknowledgement".
- Buttons for "OK" and "RESET" at the bottom.

**Hình 3.13: Ví dụ một biểu mẫu đơn xin việc**

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.

**3.2.4.1 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/m joystick... hoặc dùng bàn phím để đặt.

#### 3.2.4.2 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.

Chẳng hạn:

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

```
<INPUT tabindex=6 TYPE=SUBMIT VALUE=OK>
```

#### 3.2.4.3 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.

Chẳng hạn:

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

#### 3.2.4.4 Phần tử vô hiệu hóa

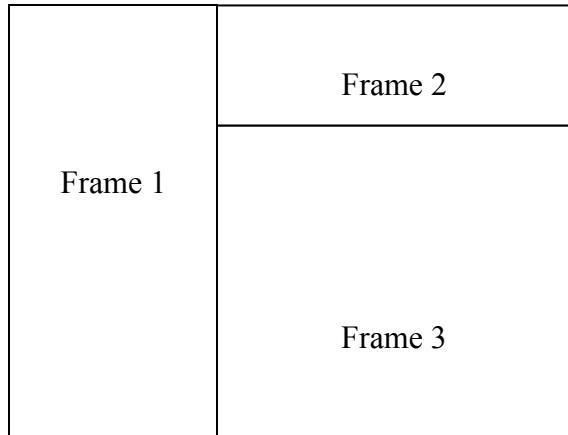
Nếu đã làm việc với trình soạn thảo văn bản, ta sẽ thấy rằng nếu không mở tài liệu nào thì các tùy chọn lưu và định dạng sẽ bị vô hiệu hóa. Đối với trang web, ta có thể vô hiệu hóa các phần tử hoặc để ở trạng thái chỉ đọc (read-only) nếu không muốn người dùng truy cập chúng. Ví dụ, khi hiển thị một biểu 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 việc 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.

Chẳng hạn:

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

### 3.3 Làm việc với khung

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 cuốn 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 cuốn độc lập nhau.



**Hình 3.11: Mô hình khung trong trang Web**

### 3.3.1 *Tại sao sử dụng khung?*

Một trang có thể có một hoặc nhiều khung. Sau đây là 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
- Đổi với bảng nội dung trong trang mà ở đó người dùng có thể kích vào và di chuyển quanh website 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 đổi 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, Internet Explorer phiên bản 2.0 hoặc trước đó và Nescape 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.

### 3.3.2 *Làm việc với khung*

#### 3.3.2.1 *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:

**Bảng 3.13: Các thuộc tính của phần tử FRAMESET**

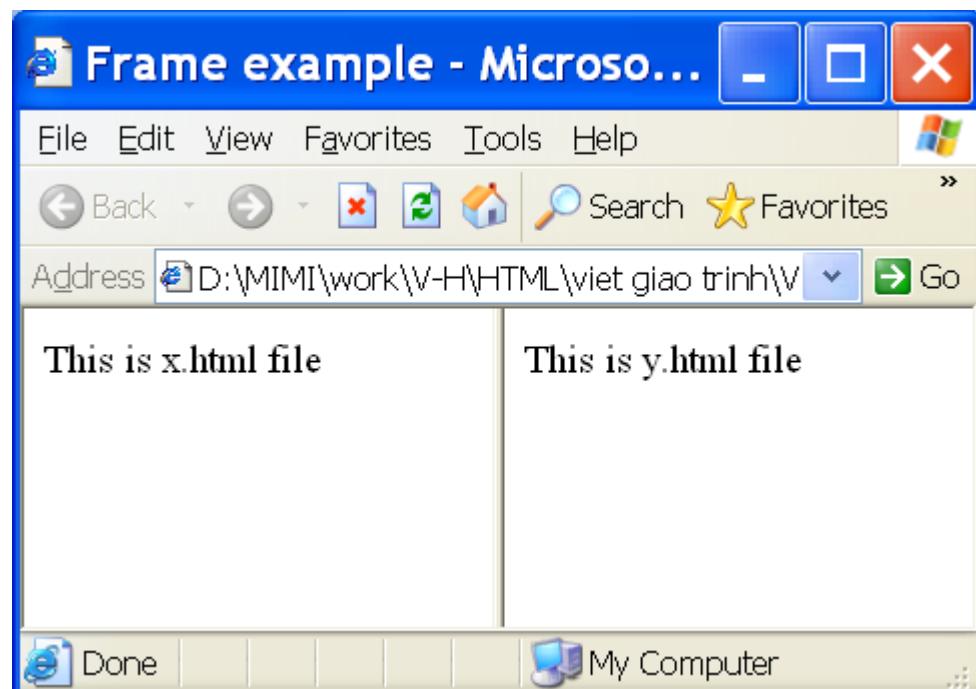
Thuộc tính	Mô tả
ROWS	Xác định độ rộng của khung, được tính theo điểm (pixels), phần trăm hoặc độ rộng tương đối. Giá trị mặc định là 100%, nghĩa là một dòng.
COLS	Xác định độ cao của khung, được tính theo điểm (pixels), phần trăm hoặc độ cao tương đối. Giá trị mặc định là 100%, nó xác định chỉ có một 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 3.13:

```
<HTML>
    <HEAD>
        <TITLE> Frame example </TITLE>
    </HEAD>
    <FRAMESET COLS="50%,*">
        <FRAME SRC=x.html>
        <FRAME SRC=y.html>
    </FRAMESET>
</HTML>
```

### Kết quả:

**Hình 3.14: Ví dụ FRAME**

Ở đây chú ý rằng file x.html và y.html được lưu cùng thư mục với file .html chính

Các thuộc tính của FRAME bao gồm:

**Bảng 3.14: Các thuộc tính của phần tử FRAME**

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 qui 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 này có thể là: <ul style="list-style-type: none"> <li>- Auto: Xuất hiện thanh cuộn khi cần thiết. Đây là giá trị mặc định.</li> <li>- Yes: Luôn luôn xuất hiện thanh cuộn trong cửa sổ của khung.</li> <li>- No: Không xuất hiện thanh cuộn trong cửa sổ của khung</li> </ul>
FRAMEBORDER	Xác định viền của khung. Các giá trị có thể là: <ul style="list-style-type: none"> <li>- 1: là giá trị mặc định. Có sự phân cách giữa khung hiện thời với các khung xung quanh.</li> <li>- 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.</li> </ul>
MARGINWIDTH	Xác định khoảng cách giữa nội dung trong khung với lề trái và lề phải của khung. Giá trị phải lớn hơn 1.
MARGINHEIGHT	Xác định khoảng cách giữa nội dung trong khung với lề trên và lề dưới của khung. Giá trị phải lớn hơn 1.

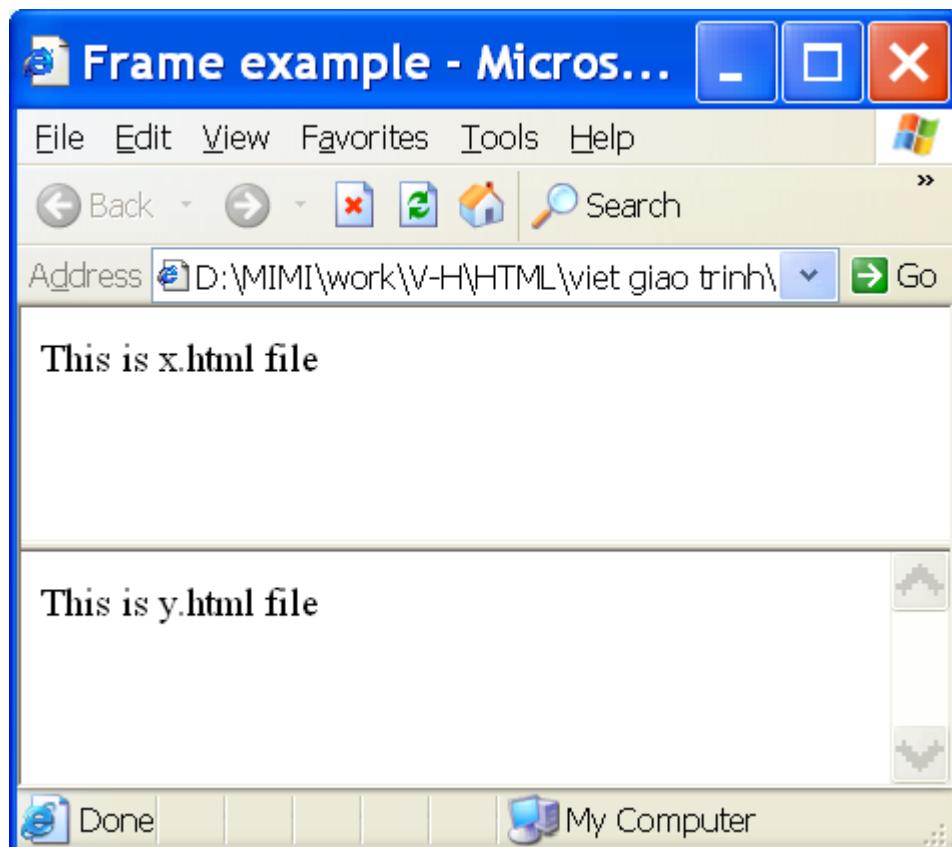
Ta không thể đóng cửa sổ khung. Khung được đóng khi cửa sổ tạo ra nó bị đóng lại. Khung không có thanh trạng thái vì vậy ta phải sử dụng thanh trạng thái của khung chính trong tài liệu.

#### Ví dụ 3.14:

```
<HTML>
  <HEAD>
    <TITLE> Frame example </TITLE>
  </HEAD>
  <FRAMESET ROWS="50%,*">
    <FRAME SRC=x.html SCROLLING=no>
```

```
<FRAME SRC=y.html SCROLLING=yes>
</FRAMESET>
</HTML>
```

**Kết quả:**



Hình 3.15: Ví dụ FRAME

**Ví dụ 3.15:**

```
<HTML>
  <HEAD>
    <TITLE> Frame example </TITLE>
  </HEAD>
  <FRAMESET COLS="20%,80%">
    <FRAMESET ROWS="100,200">
      <FRAME SRC=x.html noresize>
      <FRAME SRC=y.html>
    </FRAMESET>
    <FRAME SRC = "9shi.jpg">
  </FRAMESET>
</HTML>
```

**Kết quả:****Hình 3.16: Ví dụ FRAME**

Trong ví dụ trên, ta sử dụng phần tử FRAMESET lồng nhau (Nested framesets). Ta có thể tạo ra các frameset lồng nhau ở bất kỳ mức nào.

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 3.16:**

```
<HTML>
  <HEAD>
    <TITLE> Frame example </TITLE>
  </HEAD>
  <FRAMESET COLS="1*, 250,3*">
    <FRAME SRC=x.html >
    <FRAME SRC=y.html>
    <FRAME SRC = "9shi.jpg">
  </FRAMESET>
</HTML>
```

**Kết quả:****Hình 3.17: Ví dụ FRAME****3.3.2.2 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 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 3.17:**

&lt;HTML&gt;

&lt;FRAMESET COLS="40%, 60%"&gt;

```
<FRAMESET>
  <FRAME SRC="img7.jpg" NAME="img7" SCROLLING="yes">
  <FRAME SRC="Links.html" NAME="Links" SCROLLING="no">
</FRAMESET>

</HTML>
```

File Links.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>
```

**Kết quả:**



**Hình 3.18: Liên kết khung**

### 3.3.2.3 Phản 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. Phản tử NOFRAMES được sử dụng để làm việc đó. Nó chỉ hoạt động trong trình duyệt không hỗ trợ khung.

**Ví dụ 3.18:**

<HTML>

```
<FRAMESET COLS="40%, 60%">
  <FRAME SRC="img7.jpg" NAME="img7" SCROLLING="yes">
  <FRAMESET ROWS="60, *">
```

```

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

<FRAME SRC="y.html" NAME="y">
<NOFRAMES>
    Frames are not being displayed. Click here <A href="main.html"> for a non-frames version </A>
</NOFRAMES>
</FRAMESET>
</HTML>

```

### 3.3.2.4 Phần tử IFRAMES (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. Inline frame không thể thay đổi kích thước.

Bảng sau mô tả các thuộc tính của IFRAME:

**Bảng 3.15: Các thuộc tính của phần tử IFRAME**

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

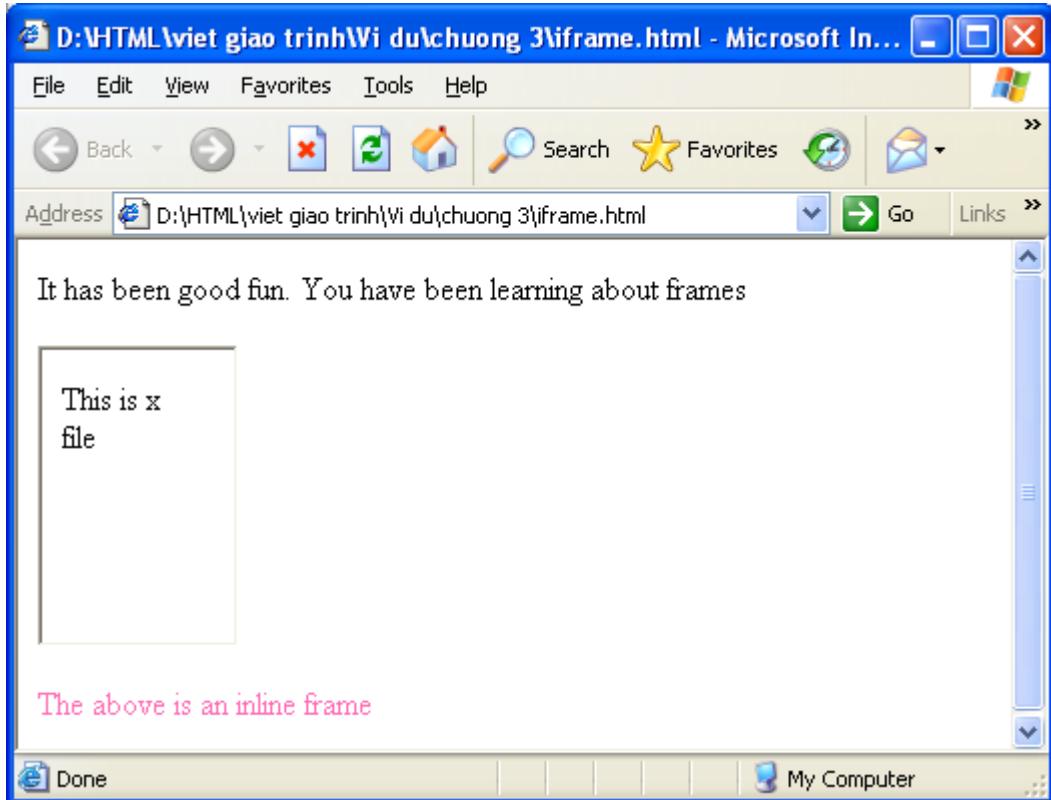
### Ví dụ 3.19:

```

<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>
</IFRAME>
<BR>
<P><FONT color=hotpink> The above is an inline frame
</HTML>

```

### Kết quả:



**Hình 3.19: phần tử IFRAME**

### 3.4 Làm việc với đa phương tiện

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 hình 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 Sử dụng hình ảnh trong tài liệu HTML

Như đã giới thiệu, ta có thể chèn hình ảnh vào một trang web. Những hình ảnh dùng để chèn vào trang web được gọi là hình ảnh nội tuyến. Ảnh có thể là biểu tượng, bullet, ảnh, logo công ty hoặc những hình mang ý nghĩa 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 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 chuyển tải khá dễ dàng, ngay cả kết nối sử dụng Modem tốc độ chậm.

- Ảnh JPEG (Joint Photographic Expert Group)

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 (có đuôi mở rộng là .jpg) và các dạng GIF đều nén ảnh để bảo đảm 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 chính là lí 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.

#### 3.4.1.1 Chèn ảnh tĩnh

Thẻ IMG được dùng để chèn những hình 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 là thuộc tính và có giá trị là một URL, chỉ định vị trí chính xác của file ảnh.

Đôi khi, chỉ hình ảnh không nói lên được 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 của ảnh với văn bản xung quanh.

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

<IMG ALIGN = position SRC = “PICTURE.GIF”>

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

#### Ví dụ 3.20:

<HTML>

<HEAD>

<TITLE> Insert an image </TITLE>

</HEAD>

<BODY>

<P><IMG ALIGN=BOTTOM SRC=“image1.jpg”> Bottom

<P><IMG ALIGN=MIDDLE SRC=“image1.jpg”> Middle

<P><IMG ALIGN=TOP SRC=“image1.jpg”> Top

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

### Kết quả:



► **Hình 3.20: Chèn ảnh tĩnh**

Ngoài ra, thuộc tính ALT được dùng để chỉ nội dung ảnh như trong ví dụ sau:

### Ví dụ 3.21:

```
<HTML>
```

```
<HEAD>
  <TITLE> Insert an image </TITLE>
</HEAD>
<BODY>
  <P><IMG ALT="Sad" SRC="image1.jpg">
</BODY>
</HTML>
```

**Kết quả:****Hình 3.21: Chèn ảnh tĩnh với lời chú thích**

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=bimage.gif>
```

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 hiệu ứng 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=bimage.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).

#### 3.4.1.2 Chèn ảnh động (.GIF) vào tài liệu HTML

Trong phần trên, chúng ta vừa thảo luận thế nào là file GIF, và nhớ rằng thẻ <IMG> được dùng để chèn một ảnh vào trong trang web.

#### Ví dụ 3.22:

```
<HTML>
```

```
<HEAD>
```

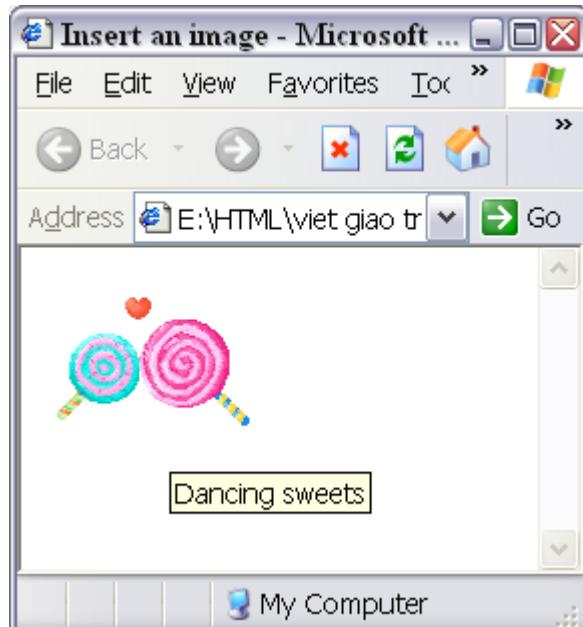
```
<TITLE> Insert an image </TITLE>
```

```

</HEAD>
<BODY>
    <IMG ALT="Dancing sweets" SRC="dacing sweets.gif">
</BODY>
</HTML>

```

**Kết quả:**



**Hình 3.22: Chèn ảnh động vào trang web**

#### 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 “bip” 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 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 âm thanh vào 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 trên, 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 hoá 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 tăng kích thước của file. Những file âm thanh được số hoá có thể được lưu trong hai định dạng, đó là file .au hay .wav.

### Ví dụ 3.23:

<HTML>

```

<HEAD>
    <TITLE> Insert sound</TITLE>
</HEAD>
<BODY>
    <BGSOUND SRC="jinglebell.wav" loop="infinite">
    <IMG ALT="Dancing sweets" SRC="dacing sweets.gif">
</BODY>
</HTML>

```

#### 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. Xem ví dụ sau:

### Ví dụ 3.24:

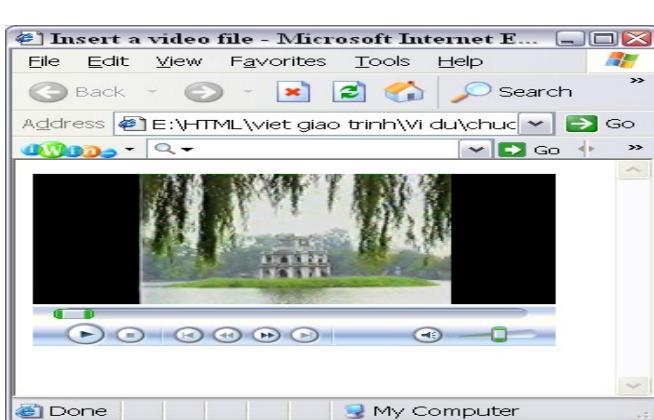
<HTML>

```

<HEAD>
    <TITLE> Insert a video file </TITLE>
</HEAD>
<BODY>
    <EMBED SRC="T0000011.avi">
</BODY>
</HTML>

```

### Kết quả:



Hình 3.23: Chèn video vào trang web

## CHƯƠNG 4

# STYLE SHEET

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. Cá 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 được 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 Style Sheet. Style Sheet là một đặc tính quan trọng có thể được sử dụng trong HTML động. Mặc dù trang web không cần có một Style Sheet, nhưng việc sử dụng Style Sheet sẽ mang lại những lợi ích nhất định. Chúng ta sẽ thảo luận về Style Sheet như là một kỹ thuật và bằng cách nào để có thể sử dụng nó trong việc thiết kế và phát triển web.

### 4.1 DHTML

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 Style Sheet và ngôn ngữ script làm cho tài liệu trở nên sinh động.

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 độc lập với nền tảng. Càng ngày, các trang web càng trở nên hấp dẫn và nhiều màu sắc hơn làm thu hút người dùng. Tuy nhiên, hình dạng 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. Lúc này, HTML vẫn còn “tĩnh”.

#### 4.1.1 *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. Các đối tượng có phương thức, thuộc tính và sự kiện có thể lập trình để điều khiển các hoạt động của chúng. Chẳng hạn như 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 sự hỗ trợ quá nhiều từ máy chủ (server). Tương tác của người dùng giờ đây có thể được xử lý bởi chính máy khách.

Lưu ý: Một ứng dụng Client/Server được tách ra hai phần 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 nó 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. Thể LAYER được dùng để cung cấp hầu hết các thuộc tính của HTML động.

#### 4.1.2 Các đặc điểm của DHTML

DHTML không dừng lại ở một số phần mở rộng của HTML. Phần này sẽ trình bày 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 trước đây của HTML, 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. 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 qui trình rõ ràng. Tuy nhiên, nếu trang đó phải được nạp lại thường xuyên sẽ tốn nhiều thời gian và làm cho máy chủ trở nên quá tải.

Trong DHTML thì cách làm này khác một chút. Bằng cách dùng các bảng kiểu (style sheet), chúng ta có thể xác định màu, kiểu hoặc kích cỡ của nội dung trang. Ngoài ra 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 DHTML, kiểu liên quan đến cách thức, định dạng xuất hiện trên trang web hơn là nội dung. 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 nhạo 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 web xét 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 DHTML, chúng ta có thể chỉ ra vị trí chính xác (tuyệt đối hay tương đối), mỗi 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 (pixel).
- 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 DHTML, 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.
- **Khả năng tải font chữ (Downloadable Fonts):** Được Netscape hỗ trợ. Đây là một đặc 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 huỷ 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):** DHTML theo một cấu trúc trúc đối tượng, nghĩa là 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.

## 4.2 Style sheet

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

### 4.2.1 Khái niệm, chức năng và các lợi ích của style sheet

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

Có hai cách để thay đổi kiểu trên trang web như sau:

- Thay đổi kiểu nội tuyến
- Viết kịch bản để thay đổi kiểu

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

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

```
<H1 style = "color: red">
```

Nếu muốn sử dụng kịch bản để thay đổi kiểu vào bất cứ lúc nào, thì ta 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, ta thực hiện như sau:

- 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.

Chẳng hạn, font-weight của CSS trở thành fontWeight trong DHTML, hoặc text-align thành textAlign

#### Ví dụ 4.1:

```
<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
```

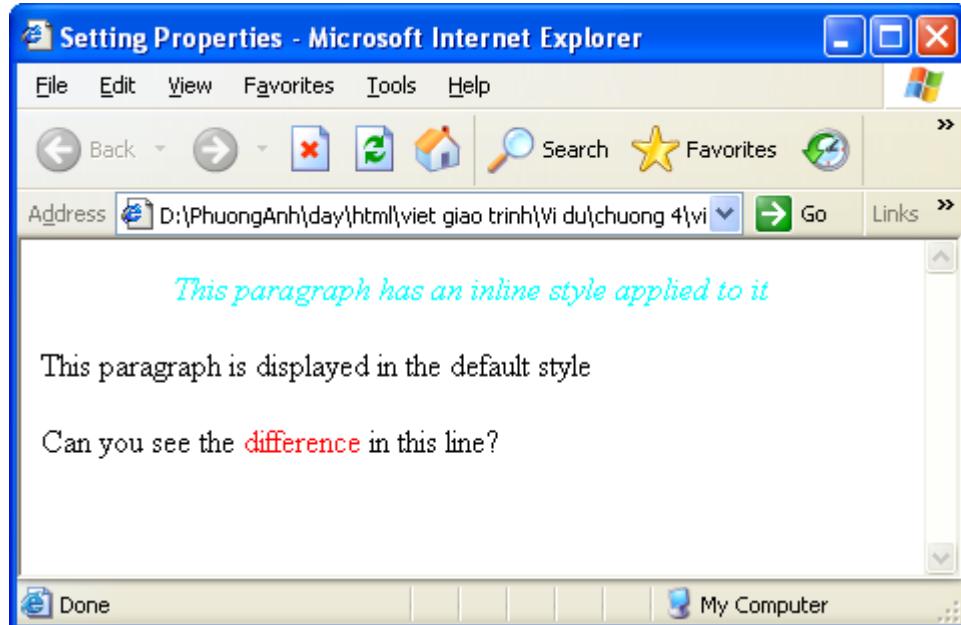
```
        <BR>
```

```
        <P> Can you see the <SPAN style=color:red> difference </SPAN> in this  
        line?
```

```
    </BODY>
```

```
</HTML>
```

#### Kết quả:



**Hình 4.1: Kết quả ví dụ 4.1**

**Lợi ích của style sheet:** Các style sheet 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, bởi vì họ không thể biết được trình duyệt nào mà người dùng cách nửa vòng trái đất sử dụng. Nhờ có các style sheet, ta có thể nạp chồng các quy ước của trình duyệt và đặt theo cách riêng của chúng ta. Chẳng hạn, có thể xác định kiểu mà trong đó một phần tử <H1> cần hiển thị như sau:

```
<H1>
<FONT SIZE = 3 COLOR = aqua>
    <B> Overriding the browser </B>
</FONT>
</H1>
```

- Bố cục trang (Page layout):** Những style sheet 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 chuyện 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 bất kỳ tài liệu HTML. Lần lượt thay thế, chúng ta có thể kết nối tất cả cá trang trên website đến stylesheet. Điều này chắc chắn rằng các trang web của chúng ta đều có cùng diện mạo khi thông tin 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 các 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.

#### 4.2.2 Quy tắc stylesheet

Stylesheet phân cấp (Cascading Style Sheet) định nghĩa các kiểu có thể áp dụng vào trang hoặc các phần tử của trang.

**Quy tắc kiểu (Style Rule):** Stylesheet phân cấp là một tập hợp các quy tắc. Quy tắc định nghĩa kiểu của tài liệu. Ví dụ, chúng ta có thể tạo ra một quy tắc kiểu được xác định cho tất cả phần tiêu đề `<H1>` hiển thị màu vàng xanh. Quy tắc kiểu có thể được áp dụng vào các thành phần được chọn của trang web. Ví dụ, chúng ta sẽ 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 của HTML đơn lẻ trên một trang web.

**Style Sheet:** Là một danh mục các quy 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 quy tắc (Rules):** Bảng kiểu có thể có một hay nhiều quy tắc. Phần đầu của quy 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 quy tắc được kèm theo trong phạm vi các dấu ngoặc mỏc 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.

Chẳng hạn 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 quy tắc có thể tách rời nhau trong phạm vi thẻ STYLE.

**Ví dụ 4.2:**

```
<HTML>
```

```
<HEAD>
```

```

<STYLE TYPE="text/css">
    H1 {color: limegreen}
    H1 {font-family: Arial}
    H2 {color:limegreen}
    H2 {font-family: Arial}
</STYLE>

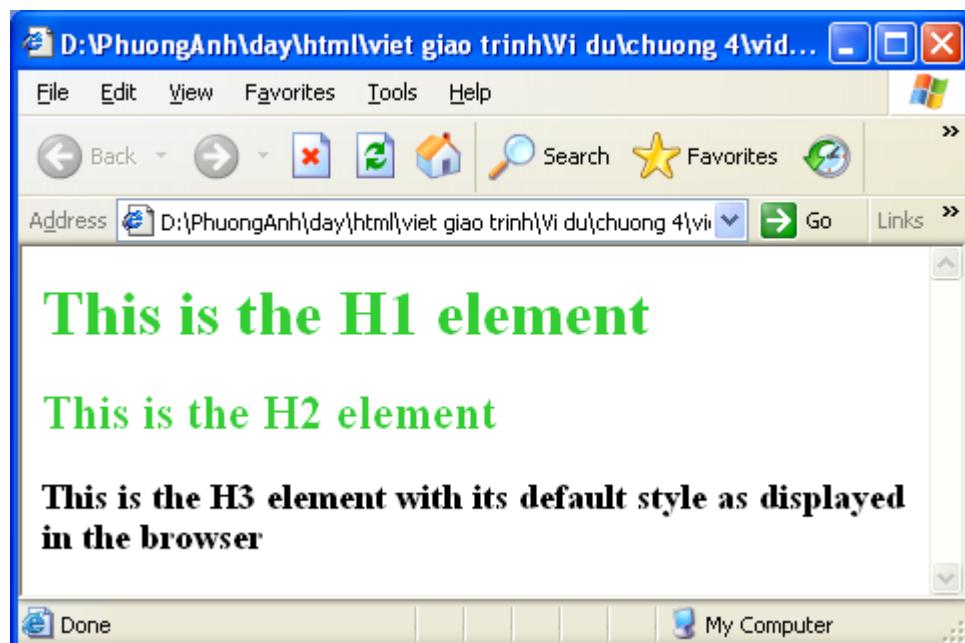
</HEAD>

<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>

```

**Kết quả:**



**Hình 4.2: Kết quả ví dụ 4.2**

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

**Ví dụ 4.3:**

```

<HTML>

<HEAD>

<STYLE TYPE="text/css">
    H1, H2 {color: limegreen; font-family: Arial;}
</STYLE>

</HEAD>

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

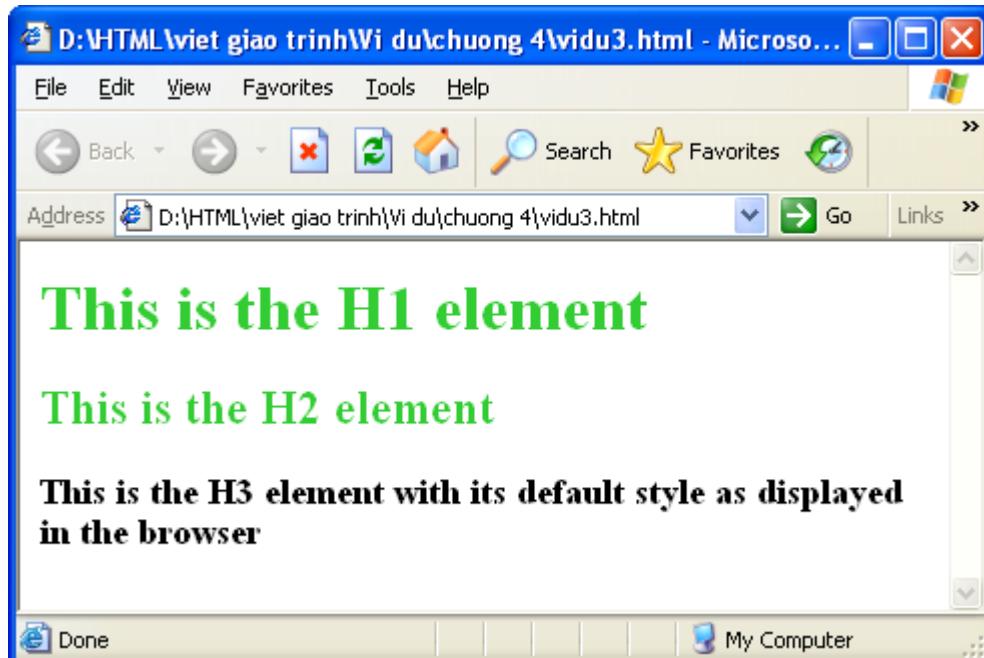
```

```

<H2> This is the H2 element </H2>
<H3> This is the H3 element with its default style as displayed in the browser
</H3>
</HTML>

```

**Kết quả:**



Hình 4.3: Kết quả ví dụ 4.3

#### 4.2.3 Các Selector trong style sheet

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. Chẳng hạn, với việc sử dụng các lớp giả, các liên kết đã được ghé qua có hiển thị khác đi so với các liên kết chưa được ghé qua như sau:

```

A:link {color:red} /*liên kết chưa được ghé qua
A:visited {color:blue} /*liên kết đã ghé qua
A:active {color:lime} /*liên kết hiện thời

```

Selector có thể định nghĩa như là “một chuỗi ký tự xác định ra các phần tử và các qui 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 mòc. Vì vậy, thẻ <P> trong HTML trở thành P và khi đó, nó được xem như là một selector. Ví dụ sau minh họa điều đó:

#### Ví dụ 4.4:

```
<HTML>
```

```
<HEAD>
```

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

```
P {font-style:italic;font-weight:bold; color:limegreen}
```

```
</STYLE>
```

```
</HEAD>
```

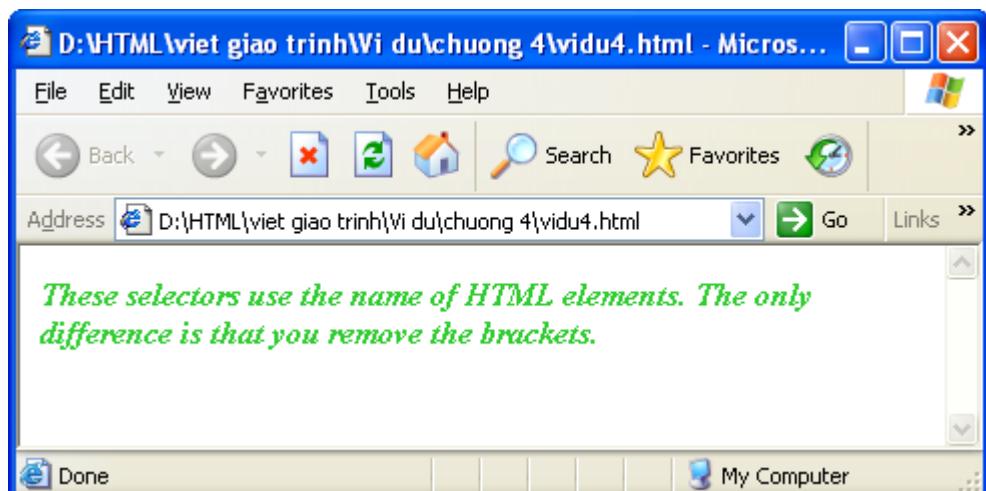
```
<BODY>
```

<P> These selectors use the name of HTML elements. The only difference is that you remove the brackets. </P>

```
</BODY>
```

```
</HTML>
```

#### Kết quả:



**Hình 4.4: Kết quả ví dụ 4.4**

- *Selector lớp*

Những selector này sử dụng thuộc tính CLASS của 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. Chẳng hạn, ta có thể muốn thẻ <H2> xuất hiện với nhiều màu khác nhau. Trong trường hợp đó, ta sử dụng định danh lớp cho thẻ <H2>.

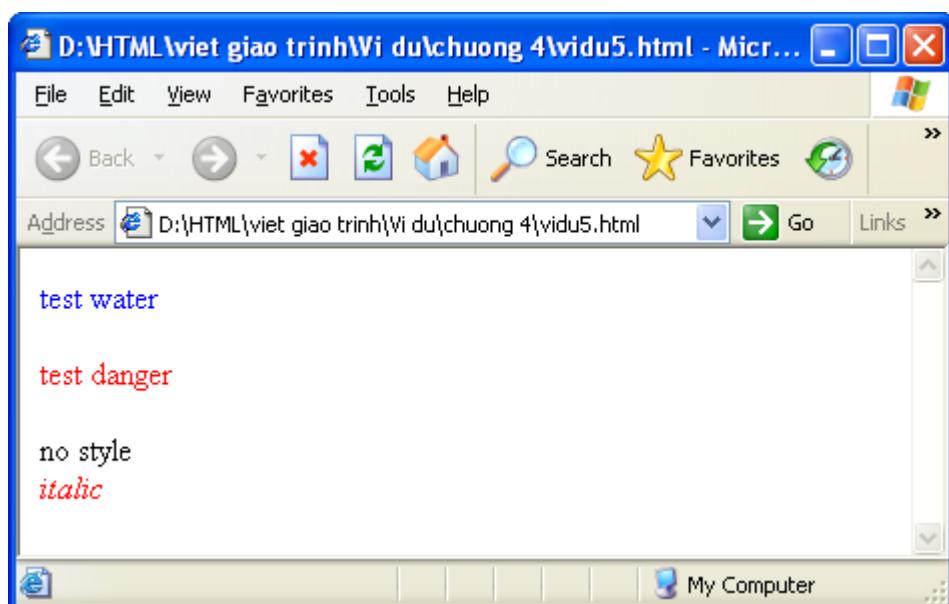
Selector lớp có dấu chấm đứng trước gọi là ký tự cờ, theo sau là tên lớp do chúng ta chọn. Tốt hơn hết nên chọn tên lớp theo mục đích của chúng chứ không nên chọn theo tên mô tả màu sắc hay kiểu. Chẳng

hạn, ta có thể muốn đoạn A hiển thị chữ nghiêng, những đoạn khác thì hiển thị kiểu khác. Trong trường hợp đó, đoạn A có thể có bộ nhận dạng lớp là .slant

#### Ví dụ 4.5:

```
<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>
```

#### Kết quả:



Hình 4.5: Kết quả ví dụ 4.5

Khi xác định một lớp kiểu, ta có thể xác định được phần tử HTML nào có thể sử dụng kiểu này.

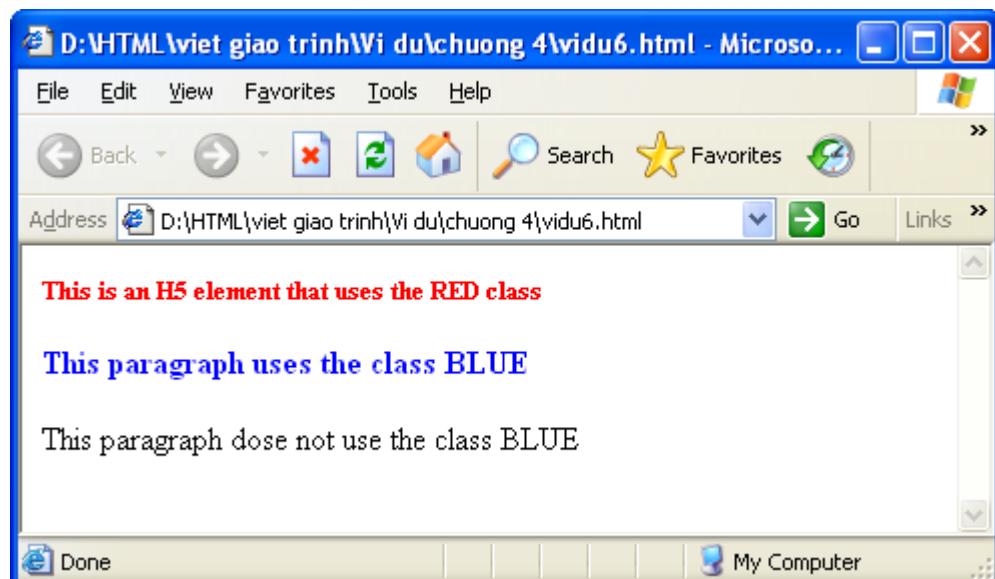
#### Ví dụ 4.6:

```

<HTML>
  <HEAD>
    <STYLE TYPE="text/css">
      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>
  <BODY>
    <P class=blue>This paragraph uses the class BLUE </P>
    <P>This paragraph dose not use the class BLUE </P>
  </BODY>
</HTML>

```

### Kết quả:



**Hình 4.6: Kết quả ví dụ 4.6**

- *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. Chẳng hạn, 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 # ( # ).

### Ví dụ 4.7:

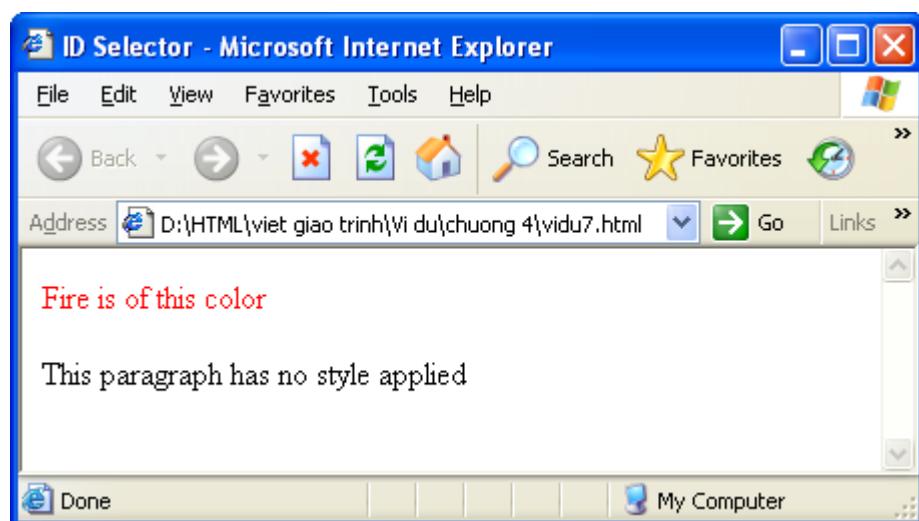
```
<HTML>
```

```

<HEAD>
    <TITLE> ID Selector </TITLE>
    <STYLE TYPE="text/css">
        #control {color:red}
    </STYLE>
</HEAD>
<BODY>
    <P id="control"> Fire is of this color
    <BR>
    <P>This paragraph has no style applied
</BODY>
</HTML>

```

**Kết quả:**



**Hình 4.7: Kết quả ví dụ 4.7**

**Ví dụ 4.8:**

```

<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>

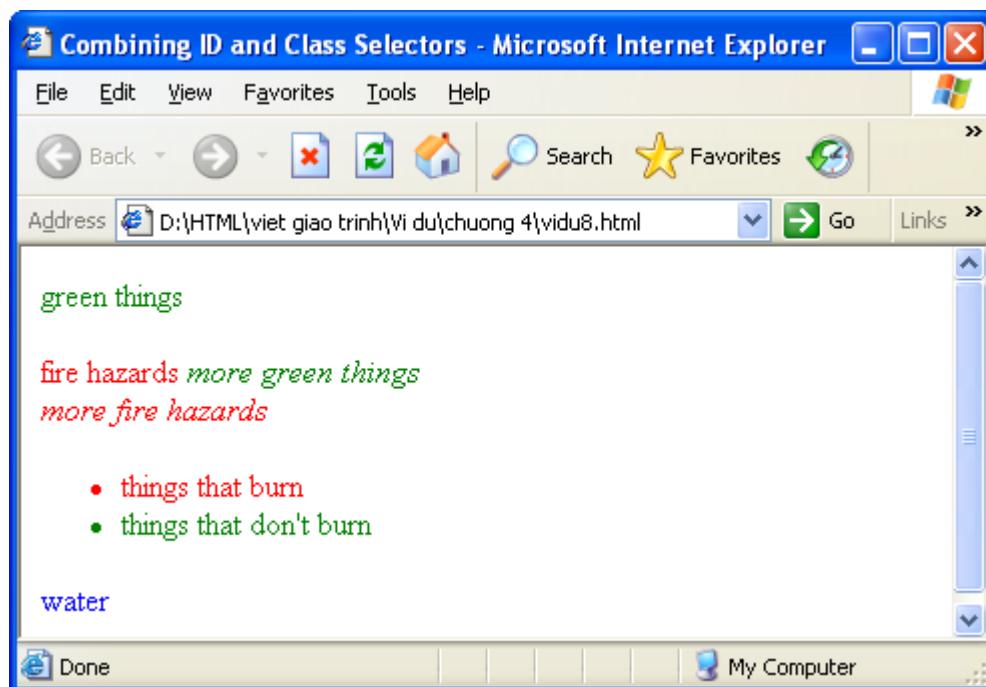
```

```

<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
</BODY>
</HTML>

```

### Kết quả:



Hình 4.8: Kết quả ví dụ 4.8

- ***Selector ngữ cảnh***

Selector ngữ cảnh liên quan đến ngữ cảnh của phần tử. Chẳng hạn, 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,

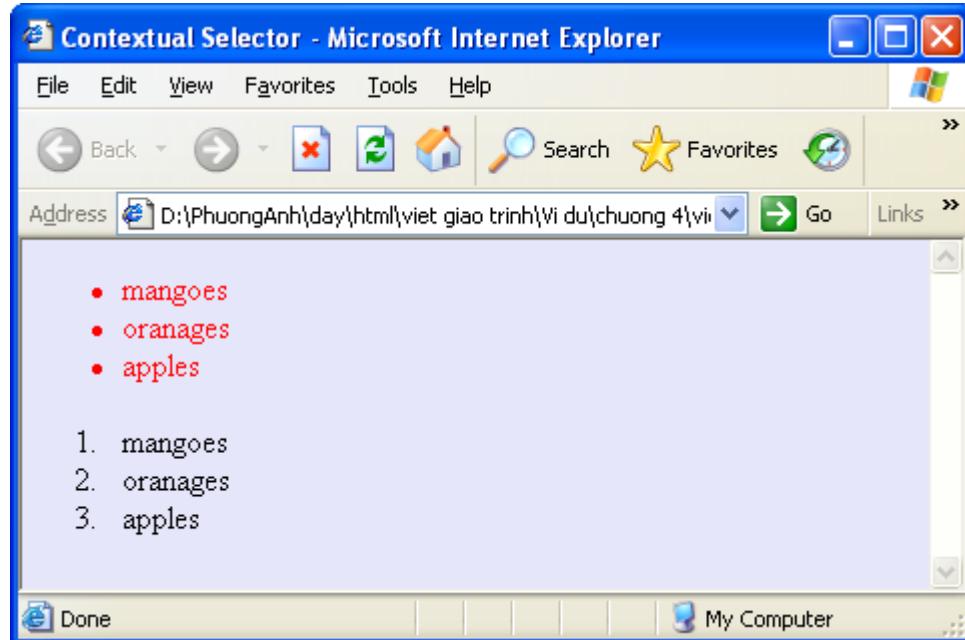
chúng ta phải có sự thay đổi đặc biệt đối với các phần tử con, nói một cách khác đó là sự nạp chồng kế thừa.

**Ví dụ 4.9:**

&lt;HTML&gt;

```
<HEAD>
    <TITLE> Contextual Selector </TITLE>
    <STYLE TYPE="text/css">
        BODY {color:blue; background:lavender;
            font-family:Arial;}
        UL {color:red}
    </STYLE>
</HEAD>
<BODY>
    <UL>
        <LI> mangoes
        <LI> oranages
        <LI> apples
    </UL>
    <OL>
        <LI> mangoes
        <LI> oranages
        <LI> apples
    </OL>
</BODY>
</HTML>
```

**Kết quả:**



**Hình 4.9: Kết quả ví dụ 4.9**

Selector UL trong style sheet xác định các mục trong danh sách (list item) được hiển thị màu đỏ. Chúng kế thừa font 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.

#### 4.2.4 Kết hợp và chèn một style sheet vào tài liệu HTML

Có một số cách giúp ta có thể kết hợp style sheet với HTML, đó là:

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 qui tắc được định nghĩa giữa thẻ mở và thẻ đóng. Khi hiển thị các trang, chỉ tài liệu nào có nhúng STYLE mới được tác động. Ví dụ sau minh họa cho cách sử dụng phần tử Style:

```
<Style Type = "text/css">
    H1 {color:maroon;}
    P {color:hotpink; font-family:Arilal;}
</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 sau là cách sử dụng thuộc tính Style:

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

Rõ ràng là chúng ta chỉ nên dùng kiểu này khi muốn 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.

- *Phần tử Link*

Nếu ta muốn sử dụng phần tử Link, thì style sheet của ta phải là một tài liệu riêng. Sau đó chúng ta phải thêm địa chỉ web của style sheet vào. Chẳng hạn:

```
<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 style sheet.

**Ví dụ 4.10:**

*File sheet1.css:*

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

*File .html*

```
<HTML>
```

```
<HEAD>
```

```
<TITLE> Linking extenal style sheet</TITLE>
```

```
<LINK REL=STYLESHEET TYPE="text/css" HREF="sheet1.css">
```

```
</HEAD>
```

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

```
<BR>
```

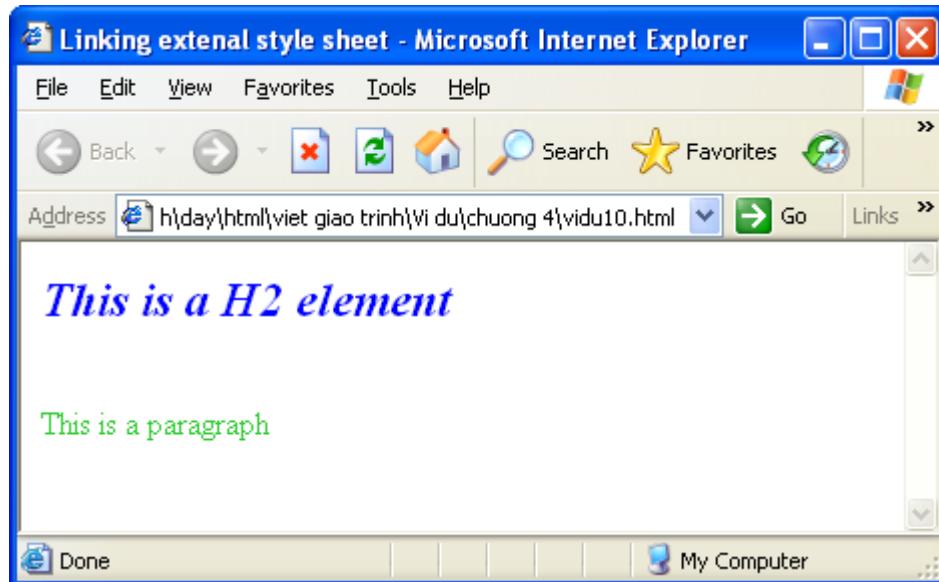
```
<BODY>
```

```
<P>This is a paragraph
```

```
</BODY>
```

```
</HTML>
```

**Kết quả:**



**Hình 4.10: Kết quả ví dụ 4.10**

#### 4.2.5 Thiết lập thuộc tính trong style sheet

Chúng ta có thể thiết lập nhiều thuộc tính trong style sheet. Bảng sau cho thấy nhiều thuộc tính có thể được sử dụng trong style sheet:

**Bảng 4.1: Các thuộc tính thường được sử dụng trong style sheet**

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 đường viền	border
	border-width
	border-bottom
	border-color
Các thuộc tính về vị trí	Clip
	height
	left
	top
	z-index

## CHƯƠNG 5

# TỔNG QUAN VỀ JAVASCRIPT

### 5.1 Giới thiệu về Javascript

#### 5.1.1 Javascript là gì?

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

Javascript là ngôn ngữ dưới dạng script có thể gắn với các file HTML. Nó là ngôn ngữ dựa trên đối tượng, có nghĩa là bao gồm nhiều kiểu đối tượng, ví dụ đối tượng Math với tất cả các chức năng toán học. Javascript có thể đáp ứng các sự kiện như tải hay loại bỏ các form. Khả năng này cho phép Javascript trở thành một ngôn ngữ script động.

Giống với HTML và Java, Javascript được thiết kế độc lập với hệ điều hành. Nó có thể chạy trên bất kỳ hệ điều hành nào có trình duyệt hỗ trợ Javascript. Các trình duyệt web như Netscape Navigator 2.0 hay Internet Explorer 3 trở đi có thể hiển thị những câu lệnh Javascript được nhúng vào trang HTML. Khi trình duyệt yêu cầu một trang, sever sẽ gửi đầy đủ nội dung của trang đó, bao gồm cả mã lệnh HTML và các câu lệnh Javascript qua mạng tới client. Client sẽ đọc trang đó từ đầu đến cuối, hiển thị các kết quả của HTML và xử lý các câu lệnh Javascript khi nào chúng xuất hiện.

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

*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.

#### 5.1.2 Tìm hiểu lịch sử của JavaScript

JavaScript được hình thành xuất phát từ nhu cầu kết hợp các trang web HTML với nội dung được nhúng vào trong đó, chẳng hạn như Java applet. Tuy nhiên, JavaScript được sử dụng cho nhiều mục đích khác nữa. Nó thường được dùng để giúp người sử dụng di chuyển vào các biểu mẫu trực tuyến, cung cấp hệ thống di chuyển trong website thông qua các menu động và các giờ mua hàng trực tuyến. Trong thực tế, có khoảng 25% website sử dụng JavaScript theo cách này hoặc cách khác.

Khi tìm hiểu tốc độ phát triển nhanh chóng để bổ sung những đặc điểm mới của các công nghệ liên qua đến web, chúng ta thấy rằng JavaScript khá ổn định. Cần 8 năm để JavaScript phát triển từ phiên bản 1.0 đến phiên bản kế tiếp là 2.0. Nhiều người cảm thấy bước phát triển chậm chạp này vừa là một điều may mắn, vừa là điều không may mắn của ngôn ngữ này.

Điều may mắn là sự hỗ trợ JavaScript tương đối nhất quán qua nhiều trình duyệt và các phiên bản khác nhau. Các nhà phát triển web có thể sử dụng một chương trình JavaScript mà không phải lo lắng nhiều về vấn đề tương thích (ngoại trừ một số kỹ thuật giải quyết vấn đề không tương thích nhất định). JavaScript đã dành được sự ứng dụng rộng rãi chủ yếu là nhờ các nhà phát triển tin tưởng rằng nó sẽ làm được việc.

Điều không may mắn là trong khi JavaScript đang đứng yên thì các ngôn ngữ khác đã phát triển để lấp đầy những khoảng trống mà ngôn ngữ này còn khiêm khuyết. Có rất nhiều website hiện nay đã sử dụng VBScript hoặc Java Server Pages (JSP) như là một ngôn ngữ kịch bản máy chủ thay vì sử dụng JavaScript. Trong thực tế, phiên bản mới nhất của một trong những phần mềm máy chủ web phổ biến (iPlanet Web Server) đã không còn tiếp tục hỗ trợ JavaScript nữa. Tuy nhiên, điều này có thể thay đổi khi JavaScript 2.0 đón nhận được sự chú ý của các nhà phát triển.

#### 5.1.2.1 Nguồn gốc của JavaScript

JavaScript lần đầu tiên xuất hiện trong phiên bản Netscape 2.0 vào năm 1995. JavaScript lúc đó chủ yếu được thiết kế để giúp đỡ việc tích hợp các trang HTML với các Java Applet - một dạng ứng dụng của Java được nhúng trong các trang web. Tuy nhiên, các nhà phát triển nhanh chóng nhận ra sức mạnh tiềm tàng thực sự của nó và rất nhanh chóng, JavaScript được sử dụng để bổ sung tính tương tác cho các website - phần lớn trường hợp đều không cần đến sự hỗ trợ của Java.

#### 5.1.2.2 JavaScript đến với Internet Explorer

Không bao lâu sau khi Netscape Navigator giới thiệu JavaScript trong trình duyệt Navigator 2.0 của mình, Microsoft đã nhận thấy tầm quan trọng của việc kết hợp ngôn ngữ này vào trong trình duyệt Internet Explorer của họ. Vì Netscape không gửi cho Microsoft mã nguồn và thậm chí mô tả kỹ thuật của ngôn ngữ này còn được bảo vệ kỹ lưỡng, Microsoft buộc phải tạo ra một phiên bản của riêng mình. Microsoft đặt tên cho sản phẩm này là Jscript vì Netscape đã đăng ký bản quyền cho từ JavaScript.

Các phiên bản ban đầu của JScript không thực hiện các chức năng theo đúng cách mà JavaScript thực hiện và vì thế tính không tương thích JavaScript giữa hai trình duyệt là điều mà các nhà phát triển phải tính tới khi lập trình cho các trang web của mình.

#### 5.1.2.3 JavaScript trở thành chuẩn chính thức

Trong những ngày đầu của web, tính tương thích giữa các trình duyệt là một vấn đề lớn - lớn hơn rất nhiều so với ngày nay. Hai hãng cung cấp trình duyệt chủ yếu đã thực hiện các thay đổi về HTML và ngôn ngữ Script để cố gắng dành được sự vượt trội của mình so với đối thủ và điều này đã gây ra không ít điều phiền toái cho các nhà phát triển web trong việc cố gắng tạo ra các website có thể hỗ trợ cho cả hai loại trình duyệt. Thật may mắn cho chúng ta là cả hai hãng này đã bớt gay gắt với nhau.

Netscape đã khôn khéo chuyển JavaScript thành chuẩn ECMA (European Computer Manufacturers Association) vào năm 1996. ECMA tập trung chủ yếu vào việc chuẩn hóa phần lõi của ngôn ngữ và để lại những phần khác, chẳng hạn như DOM – JavaScript Document Object Model, cho các nhà phát triển trình duyệt. Kết quả là tính tương thích tiếp tục tồn tại giữa hai trình duyệt.

ECMA công bố ngôn ngữ kịch bản được chuẩn hóa gọi là ECMAScript vào năm 1997. Sau đó, họ cập nhật chuẩn này hai năm một lần với tên gọi Edition 2 và Edition 3. JavaScript 1.5 tuân thủ chuẩn Edition 3.

#### *5.1.2.4 JavaScript hiện nay đã phát triển đến đâu?*

Chuẩn ECMAScript Edition 4 lần đầu tiên được công bố sau khoảng 4 năm. JavaScript 2.0 tuân theo bản Edition 4 của chuẩn ECMAScript và sự khác biệt giữa hai loại trình duyệt còn lại rất nhỏ.

Ngày nay, JavaScript của Netscape và JScript của Microsoft đều tuân theo chuẩn ECMAScript, mặc dù mỗi ngôn ngữ vẫn còn hỗ trợ một số đặc điểm không có trong chuẩn.

**Bảng 1.1: Liệt kê danh sách các phiên bản của JavaScript kèm theo danh sách những trình duyệt phổ biến có hỗ trợ cho từng phiên bản**

Phiên bản	Ngày công bố	Trình duyệt	Tương thích chuẩn
1.0	12-1995	Navigator 2, Internet Explorer 3	Không
1.1	4-1996	Navigator 4, Internet Explorer 4	Một phần chuẩn ECMAScript 1
1.2	12-1996	Navigator 4.06, Internet Explorer 5	ECMAScript 1, ISO - 16262
1.3	8-1998	Không xuất hiện trong bất kỳ trình duyệt nào.	ECMAScript 1, ISO - 16262
1.5	4-2000	Navigator 6 và 7, Internet Explorer 5.5 và 6, Mozilla 1	ECMAScript 3
2.0	2003		ECMAScript 4

*Bản đặc tả kỹ thuật dành cho JavaScript 2.0 có thể được tìm thấy tại website Mozilla.org: <http://www.mozilla.org/js/language/js20/index.html>.*

#### **5.1.3 Nhúng Javascript vào file HTML**

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

- Nhúng các câu lệnh trực tiếp vào trong tài liệu bằng cách sử dụng thẻ <SCRIPT>.

- Liên kết file nguồn Javascript với tài liệu HTML
- Đặt các biểu thức Javascript làm giá trị cho thuộc tính của thẻ HTML
- Dùng như trình xử lý sự kiện trong các thẻ HTML

#### 5.1.3.1 Dùng thẻ <SCRIPT>

Mã Javascript được nhúng vào trong tài liệu HTML bằng thẻ <SCRIPT>. Chúng ta có thể nhúng nhiều script vào trong 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 thẻ chú thích. Những thẻ này được dùng để báo cho trình duyệt bỏ qua các câu lệnh chứa trong nó. <! là thẻ mở , //--> là thẻ đóng của thẻ chú thích. Các thẻ này không bắt buộc phải có, nhưng ta nên đưa chúng vào trong các đoạn script, bởi vì 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.
```

Ví dụ sau nhúng Javascript vào trong trang web. Trong ví dụ này, 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 5.1:

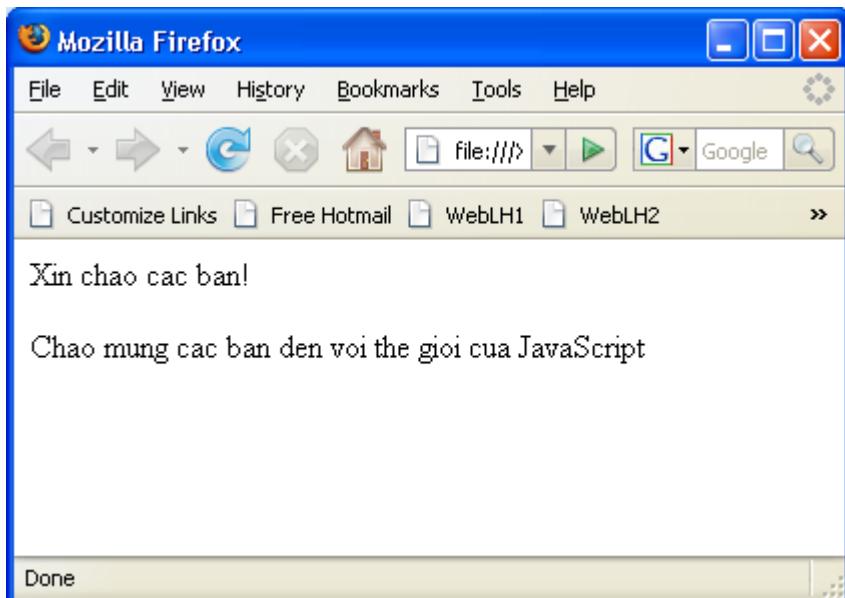
```
<HTML>
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
<!-- Phiên bản cũ sẽ bỏ qua câu lệnh tiếp theo>
document.write("Xin chao cac ban!");
//va bat dau thuc hien tu day -->
</SCRIPT>
```

```

</HEAD>
<BODY>
    <P> Chao mung cac ban den voi the gioi cua JavaScript
</BODY>
</HTML>

```

**Kết quả:**



**Hình 5.1: Nhúng JavaScript vào trong trang Web**

Trên lý thuyết các câu lệnh JavaScript có thể được 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.

#### 5.1.3.2 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 đoạn mã JavaScript mà nó cần sử dụng. Khi xác định file nguồn, ta có thể dùng tên đường dẫn tương đối hoặc tuyệt đối trong thuộc tính SRC.

```

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

```

Trong đó, filename 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 “vidu.html” là một file tài liệu HTML, file thứ hai “vidu.js” là file văn bản có chứa mã JavaScript. File này được liên kết với file thứ nhất.

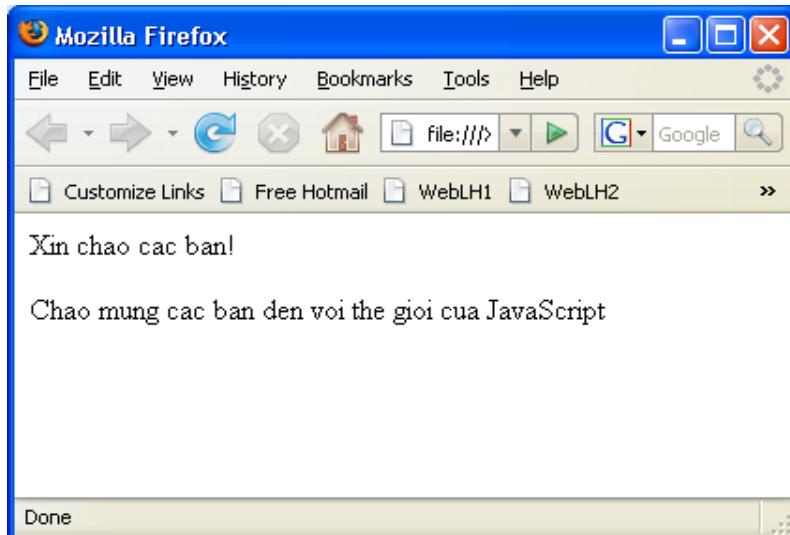
#### Ví dụ 5.2:

File vidu.html:

```
<HTML>
  <HEAD>
    <SCRIPT LANGUAGE="JavaScript" src = "vidu.js">
    </SCRIPT>
  </HEAD>
  <BODY>
    <P> Chao mung cac ban den voi the gioi cua JavaScript
  </BODY>
</HTML>
```

File vidu.js:

```
document.write ("Xin chao cac ban!")
```

**Kết quả:**

**Hình 5.2: Dùng file JavaScript liên kết với file HTML**

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 sẽ 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 thực hiện trên nhiều tài liệu HTML.

#### 5.1.3.3 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 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 5.4:

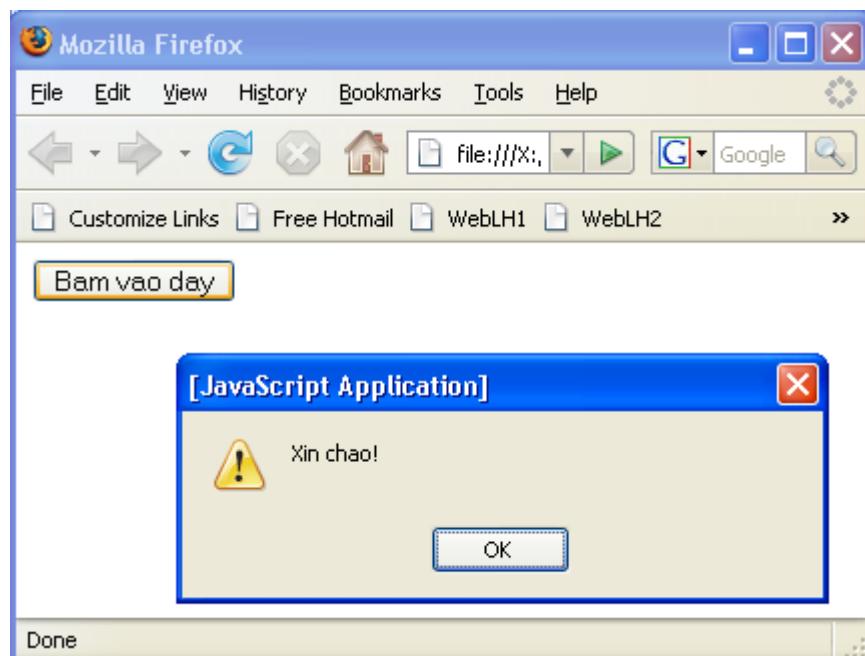
<HTML>

```

<HEAD>
    <SCRIPT>
        function vidu(){
            alert ("Xin chao!");
        }
    </SCRIPT>
</HEAD>
<BODY>
    <FORM>
        <INPUT TYPE="button" VALUE="Bam vao day"
               onClick="vidu()">
    </FORM>
</BODY>
</HTML>

```

#### Kết quả:



Hình 5.4: Dùng JavaScript trong trình xử lý sự kiện

#### 5.1.4 Thể <NOSCRIPT> và </NOSCRIPT>

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

### Ví dụ 5.5:

<NOSCRIPT>

<B> Trang này có sử dụng JavaScript. Do đó bạn cần sử dụng trình duyệt Netscape Navigator từ version 2.0 trở đi

<BR>

<A HREF="<http://home.netscape.com/comprd/mirror/index.html>">

Hãy kích chuột vào đây để tải về phiên bản Netscape mới hơn

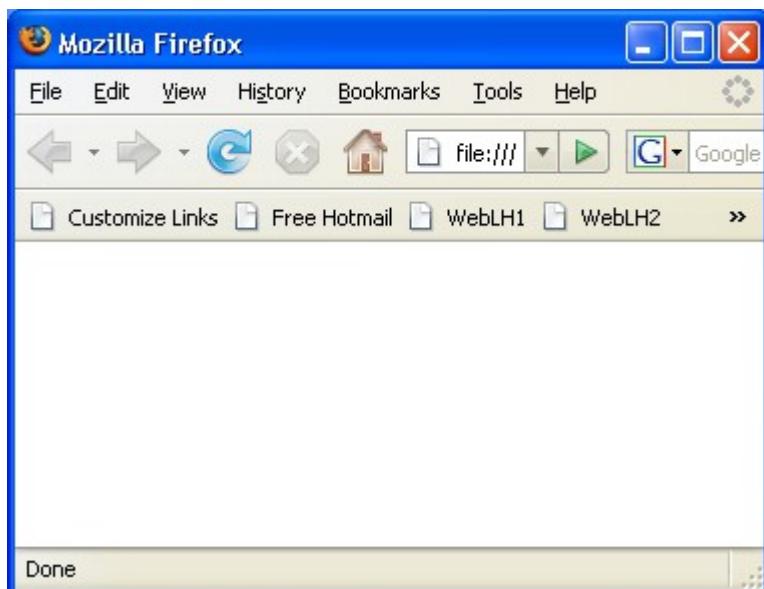
</A>

</BR>

Nếu bạn đã sử dụng trình duyệt Netscape từ 2.0 trở đi mà vẫn đọc được những dòng chữ này thì hãy bật Preferences/Advanced/JavaScript lên

</NOSCRIPT>

### Kết quả:



**Hình 5.5: Minh họa thẻ <NOSCRIPT> với Internet Explorer**

## 5.2 Biến, hằng và các kiểu dữ liệu trong JavaScript

### 5.2.1 Biến và phân loại 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 sử dụng các biến như tên tượng

trung cho các giá trị trong ứng dụng. Tại mỗi thời điểm thực hiện, biến có thể chứa một giá trị mới. Khi muốn xem, sử dụng hay thay đổi giá trị của biến, ta chỉ cần dùng tên của biến.

Các biến trong JavaScript phải tuân thủ các qui ước đặt tên sau:

- Tên biến phải được bắt đầu bằng một chữ cái, dấu gạch dưới (\_), hoặc một dấu đôla (\$).
- Các kí tự tiếp theo có thể là chữ số (0-9) hoặc chữ cái.

Ở đây lưu ý rằng JavaScript có phân biệt chữ hoa và chữ thường, nên tên biến chứa chữ hoa và chữ thường sẽ là khác nhau. Ví dụ biến “tong” sẽ khác với biến “Tong”.

### **Khai báo biến:**

Chúng ta sử dụng từ khóa “var” để khai báo biến, đồng thời cũng có thể khởi tạo giá trị cho biến ngay khi khai báo:

Ví dụ: var A = 5;

Ta cũng có thể khai báo biến bằng cách gán giá trị cho nó mà không cần từ khóa “var”

Ví dụ: B = 7;

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

### **Phạm vi của biến:**

Phạm vi của biến được xác định tại vị trí mà nó được khai báo trong script. Khi một biến được khai báo ngay phần đầu của script, thì nó được xem là một *biến toàn cục*, bởi vì nó có thể được sử dụng mọi nơi trong ứng dụng hiện thời. Nếu ta khai báo một biến bên trong một hàm, biến đó được gọi là *biến cục bộ*, bởi vì nó chỉ được sử dụng bên trong hàm đó.

Sử dụng từ khóa “var” để khai báo một biến toàn cục là tùy ý, tuy nhiên, bạn phải sử dụng từ khóa “var” để khai báo một biến cục bộ.

Bạn có thể truy xuất các biến toàn cục đã khai báo trong một cửa sổ hoặc một khung từ một cửa sổ hoặc một khung khác bằng cách chỉ ra tên của cửa sổ hoặc khung đó. Ví dụ, nếu một biến có tên là phoneNumber được khai báo trong một tài liệu FRAMESET, thì bạn có thể tham khảo đến biến này từ một khung con dưới dạng: parent.phoneNumber.

### **5.2.2 Hằng**

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

Chúng ta có thể tạo ra một hằng số chỉ đọc cùng với tên của nó bằng cách sử dụng từ khóa const. Quy ước đặt tên cho hằng giống như cho tên biến, tức là nó phải được bắt đầu với một chữ cái hoặc dấu gạch dưới và các ký tự còn lại có thể bao gồm các ký tự thuộc bảng chữ cái, các số hay dấu gạch dưới.

Ví dụ: const prefix = ‘212’;

Một hằng số không thể thay đổi giá trị qua phép gán hoặc được khai báo lại trong khi script đang thi hành. Các quy tắc về phạm vi cho các hằng số giống như cho các biến, ngoại trừ từ khóa const luôn luôn được yêu cầu, ngay cả các hằng số toàn cục. Nếu từ khóa này bị bỏ quên thì đó được xem như là một biến.

Lưu ý là chúng ta không thể khai báo một hằng số có tên trùng với tên hàm hoặc biến trong cùng một phạm vi.

### 5.2.3 Các kiểu dữ liệu trong JavaScript

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

#### Ví dụ 5.6:

<HTML>

<HEAD>

<TITLE> Datatype Example </TITLE>

<SCRIPT LANGUAGE= "JavaScript">

```
var fruit='apples';
```

```
var numfruit=12;
```

```
numfruit = numfruit + 20;
```

```
var temp ="There are " + numfruit + " "+ fruit + ". ";
```

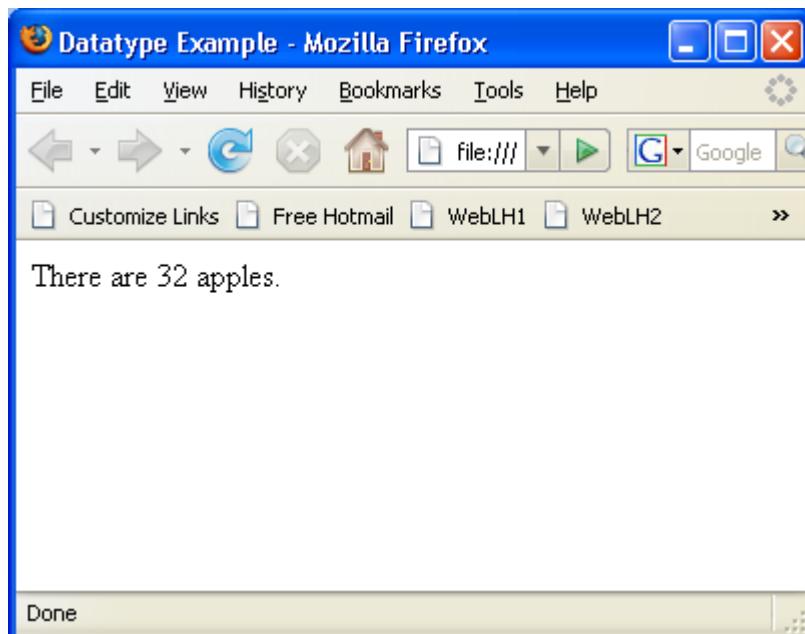
```
document.write(temp);
```

</SCRIPT>

</HEAD>

</HTML>

Các trình duyệt hỗ trợ JavaScript sẽ xử lý chính xác ví dụ trên và đưa ra kết quả dưới đây:



### Hình 5.6: Kết quả của xử lý dữ liệu

Trong ví dụ trên, trình diễn dịch JavaScript sẽ xem biến numfruit có kiểu nguyên khi cộng với 20 và có kiểu chuỗi khi kết hợp với biến temp.

Trong JavaScript có các kiểu dữ liệu như sau: Number (số nguyên hoặc số thực), Logical (hay Boolean), String, Null.

#### 5.2.3.1 Kiểu số nguyên

Các số nguyên có thể được biểu diễn trong hệ thập phân (cơ số 10), hệ thập lục phân (cơ số 16) hoặc hệ bát phân (cơ số 8). Một chữ số nguyên thập phân gồm có một dãy các số mà không có số 0 đứng đầu. Một số 0 đứng đầu trong một chữ số nguyên cho biết nó được biểu diễn trong hệ bát phân, nếu đứng đầu một chữ số nguyên là 0x (hoặc 0X) thì chỉ ra rằng nó được biểu diễn trong hệ thập lục phân.

Các số nguyên hệ thập phân bao gồm các số từ 0 đến 9. Các số nguyên hệ thập lục phân có thể bao gồm các số từ 0 đến 9 và các chữ cái từ a đến f và A đến F. Các số nguyên bát phân chỉ có thể bao gồm các số từ 0 đến 7.

Các chữ số nguyên bát phân không được tách thành và đã bị loại khỏi chuẩn ECMA-262 bản 3. JavaScript vẫn hỗ trợ các chữ số nguyên bát phân để tương thích với các phiên bản trước.

Ví dụ về số nguyên:

42

0xFFFF

-345

#### 5.2.3.2 Kiểu số thực (kiểu số dấu chấm động)

Kiểu số thực có thể có các thành phần sau:

- Phần nguyên thập phân (là một số nguyên thập phân)
- Một dấu chấm thập phân (“.”)
- Phần dư (là một số thập phân khác)
- Phần mũ

Trong đó phần số mũ là một chữ “e” hay “E”, theo sau là một số nguyên, có thể được đánh dấu (được đặt trước bởi dấu “+” hoặc “-”). Một số dấu chấm động phải có ít nhất một con số và một dấu chấm thập phân hoặc “e” (hay “E”).

Ví dụ về số thực:

3.114

-3.1E12

.1e12

2E-12

#### 5.2.3.3 Kiểu Logical (hay Boolean)

Kiểu logic được sử dụng để chỉ hai điều kiện: đúng hoặc sai.

Miền giá trị của kiểu này chỉ có hai giá trị:

- true.
- false.

#### 5.2.3.4 Kiểu chuỗi (String)

Một chuỗi chữ gồm không hoặc nhiều ký tự được đặt trong các dấu nháy kép ("") hoặc nháy đơn (''). Một chuỗi phải được phân định bởi các dấu trích dẫn cùng kiểu, tức là cả hai dấu đều phải là dấu nháy đơn hoặc đều là dấu nháy kép.

Ví dụ về các chuỗi:

“Hello”

‘Error!’

“12345”

Chúng ta có thể gọi bất cứ một phương thức nào của đối tượng String trên một giá trị chuỗi chữ - JavaScript sẽ tự động chuyển đổi chuỗi chữ thành một đối tượng String tạm, gọi phương thức được yêu cầu, rồi sau đó loại bỏ đối tượng String tạm.

Khi dùng chuỗi, ngoài các ký tự thông thường, ta cũng có thể chèn các ký tự đặc biệt vào chuỗi đó. Các ký tự đặc biệt sẽ thực hiện một công việc cụ thể nào đó.

Ví dụ: “one line \n another line”

Trong ví dụ trên, dấu “\” kết hợp với ký tự “n” sẽ mang ý nghĩa là sang dòng. Như vậy khi thực hiện câu lệnh trên thì kết quả sẽ hiển thị là:

one line

another line

Dưới đây là các ký tự đặc biệt và ý nghĩa của chúng trong các chuỗi JavaScript:

Ký tự	Ý nghĩa
\b	Phím lùi (Backspace)
\f	Sang trang mới (Form feed)
\n	Sang dòng mới (new line)
\r	Đưa con trỏ về đầu dòng hiện tại (Carriage return)
\t	Cách một khoảng Tab (Tab)

Ngoài ra, chúng ta cũng có thể chèn một số ký tự đặc biệt 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 chúng ta muốn hiển thị các ký tự ‘, “ hay \ trong chuỗi thì sẽ phải đặt dấu backslash ở phía trước, đó là '\', \" và \\.

#### 5.2.3.5 Kiểu null

Kiểu null chỉ có duy nhất một giá trị: null. Null mang ý nghĩa là không có dữ liệu, nó thực hiện chức năng là giữ chỗ trong một biến với ý nghĩa là ở đó không có hữu dụng gì.

Số 0 hay một xâu rỗng và null là các giá trị khác nhau.

### 5.3 Câu hỏi và bài tập

1. Mã JavaScript thường được nhúng vào tài liệu HTML bằng cách dùng thẻ \_\_\_\_\_.
2. JavaScript có phân biệt giữa chữ hoa và chữ thường hay không? \_\_\_\_\_ (có/không)
3. Nếu trình duyệt có hỗ trợ JavaScript thì đoạn mã trong cặp thẻ <NOSCRIPT> và </NOSCRIPT> có được thực hiện hay không? \_\_\_\_\_ (có/không)
4. Biến có thể được sử dụng mọi nơi trong ứng dụng hiện thời được gọi là \_\_\_\_\_.
5. Nếu ta khai báo một biến bên trong một hàm, biến đó được gọi là \_\_\_\_\_, bởi vì nó chỉ được sử dụng bên trong hàm đó.
6. JavaScript là một ngôn ngữ kịch bản dựa trên đối tượng chỉ để phát triển các ứng dụng Internet trên máy client? \_\_\_\_\_ (Đúng/Sai)
7. Câu lệnh nào để hiển thị chuỗi sau: He said “Hello!”
8. Dùng JavaScript hiển thị một câu thông báo “Xin chào các bạn!”
9. Cũng thực hiện câu 8 với yêu cầu nhúng file JavaScript vào file HTML  
(Gợi ý: Viết hai file, file cau9.js và cau9.html )

## CHƯƠNG 6

# TOÁN TỬ VÀ BIỂU THỨC TRONG JAVASCRIPT

## 6.1 Các toán tử trong JavaScript

### 6.1.1 Các toán tử thông dụng

Các toán tử được sử dụng để thực hiện tính toán trên dữ liệu. Dữ liệu ở đây có thể là một hoặc nhiều biến hoặc giá trị (toán hạng) và trả về một giá trị mới. 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. Kết quả trả về có thể là một giá trị kiểu số, kiểu chuỗi hay kiểu logic. JavaScript cho phép sử dụng các toán tử thông dụng sau:

- *Toán tử gán*
- *Toán tử số học*
- *Toán tử so sánh*
- *Toán tử logic*
- *Toán tử thao tác trên bit*
- *Toán tử chuỗi*

#### 6.1.1.1 Toán tử gán

Toán tử gán (dấu `=`) dùng để thực hiện việc gán giá trị của toán hạng bên phải cho toán hạng bên trái. Ví dụ `x = y` nghĩa là lấy giá trị của `y` gán cho `x`.

Bên cạnh đó, JavaScript còn hỗ trợ một số kiểu toán tử rút gọn với ý nghĩa được trình bày trong bảng sau:

**Bảng 6.1: Các toán tử gán trong JavaScript**

Kiểu gán thông thường	Kiểu gán rút gọn
<code>x = x + y</code>	<code>x += y</code>
<code>x = x - y</code>	<code>x -= y</code>
<code>x = x * y</code>	<code>x *= y</code>
<code>x = x / y</code>	<code>x /= y</code>
<code>x = x % y</code>	<code>x %= y</code>
<code>x = x &lt;&lt; y</code>	<code>x &lt;&lt;= y</code>
<code>x = x &gt;&gt; y</code>	<code>x &gt;&gt;= y</code>
<code>x = x &amp; y</code>	<code>x &amp;= y</code>
<code>x = x ^ y</code>	<code>x ^= y</code>
<code>x = x   y</code>	<code>x  = y</code>

### 6.1.1.2 Toán tử số học

Các toán tử số học đòi hỏi các toán hạng là các giá trị số (các chữ hoặc các biến) và trả về một giá trị số duy nhất. Các toán tử số học tiêu chuẩn là cộng (+), trừ (-), nhân (\*) và chia (/). Các toán tử này làm việc giống như trong hầu hết các ngôn ngữ lập trình khác, ngoại trừ toán tử chia (/) trong JavaScript trả về phép chia dấu chấm động, tức là phép chia không cắt xén kết quả trả về như nó thực hiện trong các ngôn ngữ khác (như C hoặc Java).

Ví dụ:  $1/2$  sẽ trả về kết quả là  $0.5$  trong JavaScript, nhưng trong C hoặc Java thì sẽ trả về kết  $0$ .

Ngoài ra, JavaScript còn cung cấp một số toán tử số học khác như toán tử %, ++, -- và -.

Các toán tử số học trong JavaScript và ý nghĩa của chúng được trình bày trong bảng sau:

**Bảng 6.2: Các toán tử số học trong JavaScript**

Toán tử	Mô tả	Ví dụ
+	Phép cộng	$a = 5+6 // a = 11$
-	Phép trừ	$a = 7- 2 // a = 5$
*	Phép nhân	$a = 5*6 // a = 30$
/	Phép chia	$a = 10/5 // a = 2$
%	Phép chia lấy phần dư. Kết quả trả về là số dư kiểu interger trong phép chia hai toán hạng.	$a = 10%3 // a = 1$
++	Toán tử này nhận một toán hạng, và sẽ tăng giá trị của toán hạng này lên một đơn vị. Giá trị được trả về sẽ tùy thuộc vào toán tử ++ nằm trước hay nằm sau toán hạng (xem phần chú ý phía dưới bảng).	$x = 5$ $a = ++x // a=6, x=6$ $b = x++ // b=5, x=6$
--	Tương tự như ++, toán tử -- nhận một toán hạng, và sẽ giảm giá trị của toán hạng này xuống một đơn vị. Giá trị được trả về sẽ tùy thuộc vào toán tử -- nằm trước hay nằm sau toán hạng (xem phần chú ý phía dưới bảng).	$x = 5$ $a = --x // a=4, x=4$ $b = x-- // b=5, x=4$
-	Toán tử này sẽ trả về giá trị đối (phủ định) của toán hạng.	$a = 5 \text{ thì } -a = -5$

**Chú ý:**

- Trong các toán tử số học trên, các toán tử `+`, `-`, `*`, `/`, `%` được gọi là các toán tử số học hai ngôi, và các toán tử `++`, `--`, `-` được gọi là toán tử số học một ngôi, có nghĩa là nó chỉ có tác dụng với một toán hạng đi kèm.
- Nếu toán tử `++` hay `--` kết hợp với một toán tử khác, ví dụ như kết hợp với toán tử gán, thì kết quả trả về sẽ là khác nhau phụ thuộc vào vị trí xuất hiện trước hay sau của `++` hay `--` với tên biến, có nghĩa là `y = ++x` sẽ cho ra kết quả khác với `y = x++`. Nếu `++` hay `--` đứng trước `x` thì `x` sẽ được tăng hoặc giảm một đơn vị trước khi giá trị `x` được gán cho `y`. Nếu `++` hay `--` đứng sau `x` thì giá trị của `x` sẽ được gán cho `y` trước khi nó được tăng hay giảm. (Xem ví dụ trong bảng trên)

**6.1.1.3 Toán tử so sánh**

Các toán tử so sánh sẽ so sánh các toán hạng của nó và trả về một giá trị logic trên cơ sở phép so sánh có đúng hay không. Toán hạng có thể là các số, chuỗi, logic hay đối tượng. Các chuỗi được so sánh trên cơ sở thứ tự từ điển tiêu chuẩn, sử dụng các giá trị Unicode. Kết quả trả về của toán tử này là một giá trị `true` (đúng) hoặc `false` (sai).

Bảng sau mô tả các toán tử so sánh và ý nghĩa của chúng trong JavaScript:

**Bảng 6.3: Các toán tử so sánh trong JavaScript**

Toán tử so sánh	Mô tả	Ví dụ (trả về kết quả <code>true</code> )
<code>==</code> (Bằng)	Trả về giá trị <code>true</code> nếu các toán hạng bằng nhau. Nếu hai toán hạng không cùng kiểu, JavaScript sẽ thử đổi các toán hạng thành một kiểu tương ứng để so sánh.	<code>3 == var1</code> <code>“3” == var1</code> <code>3 = ‘3’</code>
<code>!=</code> (Không bằng)	Trả về giá trị <code>true</code> nếu các toán hạng không bằng nhau. Nếu hai toán hạng không cùng kiểu, JavaScript sẽ thử đổi các toán hạng thành một kiểu tương ứng để so sánh.	<code>var1 !=4</code> <code>var2 !=”3”</code>
<code>====</code> (Bằng tuyệt đối)	Trả về giá trị <code>true</code> nếu hai toán hạng bằng nhau và có cùng kiểu.	<code>3====var1</code>
<code>!==</code> (Không bằng tuyệt đối)	Trả về giá trị <code>true</code> nếu các toán hạng không bằng nhau và/hoặc không cùng kiểu.	<code>var1!==”3”</code> <code>3!==’3’</code>
<code>&gt;</code>	Trả về giá trị <code>true</code> nếu toán hạng	<code>var2&gt;var1</code>

(Lớn hơn)	bên trái lớn hơn toán hạng bên phải.	
>= (Lớn hơn hoặc bằng)	Trả về giá trị true nếu toán hạng bên trái lớn hơn hoặc bằng toán hạng bên phải.	var2>=var1 var1>=3
< (Nhỏ hơn)	Trả về giá trị true nếu toán hạng bên trái nhỏ hơn toán hạng bên phải.	var1<var2
<= (Nhỏ hơn hoặc bằng)	Trả về giá trị true nếu toán hạng bên trái nhỏ hơn hoặc bằng toán hạng bên phải.	var1<=var2 var2<=5

Trong các ví dụ ở bảng trên, ta giả sử rằng var1 được gán giá trị là 3 và var2 được gán giá trị là 4.

#### 6.1.1.4 Toán tử logic

Các toán tử logic được sử dụng tiêu biểu cho các giá trị Boolean, khi các toán hạng có giá trị là boolean thì chúng trả về giá trị kiểu boolean. Tuy nhiên, các toán tử **&&** và **||** thực sự trả về giá trị của một trong các toán hạng chỉ định. Như vậy nếu các toán tử này được gán với các giá trị không có kiểu boolean, thì chúng có thể trả về một giá trị không có kiểu boolean. Các toán tử logic được mô tả trong bảng sau:

Bảng 6.4: Các toán tử logic trong JavaScript

Toán tử	Sử dụng	Mô tả
<b>&amp;&amp;</b>	bt1 && bt2	Toán tử logic AND, trả về giá trị là bt1 nếu bt1 có thể được chuyển đổi tương ứng thành false; ngoài ra trả về bt2. Như vậy khi sử dụng nó với các giá trị boolean, <b>&amp;&amp;</b> trả về true nếu cả hai toán hạng đều là true, ngoài ra trả về giá trị false.
<b>  </b>	bt1    bt2	Toán tử logic OR trả về giá trị bt1 nếu bt1 có thể được chuyển đổi tương ứng thành true, ngoài ra trả về bt2. Như vậy, khi sử dụng với các giá trị boolean, <b>  </b> trả về true nếu mỗi toán hạng là true; nếu cả hai toán hạng là false thì trả về giá trị false.
<b>!</b>	<b>!bt</b>	Toán tử logic NOT trả về false nếu toán hạng của nó có thể được chuyển đổi tương ứng thành true; ngoài ra trả về true

Từ những mô tả trên, ta rút ra được bảng chân trị (với các giá trị boolean) sau:

**Bảng 6.5: Bảng chân trị cho các toán tử logic**

<i>bt1</i>	<i>bt2</i>	<i>bt1 &amp;&amp; bt2</i>	<i>bt1    bt2</i>	<i>!bt1</i>
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

Các biểu thức có thể được chuyển đổi tương ứng thành false là các biểu thức được định giá trị là null, 0, chuỗi rỗng ("") hoặc không xác định.

Đối với các giá trị khác (không phải là boolean) ta xét các ví dụ sau:

```
A1 = "cat" && "dog" // trả về giá trị dog
A2 = false && "dog" // trả về giá trị false
A3 = "cat" && false // trả về giá trị false
A4 = "cat" || "dog" // trả về giá trị cat
A5 = false || "dog" // trả về giá trị dog
A6 = "cat" || false // trả về giá trị cat
A7 = !"cat" // trả về giá trị false
```

### **Định giá trị ngay lập tức**

Vì các biểu thức logic được định giá trị từ trái sang phải, nên chúng được kiểm tra để có thể định giá trị ngay lập tức bằng cách sử dụng các luật sau:

- false && anything được định giá trị ngay lập tức là false
- true || anything được định giá trị ngay lập tức là true

Các luật logic đảm bảo rằng việc định giá trị này luôn chính xác. Ở đây lưu ý rằng phần *anything* của các biểu thức trên không cần thiết phải định giá trị, nói cách khác là việc thực hiện các biểu thức trên sẽ không quan tâm tới giá trị của *anything*.

#### **6.1.1.5 Toán tử thao tác trên bit**

Các toán tử thao tác bit xem xét các toán hạng của chúng như một tập hợp 32 bit (các số 0 và số 1), thay vì các số thập phân, thập lục phân và bát phân. Ví dụ, số thập phân 9 có chuỗi nhị phân mô tả là 1001. Các toán tử thao tác bit thực hiện các hoạt động của chúng trên các chuỗi bit mô tả như vậy, nhưng chúng trả về các giá trị số tiêu chuẩn của JavaScript.

Bảng sau đây mô tả các toán tử thao tác trên bit của JavaScript:

**Bảng 6.6: Các toán tử thao tác trên bit**

Toán tử	Sử dụng	Mô tả
AND	a & b	Trả về giá trị 1 trong mỗi vị trí bit, nếu các bit tương ứng của hai toán hạng đều có giá trị là 1.
OR	a   b	Trả về giá trị 1 trong mỗi vị trí bit, nếu các bit tương ứng của một hoặc hai toán hạng có giá trị là 1.
XOR	a ^ b	Trả về giá trị 1 trong mỗi vị trí bit, nếu các bit tương ứng của cả hai toán hạng không cùng bằng 1.
NOT	~a	Đảo các bit của toán hạng
Dịch trái	a << b	Dịch chuỗi bit mô tả giá trị a sang trái b bit, thêm các số 0 vào bên phải.
Dịch phải nhân dấu	a >> b	Dịch chuỗi bit mô tả giá trị a sang phải b bit, loại bỏ đi các bit bị đẩy ra ngoài.
Dịch phải điền giá trị 0	a >>> b	Dịch chuỗi bit mô tả giá trị a sang phải b bit, loại bỏ đi các bit bị đẩy ra ngoài, điền các số 0 vào bên trái

Ta chia các toán tử thao tác trên bit ra làm hai loại: Các toán tử logic thao tác bit ( $\&$ ,  $|$ ,  $^$ ,  $\sim$ ) và các toán tử dịch của thao tác bit ( $<<$ ,  $>>$  và  $>>>$ ).

### Các toán tử logic thao tác bit

Các toán tử logic thao tác bit làm việc như sau:

- Các toán hạng được đổi thành các số nguyên 32 bit và được biểu diễn bằng một chuỗi bit (0 và 1)
- Mỗi bit trong toán hạng thứ nhất được ghép với bit tương ứng trong toán hạng thứ hai: bit đầu với bit đầu, bit thứ hai với bit thứ hai...
- Toán tử được áp dụng cho mỗi cặp bit và giá trị trả về được xây dựng từ kết quả của việc áp dụng toán tử cho các cặp bit đó.

Ví dụ, số 9 được mô tả bằng dãy bit 1001, và số 15 được mô tả bằng dãy bit 1111, khi các toán tử thao tác bit được áp dụng cho các giá trị này thì kết quả trả về như sau:

- $15 \& 9 = 9$  ( $1111 \& 1001 = 1001$ )
- $15 | 9 = 15$  ( $1111 | 1001 = 1111$ )
- $15 ^ 9 = 6$  ( $1111 ^ 1001 = 0110$ )

### Các toán tử dịch của thao tác bit

Các toán tử dịch bit cần hai toán hạng, toán hạng thứ nhất là số được dịch, và toán hạng thứ hai chỉ ra số vị trí mà toán hạng thứ nhất được dịch chuyển. Hướng của toán tử dịch chuyển bit được chỉ ra bởi toán tử sử dụng.

Các toán tử dịch chuyển sẽ đổi các toán hạng thứ nhất thành số nguyên 32 bit, dịch theo hướng của toán tử yêu cầu sang số bước dịch được chỉ ra ở toán hạng thứ hai, và sau đó trả về kết quả cùng kiểu với toán hạng thứ nhất.

Các toán tử dịch chuyển và ý nghĩa của chúng được liệt kê và phân tích trong bảng sau:

**Bảng 2.7: Các toán tử dịch chuyển bit**

Toán tử	Mô tả	Ví dụ
<code>&lt;&lt;</code> (Dịch trái)	Toán tử này dịch chuyển toán hạng đầu tiên sang trái một số bit xác định bởi toán hạng thứ hai. Các bit đẩy sang trái bị loại bỏ, điền các bit 0 vào bên phải	$9 << 2 = 36$ (vì 9 được viết thành 1001, dịch sang trái hai bit thành 100100, tức là bằng 36 )
<code>&gt;&gt;</code> (Dịch phải duy trì dấu)	Toán tử này dịch chuyển toán hạng đầu tiên sang phải một số bit xác định bởi toán hạng thứ hai. Các bit đẩy sang phải sẽ bị loại bỏ. Sao chép các bit bên trái nhất được dịch vào từ bên trái. Chính vì vậy mà dấu của toán hạng thứ nhất được duy trì sau phép dịch này.	<ul style="list-style-type: none"> <li><math>9 &gt;&gt; 2 = 2</math> (vì 1001 dịch sang phải hai bit thành 10, tức là 2)</li> <li><math>-9 &gt;&gt; 2 = -3</math> (bởi vì dấu được duy trì)</li> </ul>
<code>&gt;&gt;&gt;</code> (Dịch phải điền 0)	Toán tử này dịch toán hạng thứ nhất sang phải một số bit xác định. Các bit đẩy ra ngoài bị loại bỏ. Các bit 0 được điền vào bên trái.  Đối với các số không âm, dịch phải điền giá trị 0 và dịch phải duy trì dấu có cùng kết quả.	$19 >>> 2 = 4$ (vì 19 được chuyển thành 10011, dịch phải 2 bit thành 100, tức là 4)

#### 6.1.1.6 Toán tử chuỗi

Các toán tử so sánh có thể được sử dụng cho các giá trị chuỗi, toán tử nối (+) nối hai giá trị chuỗi với nhau, và trả về một chuỗi mới là sự hợp nhất của hai chuỗi toán hạng.

Ví dụ: “my” + “string” trả về chuỗi “mystring”

Toán tử gán viết gọn `+=` có thể cũng được sử dụng để nối các chuỗi. Ví dụ, nếu biến mystring có giá trị là “alpha”, thì biểu thức mystring `+= “bet”` có giá trị là “alphabet” và gán giá trị này cho biến mystring.

### **6.1.2 Một số toán tử khác**

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:

#### **6.1.2.1 Toán tử điều kiện**

Toán tử điều kiện là một toán tử của JavaScript cần ba toán hạng . Toán tử có thể có một trong hai giá trị tùy thuộc vào điều kiện.

Cú pháp:

condition ? val1 : val2

Nếu điều kiện là true, thì toán tử có giá trị là val1. Nếu không thì nó có giá trị là val2.

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

Câu lệnh này gán giá trị "adult" cho biến status nếu age >= 18. Ngược lại, nó gán cho biến này giá trị "minor"

#### **6.1.2.2 Toán tử dấu phẩy**

Toán tử dấu phẩy (,) định giá trị cho cả hai toán hạng của nó và trả về giá trị của toán hạng thứ hai. Toán tử này được sử dụng chủ yếu trong vòng lặp for, để cho phép nhiều biến được gán giá trị ban đầu hoặc nhiều biến được cập nhật lại giá trị thông qua mỗi bước lặp.

Ví dụ: Xét bài toán tính tổng các số từ 1 đến 10 dùng vòng lặp for. Ta sẽ khởi tạo giá trị ban đầu cho hai biến, biến dem=1 và biến tong=0, lúc này ta sẽ sử dụng toán tử dấu phẩy.

```
for (var dem=1,tong=0;dem<=10;dem++)
    tong += dem;
```

#### **6.1.2.3 Toán tử new**

Chúng ta có thể sử dụng toán tử *new* để tạo ra một thê hiện (instance) của kiểu đối tượng được định nghĩa bởi người sử dụng hoặc một kiểu đối tượng được định nghĩa trước như Array, Boolean, Date, Function, Image, Number, Object, Option, RegExp, hoặc String. Trên server ta cũng có thể sử dụng nó với DbPool, Lock, File, hoặc SendMail.

Cú pháp của toán tử này như sau:

objectName= new objectType (param1 [,param2]... [,paramN])

#### **6.1.2.4 Toán tử typeof**

Toán tử typeof trả về chuỗi cho biết tên kiểu dữ liệu của toán hạng. Toán hạng có thể là một chuỗi, một biến, từ khóa, hoặc đối tượng.

Cú pháp: typeof (operand)

Trong đó dấu ngoặc đơn không bắt buộc.

Ví dụ: Giả sử ta có các biến:

```
var x = 5;
```

```
var shape = "round";
```

Toán tử typeof trả về các kết quả sau cho các biến này:

typeof x is number

typeof shape is string.

Với từ khóa true và null, toán tử typeof trả về các kết quả sau:

typeof true is boolean

typeof null is object

Đoạn mã sau minh họa cho các trường hợp trên:

### Ví dụ 6.1:

<HTML>

<HEAD>

<SCRIPT>

```
var x=5;
```

```
var shape= "round"
```

```
document.write(typeof (x));
```

```
document.write("<br>" + typeof (shape));
```

```
document.write("<br>" + typeof (true));
```

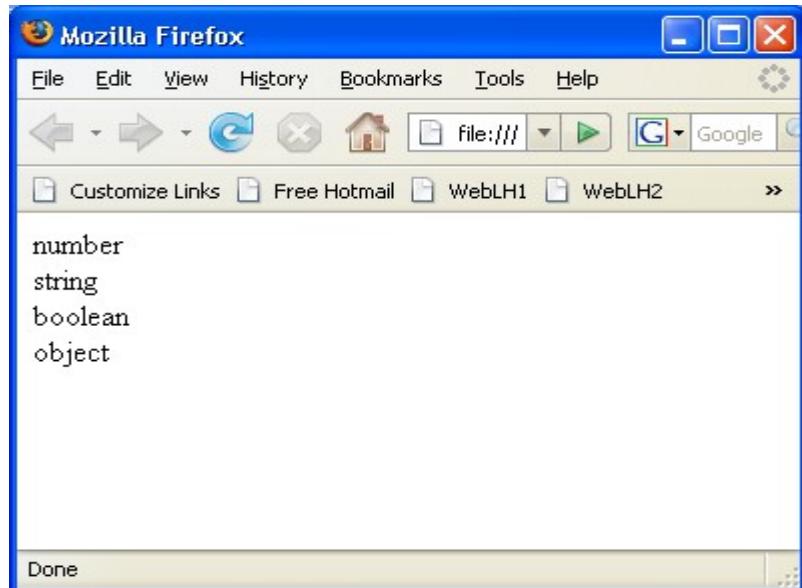
```
document.write("<br>" + typeof (null));
```

</SCRIPT>

</HEAD>

</HTML>

### Kết quả:



**Hình 6.1: Toán tử typeof**

### 6.1.2.5 Toán tử this

Câu lệnh ‘this’ 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.

Cú pháp:

this [property]

Nếu 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 6.2:

<HTML>

<HEAD>

<script>

function dispname (name) {

alert("welcome, "+name);

}

</script>

</HEAD>

<form>

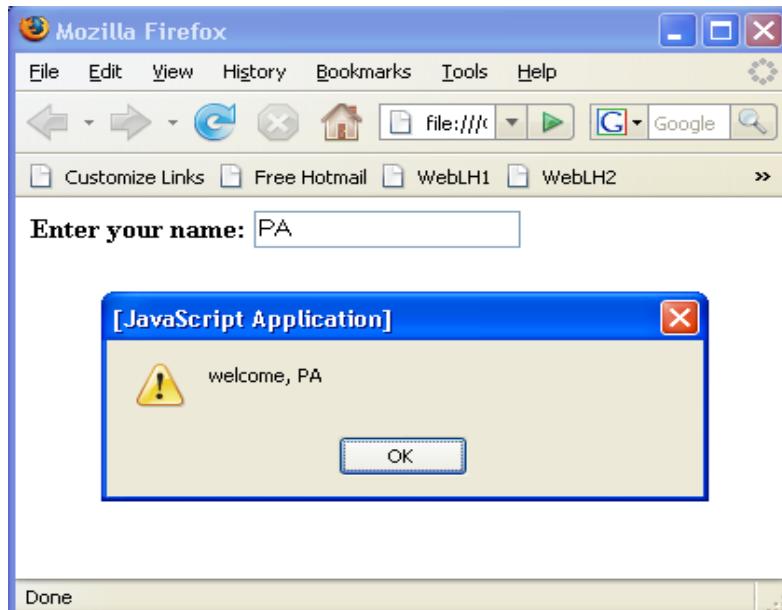
<B> Enter your name: </B>

<INPUT TYPE = "text" NAME = "text1" SIZE = 20  
onChange="dispname(this.form.text1.value)">

</form>

</HTML>

**Kết quả:**



**Hình 6.2: Toán tử this**

### 6.1.3 Thứ tự 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 các 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 muốn thay đổi trật tự thực hiện của các toán tử, chúng ta phải sử dụng các cặp dấu ngoặc đơn ( ).

Bảng dưới đây liệt kê độ ưu tiên của các toán tử từ thấp đến cao:

**Bảng 6.8: Độ ưu tiên của các toán tử**

Kiểu toán tử	Các toán tử
Dấu phẩy	
Phép gán	= += -= *= /= %= <=>= >>= &= ^=  =
Điều kiện	?:
Toán tử logic OR	
Toán tử logic AND	&&
Toán tử thao tác bit OR	
Toán tử thao tác bit XOR	^
Toán tử thao tác bit AND	&
So sánh bằng	== != === !==
Quan hệ	< <= > =
Toán tử dịch chuyển trên bit	<<>>>>
Cộng/Trừ	+ -
Nhân/Chia	* / %
Phủ định/Tăng	! ~ - ++ --

## 6.2 Các biểu thức trong JavaScript

Để sử dụng các biến hiệu quả, ta phải có thể thao tác và tính toán chúng mỗi khi cần thiết. Chúng ta thực hiện được điều này nhờ sử dụng các biểu thức (expression).

Một biểu thức là một tập hợp hợp lệ gồm các hằng, các biến và các toán tử để tính toán và trả về một giá trị đơn. Giá trị này có thể là một số, một chuỗi hay bất kỳ một giá trị logic nào đó.

Có hai kiểu biểu thức: gán một giá trị cho một biến, và đơn giản là chỉ có một giá trị. Ví dụ, biểu thức `x = 7` là biểu thức mà biến `x` được gán giá trị là 7. Biểu thức này tự nó định giá trị là 7. Các biểu thức như vậy sử dụng các toán tử gán. Một ví dụ khác, ta có biểu thức `3 + 4` định giá trị là 7, nó không thực hiện phép gán. Các toán tử được sử dụng trong các biểu thức như vậy đơn giản được tham khảo tới như là các operator (toán tử).

JavaScript có các biểu thức sau đây:

- Số học: định giá trị là một số, ví dụ 3.14159
- Chuỗi: định giá trị là một chuỗi ký tự, ví dụ “Fred” hoặc “1234”
- Logic: định giá trị là true hoặc false
- Đôi tượng: định giá trị là một đối tượng

### 6.2.1 Biểu thức regular

Biểu thức regular là các mẫu được sử dụng để so khớp các liên kết ký tự trong các chuỗi. Với biểu thức regular, ta có thể tìm kiếm theo mẫu trong các chuỗi ký tự do người dùng nhập vào. Ví dụ, ta tạo một mẫu tìm kiếm gồm từ “cat” và sẽ tìm kiếm tất cả các xuất hiện của từ này trong một xâu nào đó. Trong JavaScript, các biểu thức regular cũng là các đối tượng.

Mẫu tìm kiếm của biểu thức regular có thể bao gồm:

#### Sử dụng các mẫu đơn giản

Các mẫu đơn giản được xây dựng từ các ký tự mà bạn muốn so khớp trực tiếp. Ví dụ, mẫu /abc/ so khớp sự liên kết ký tự trong các chuỗi chỉ khi các ký tự ‘abc’ đi liền nhau theo thứ tự một cách chính xác. Như vậy việc so khớp sẽ thành công trong chuỗi “Hi, do you know your abc’s?”, và không có sự so khớp trong chuỗi “Grab crab” bởi vì nó không chứa chuỗi con ‘abc’.

#### Sử dụng các ký tự đặc biệt

Khi tìm kiếm một trùng khớp yêu cầu nhiều điều hơn việc so khớp trực tiếp, như việc tìm một hoặc nhiều ký tự b chẵng hạn, hoặc tìm các khoảng trắng, mẫu có chứa các ký tự đặc biệt. Ví dụ, mẫu /ab\*c/ so khớp với bất cứ liên kết ký tự nào, trong đó ký tự duy nhất ‘a’ được theo sau bởi không hoặc nhiều ký tự ‘b’ (\* có nghĩa là không hoặc nhiều ký tự có sẵn) và ngay lập tức được theo sau bởi ký tự ‘c’. Trong chuỗi “cbbabbbbcdabc” mẫu so khớp là chuỗi con ‘abbbbcd’.

Bảng sau liệt kê và mô tả một số ký tự đặc biệt có thể được sử dụng trong các biểu thức regular:

**Bảng 6.9: Các ký tự đặc biệt trong các biểu thức regular**

Ký tự	Ý nghĩa	Ví dụ
\	<ul style="list-style-type: none"> <li>• Có một trong các ý nghĩa sau:           <ul style="list-style-type: none"> <li>- Với các ký tự được xem xét theo từng chữ, chỉ ra ký tự tiếp theo là ký tự đặc biệt và không được thông dịch theo từng chữ.</li> <li>- Với các ký tự được xem xét một cách đặc biệt, chỉ ra ký tự tiếp theo không phải là ký tự đặc biệt</li> </ul> </li> </ul>	<ul style="list-style-type: none"> <li>- Ví dụ, /b/ so khớp với ký tự ‘b’. Bằng cách đặt một dấu gạch chéo ngược trước b như thế này /b/, ký tự trở nên đặc biệt có ý nghĩa so khớp với một từ phân định.</li> <li>- Ví dụ, * là một ký tự đặc biệt có nghĩa là không hoặc nhiều ký tự của chuỗi có sẵn sẽ được so khớp; ví dụ, /a*/ có nghĩa là so khớp với không hoặc nhiều</li> </ul>

	và sẽ được thông dịch theo từng chữ.	ký tự ‘a’. Để so khớp với ký tự ‘*’ theo từng chữ, hãy đặt trước nó một dấu gạch chéo ngược; ví dụ, /a\*/ so khớp với ‘a*’.
•		
•		
^	So khớp với phần đầu của chuỗi nhập. Nếu cờ đa dòng được thiết lập là true, thì sẽ so khớp ngay lập tức sau khi có một ký tự xuống dòng.	/^A/ không so khớp với ký tự ‘A’ trong chuỗi “an A”, nhưng so khớp với ký tự ‘A’ đầu tiên trong chuỗi “An A”.
\$	So khớp với phần cuối của chuỗi nhập. Nếu cờ đa dòng được thiết lập là true, thì sẽ so khớp ngay lập tức trước khi có một ký tự xuống dòng	/t\$/ không so khớp với ‘t’ trong chuỗi “eater”, nhưng so khớp nó trong chuỗi “eat”
*	So khớp với ký tự đi trước không hoặc nhiều lần.	/bo*/ so khớp với chuỗi ‘boooo’ trong chuỗi “A ghost booooed” và ký tự ‘b’ trong chuỗi “A bird warbled”, nhưng không phải trong chuỗi “A goat grunted”.
+	So khớp với ký tự đi trước một hoặc nhiều lần. Tương đương với {1,}.	/a+/ so khớp với ‘a’ trong chuỗi “candy” và tất cả các ký tự ‘a’ trong chuỗi “caaaandy”.
?	So khớp với ký tự đi trước không hoặc một lần. Nếu được sử dụng ngay sau các ký tự *, +, ? hoặc {}, thì tạo thành ký tự non-greedy (so khớp với số lần nhỏ nhất), trái lại là greedy (so khớp số lần lớn nhất).	/e?le?/ so khớp với ‘el’ trong “angel” và ‘le’ trong “angle”.
.	(Đáu chấm thập phân) so khớp với ký tự duy nhất nào đó ngoại trừ ký tự dòng mới.	
(x)	So khớp với chuỗi ‘x’ và nhớ sự so khớp này. Các dấu ngoặc đơn này được gọi là các dấu ngoặc đơn nhóm.	/(foo)/ so khớp với ‘foo’ trong chuỗi “foo bar” và ghi nhớ ‘foo’. Chuỗi con được so khớp có thể gọi lại từ các phần tử [1], ..., [n] của mảng kết quả.
x y	So khớp với ‘x’ hoặc ‘y’.	/green red/ so khớp với ‘green’ trong “green apple” và ‘red’ trong “red apple”.

{n,m}	Với n, m là các số nguyên dương. So khớp với ít nhất n và nhiều nhất m sự kiện của ký tự đi trước.	/a{1,3}/ không so khớp với cái gì trong chuỗi “cndy” cả, nhưng so khớp với ký tự ‘a’ trong “candy”, hai ký tự ‘a’ đầu tiên trong “caandy” và ba ký tự ‘a’ đầu tiên trong “aaaaaaaaandy”. Nhớ rằng khi so khớp “aaaaaaaaandy”, chuỗi so khớp là “aaa”, thậm chí khi chuỗi nguyên thủy có nhiều ‘a’ trong nó.
[xyz]	Một tập ký tự. So khớp với một trong các ký tự bao hàm trong tập ký tự. Chúng ta có thể xác định một miền các ký tự bằng cách sử dụng một dấu nối (-).	[abcd] tương đương với [a-d]. Chúng so khớp với ‘b’ trong chuỗi “brisket” và ‘c’ trong chuỗi “ache”.
[^xyz]	Một tập ký tự bù của [^xyz]. Đó là, nó so khớp với bất cứ ký tự nào mà không nằm trong dấu ngoặc vuông. Chúng ta có thể xác định một miền các ký tự bằng cách sử dụng một dấu nối (-).	[^abc] tương tự như [^a-c]. Ngay đầu tiên chúng so khớp với ký tự ‘r’ trong “brisket” và ‘h’ trong “chop”.
\d	So khớp với một ký tự số. Tương đương với [0-9].	\d/ hoặc /[0-9]/ so khớp với ‘2’ trong “B2 is the suit number”.
\s	So khớp với một ký tự khoảng trắng duy nhất, bao gồm các ký tự khoảng trắng, tab, ký tự sang trang, ký tự chuyển dòng. Tương đương với [\f\n\r\t\v].	\s\w*/ so khớp với ‘bar’ trong “foo bar”.
\t	So khớp với phím tab	
\w	So khớp với một ký tự nào đó trong bảng chữ cái, kể cả ký tự gạch dưới (_). Tương đương với [A-Za-z0-9 ]	\w/ so khớp với ‘a’ trong “apple”, ‘5’ trong “\$5.28” và ‘3’ trong “3D”.
\n	Với n là một số nguyên dương. Một tham khảo ngược đến chuỗi con cuối cùng so khớp n dấu ngoặc đơn mở trong biểu thức regular (tổng số các dấu ngoặc đơn trái).	/apple(,)\sorange\1/ so khớp với ‘apple, orange,’ trong “apple, orange, cherry, peach”

### 6.2.2 Tạo ra một biểu thức regular

Một biểu thức regular là một mẫu tìm kiếm để tìm kiếm dữ liệu cùng dạng. JavaScript xem một biểu thức regular như một đối tượng. Vì vậy, chúng ta phải tạo

một biểu thức regular trước khi sử dụng chúng. Có thể tạo ra một biểu thức regular bằng một trong hai cách sau:

#### 6.2.2.1 Khởi tạo đối tượng (Object initializer).

Cách này trước đây được gọi là tạo đối tượng bằng cách sử dụng các ký hiệu nguyên dạng, sau đó nó được chuyển thành *khởi tạo đối tượng* để giống với thuật ngữ của C++. Nếu ta muốn tạo ra một thể hiện của một đối tượng, ta phải dùng một *Object initializer*.

Cú pháp của việc khởi tạo đối tượng như sau:

objectname = {expression}

Trong đó *objectname* là tên của đối tượng mới, *expression* là một khuôn mẫu để tạo đối tượng.

Chẳng hạn:

`re = /xy+z/`

Trong định nghĩa trên, một biểu thức regular “`xy+z`” được tạo và gán cho đối tượng *re*. Bây giờ chúng ta có thể dùng đối tượng *re* để tìm kiếm các mẫu theo yêu cầu.

Khi chúng ta khởi tạo đối tượng, biểu thức regular sẽ được dịch khi script được đánh giá. Nếu biểu thức regular không thay đổi, thì sử dụng khởi tạo đối tượng hiệu quả hơn.

#### 6.2.2.2 Gọi hàm khởi tạo của đối tượng RegExp

JavaScript cung cấp đối tượng biểu thức định nghĩa trước, đó là *RegExp*. Một hàm khởi tạo được dùng để tạo một kiểu đối tượng và định nghĩa các thuộc tính của đối tượng. Ví dụ, chúng ta có thể tạo một đối tượng gọi là “*employee*”. Các thuộc tính của đối tượng là *empID*, *join\_dt*, *salary*.

```
function employee (empID, join_dt, salary)
{
    this.empID = empID
    this.join_dt = join_dt
    this.salary = salary
}
```

Sau khi hàm được tạo, chúng ta phải dùng hàm để tạo một thể hiện của đối tượng bằng toán tử *new*. Ví dụ:

`employee1=new employee("123", "17/11/07", 2500)`

Khi chúng ta dùng hàm khởi tạo, biểu thức được dịch trong thời gian thực thi. Nếu biểu thức regular thay đổi hoặc nếu nó phụ thuộc vào dữ liệu nhập vào từ người dùng, sử dụng hàm khởi tạo là hợp lý nhất.

Ví dụ, hàm *getdetails()* tìm kiếm một mẫu dữ liệu được nhập từ người dùng. Dữ liệu nhập bởi người dùng rất đa dạng. Chọn mẫu tìm kiếm là `(\w+)\s(\d+)`. Có nghĩa là, một hoặc nhiều ký tự bất kỳ xuất hiện theo sau một ký tự trắng hoặc xuất hiện bất kỳ

một ký tự số nào. Dấu cộng (+) chỉ ra một hoặc nhiều ký tự xuất hiện. Dấu sao (\*) chỉ ra 0 hoặc nhiều ký tự xuất hiện.

```
function getdetails()
{
    re = /(\w+)\s(\d+)/
    re.exec();
    window.alert(RegExp.$1+“, your age is ” + RegExp.$2);
}
```

### 6.2.3 Sử dụng biểu thức regular

Các biểu thức regular được sử dụng với các phương thức test và exec của đối tượng RegExp và các phương thức match, replace, search, và split của đối tượng String.

Các phương thức này được mô tả trong bảng sau:

**Bảng 6.10: Các phương thức sử dụng các biểu thức regular**

Phương thức	Mô tả
Exec	Tìm kiếm một mẫu tương xứng trong một chuỗi. Nó trả về một mảng thông tin.
Test	Kiểm tra tương xứng trong một chuỗi. Trả về giá trị đúng hoặc sai.
Match	Tìm kiếm tương xứng trong một chuỗi. Trả về một mảng thông tin hoặc giá trị null nếu sai.
Search	Kiểm tra sự tương xứng trong một chuỗi. Trả về giá trị chỉ số của tương xứng nếu tồn tại, -1 nếu bị sai.
Replace	Tìm kiếm sự tương xứng trong một chuỗi, và thay thế chuỗi con tìm kiếm tương xứng bằng một chuỗi con thay thế khác.
Split	Dùng để tách một chuỗi thành một mảng các chuỗi con.

Để dùng một phương thức, chúng ta phải xác định đối tượng được sử dụng.

Cú pháp là:

```
objectname.method = function_name
```

Sau đó chúng ta có gọi phương thức trong ngữ cảnh của đối tượng.

Cú pháp là:

```
Objectname.methodname(parameters)
```

Chúng ta có thể dùng các cờ với biểu thức regular. Hai cờ “g” và “i” được chọn tùy ý, có thể dùng riêng hoặc dùng cả hai cờ. Cờ “g” được dùng để chỉ dẫn tìm kiếm toàn cục. Cờ “i” dùng để chỉ dẫn tìm kiếm có phân biệt chữ hoa và chữ thường.

Chẳng hạn: re = /\w+\s/g ; //use a global search

**Ví dụ 6.3:**

Đoạn mã dưới đây dùng để kiểm tra phương thức tìm kiếm một mẫu trong chuỗi. Nếu mẫu được tìm thấy, giá trị trả về là “true” và ngược lại thì trả về “false”.

&lt;HTML&gt;

&lt;HEAD&gt;

&lt;SCRIPT&gt;

re = /Time/

str = re.test ("Time and Tide wait for none");

window.alert (str);

&lt;/SCRIPT&gt;

&lt;/HEAD&gt;

&lt;/HTML&gt;

**Kết quả:****Hình 6.2: Minh họa biểu thức regular****Ví dụ 6.4:**

Đoạn mã sau đây tìm kiếm sự xuất hiện của ký tự x, y, hoặc z.

&lt;HTML&gt;

&lt;HEAD&gt;

&lt;SCRIPT&gt;

re = /[xyz]/

str = re.exec ("It is very coooooold");

window.alert (str);

&lt;/SCRIPT&gt;

&lt;/HEAD&gt;

&lt;/HTML&gt;

**Kết quả:****Hình 6.3: Minh họa biểu thức regular**

### 6.3 Câu hỏi và bài tập

**Câu hỏi:**

1. JavaScript dùng cả toán tử một ngôi và toán tử hai ngôi? \_\_\_\_\_ (Đúng/Sai)
2. Các toán tử một ngôi bao gồm \_\_\_\_\_
3. Kết quả trả về của hai biểu thức  $y = ++x$  và  $y = x++$  có giống nhau không?  
\_\_\_\_\_ (Có/Không)
4. Trong JavaScript, toán tử logic chỉ được sử dụng với các toán hạng mang giá trị boolean \_\_\_\_\_ (Đúng/Sai)
5. Đối với toán tử logic  $&&$ , nếu một toán hạng có giá trị false thì kết quả trả về là \_\_\_\_\_.
6. Đối với các số không âm, dấu phải điền giá trị 0 và dấu phải duy trì dấu có kết quả khác nhau \_\_\_\_\_ (Đúng/Sai)
7. Toán tử điều kiện là một toán tử của JavaScript cần \_\_\_\_\_ toán hạng.
8. Khi có nhiều toán tử trong cùng một biểu thức, \_\_\_\_\_ của toán tử xác định thứ tự thực hiện của các toán tử đó.
9. Một \_\_\_\_\_ là một tập hợp hợp lệ gồm các hằng, các biến và các toán tử để tính toán và trả về một giá trị đơn.
10. Giá trị trả về (hay kết quả trả về) của một biểu thức chỉ có thể là một số \_\_\_\_\_ (Đúng/Sai).
11. Hãy đánh giá các biểu thức sau:
  - a.  $7 + 5$
  - b. "7" + "5"
  - c.  $7 == 7$
  - d.  $7 >= 5$
  - e.  $7 <= 7$
  - f.  $(7 < 5) ? 7 : 5$
  - g.  $(7 >= 5) \&\& (5 > 5)$
  - h.  $(7 >= 5) || (5 > 5)$
12. Đoạn mã sau sẽ in ra kết quả?
 

```
<HTML>
  <HEAD>
    <SCRIPT>
      var i = 1;
      document.write(i++);
      document.write(++i);
    </SCRIPT>
  </HEAD>
</HTML>
```

- a. 12
- b. 22
- c. 13
- d. 23

13. Đoạn mã sau sẽ in ra kết quả?

```
<HTML>
  <HEAD>
    <SCRIPT>
      var x = 11;
      var y = 5
      document.write(x|y);
    </SCRIPT>
  </HEAD>
</HTML>
```

**Bài tập thực hành chương 6:**

1. Dùng JavaScript hiển thị một thông báo “Chào bạn!”
2. Sử dụng prompt cho phép người dùng nhập tên vào, sau đó hiển thị câu chào “Xin chào .....”. Trong đó .... là tên vừa nhập.
3. Cho đoạn mã sau:

```
<HTML>
```

```
    <HEAD>
```

```
        <SCRIPT>
```

```
            var x=prompt ("Please enter the first number: ","");
            var y=prompt ("Please enter the second number: ","");
            r=x+y;
            document.write("The result is: "+r);
```

```
        </SCRIPT>
```

```
    </HEAD>
```

```
</HTML>
```

Dự đoán kết quả của chương trình khi hai số nhập vào là 3 và 5.

Gõ lại đoạn mã trên vào trình soạn thảo để kiểm tra kết quả.

4. Cho người dùng nhập vào một chuỗi, chuyển tất cả các ký tự ‘a’ trong chuỗi thành ‘AB’, sau đó hiển thị lại chuỗi vừa biến đổi  
(Gợi ý: sử dụng phương thức replace())
5. Gõ lại câu hỏi 12 và 13 để kiểm tra kết quả
6. Gõ lại các ví dụ trong chương 1 và 2.

## CHƯƠNG 7

# CÂU LỆNH ĐIỀU KIỆN

### 7.1 Lệnh và khối lệnh

#### 7.1.1 Lệnh và quy ước lệnh trong JavaScript

Cũng như trong hầu hết các ngôn ngữ khác, đơn vị làm việc cơ bản của JavaScript là câu lệnh. Trong hai chương trước, chúng ta đã làm quen với rất nhiều câu lệnh trong JavaScript. Nó có thể là kết quả của một phép gán giá trị cho một biến, có thể là lời gọi một hàm, hay biểu diễn một dạng phép tính, hoặc thậm chí là sự kết hợp của tất cả những công việc đó. Trong các ví dụ trước đây, một trong những câu lệnh mà chúng ta đã làm quen là câu lệnh khai báo, câu lệnh này không những dùng để khởi tạo (hay định nghĩa) một biến mới, mà còn có thể gán giá trị cho nó, ví dụ như:

```
var x = 10;
```

Như đã nói ở trên, một chương trình JavaScript là một tập hợp của các câu lệnh, các câu lệnh này có thể được tổ chức thành từng hàm (sẽ được đề cập trong chương 5). Các câu lệnh JavaScript bao gồm các từ khóa được sử dụng với cú pháp thích hợp và được kết thúc bởi dấu chấm phẩy (;). Một câu lệnh duy nhất có thể nằm trên nhiều dòng. Nhiều câu lệnh cũng có thể được viết trên một dòng duy nhất nếu mỗi câu lệnh được phân tách bởi một dấu chấm phẩy (;).

#### 7.1.2 Khối lệnh

Một khối lệnh được sử dụng để nhóm các câu lệnh. Các câu lệnh này được gọi là đồng cấp và sẽ được nhóm lại bởi một cặp dấu ngoặc mớc ({}).

Bên trong một khối lệnh lại có thể viết lòng khối lệnh khác. Sự lòng nhau theo cách như vậy là không hạn chế.

### 7.2 Các câu lệnh điều kiện

Một câu lệnh điều kiện là một tập hợp các lệnh thi hành nếu điều kiện chỉ định là đúng. Kết quả của điều kiện xác định câu lệnh hoặc khối lệnh sẽ được thực thi. JavaScript cung cấp hai câu lệnh điều kiện: if...else và switch.

#### 7.2.1 Câu lệnh if...else

Câu lệnh này dùng để kiểm tra điều kiện, nó thực thi việc tính toán trên một biểu thức, nó kiểm tra điều kiện là đúng hay sai để thực hiện khối lệnh tương ứng.

Một câu lệnh if đơn giản có cú pháp lệnh như sau:

```
if (điều kiện )
{
    // các câu lệnh ứng với điều kiện đúng
}
```

Đây là cú pháp lệnh đơn giản, nó sẽ kiểm tra nếu điều kiện sau theo sau if là đúng thì khối lệnh sẽ được thực thi.

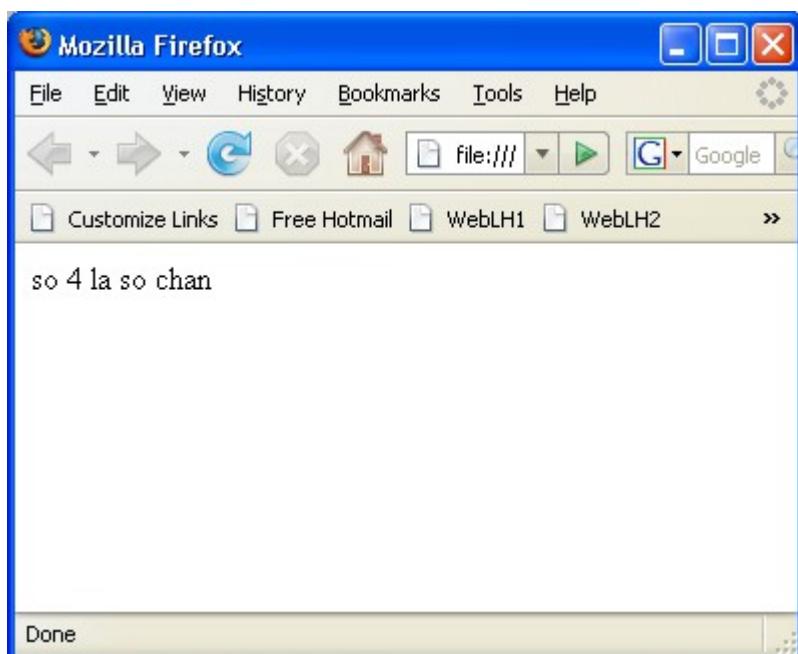
**Ví dụ 7.1:** Kiểm tra một số có phải là số chẵn hay không? Nếu là số chẵn thì hiển thị kết luận số chẵn.

Đối với bài toán này, ta sẽ sử dụng phép chia lấy dư (%) để kiểm tra. Nếu một số thực hiện phép chia lấy dư cho 2 mà trả về kết quả là 0 thì kết luận số đó là số chẵn.

Đoạn mã sau minh họa cho bài toán trên. Trong đoạn mã này, ta lưu ý đến cách sử dụng câu lệnh if:

```
<HTML>
  <HEAD>
    <SCRIPT>
      var x = 4;
      r=x%2;
      if (r==0)
      {
        document.write("so "+x+" la so chan");
      }
    </SCRIPT>
  </HEAD>
</HTML>
```

**Kết quả:**



**Hình 7.1: Câu lệnh điều kiện if đơn giản**

Nếu trong ví dụ trên, ta thay giá trị của  $x = 5$  thì trên màn hình sẽ không xuất hiện gì cả, nói cách khác, nó không thực hiện khối lệnh sau if, vì trong trường hợp này, biểu thức  $r == 0$  trả về giá trị sai (false).

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

Cú pháp như sau:

if (điều kiện)

```

{
    // các câu lệnh ứng với điều kiện đúng
}
else
{
    // các câu lệnh ứng với điều kiện sai
}

```

Cú pháp trên được hiểu như sau: Nếu điều kiện là đúng (true) thì khối lệnh sau if sẽ được thực hiện, và ngược lại, nếu là sai (false) thì khối lệnh sau else sẽ được thực hiện.

Trong cả hai cú pháp lệnh trên, điều kiện có thể là bất cứ biểu thức JavaScript nào có giá trị là true hoặc false. Khối lệnh sau if hoặc else cũng có thể là bất cứ câu lệnh JavaScript nào, kể cả các câu lệnh if được lồng thêm vào trong. Nếu chúng ta muốn sử dụng thêm một hoặc nhiều câu lệnh sau một câu lệnh if hoặc else thì ta phải đóng các câu lệnh bằng các dấu ngoặc mỏng ({}).

Ví dụ sau minh họa cho câu lệnh điều kiện if...else. Trong ví dụ này, ta cũng xét một số là số chẵn hay lẻ, sau đó hiển thị kết quả ra màn hình.

Cũng như ví dụ 3.1, ta cũng sẽ sử dụng phép chia lấy dư (%) để kiểm tra. Nếu một số thực hiện phép chia lấy dư cho 2 mà trả về kết quả là 0 thì kết luận số đó là số chẵn, ngược lại thì kết luận nó là số lẻ

Đoạn mã sau minh họa cho bài toán trên.

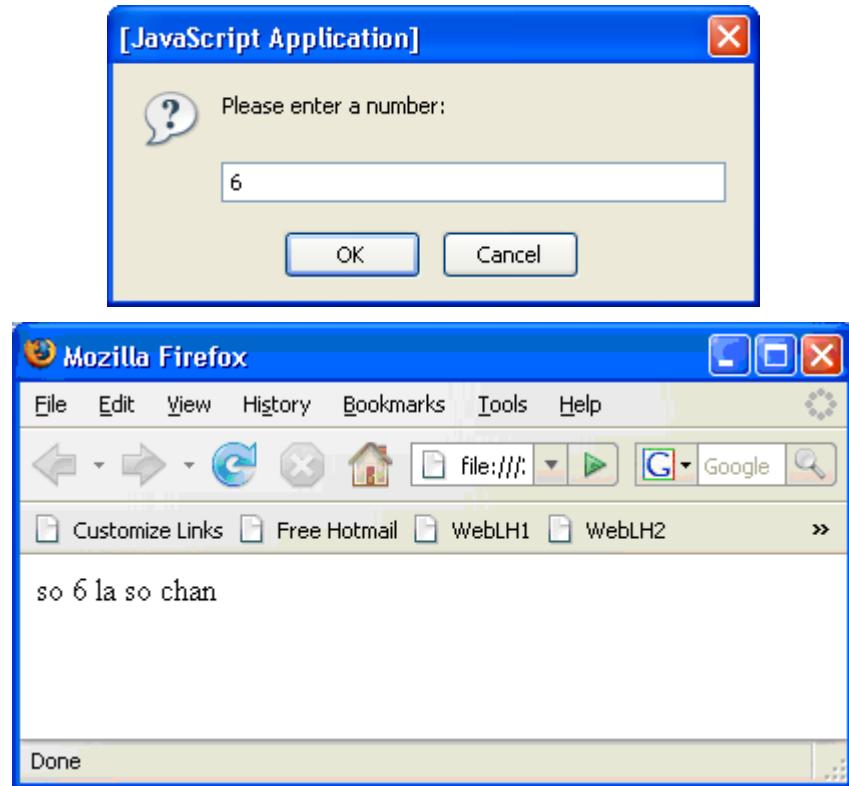
### Ví dụ 7.2:

```

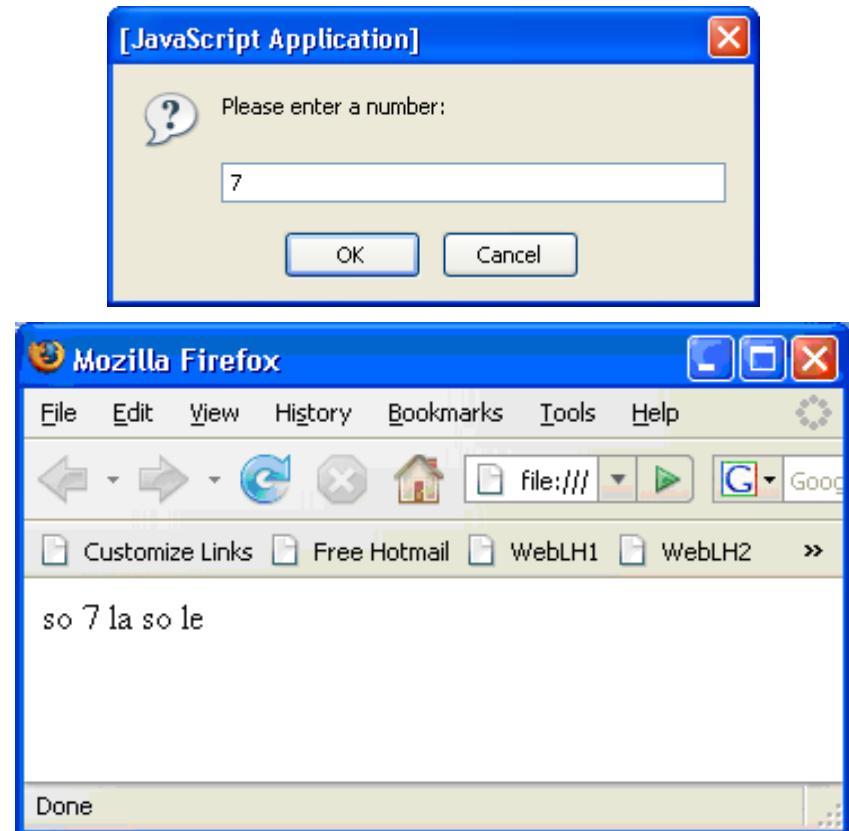
<HTML>
    <HEAD>
        <SCRIPT>
            var x=prompt ("enter a num: ","");
            r=x%2;
            if (r==0)
            {
                document.write("so "+x+" la so chan");
            }
            else
            {
                document.write("so "+x+" la so le");
            }
        </SCRIPT>
    </HEAD>
</HTML>

```

Kết quả:



Hình 7.2.1: Trường hợp nhập vào một số chẵn



Hình 7.2.2: Trường hợp nhập vào một số lẻ

### 7.2.2 Câu lệnh switch

Khi có nhiều tùy chọn if...else thì tốt hơn ta nên sử dụng lệnh switch. Lệnh này còn được xem là lệnh case. Câu lệnh switch cho phép một chương trình định giá trị một biểu thức và thử so khớp giá trị của biểu thức với từng trường hợp. Nếu so khớp thỏa mãn thì chương trình thi hành câu lệnh tương ứng. Nếu không tìm thấy một giá trị nào trong danh sách các case của nó, khỏi lệnh trong phần default sẽ được thực hiện. Lệnh break dùng để thoát ra khỏi câu lệnh switch.

Câu lệnh switch có dạng như sau:

```
switch (expression){
    case label:
        statements;
        break;
    case label:
        statements;
        break;
    ...
    default: statements;
}
```

Đầu tiên chương trình tìm một nhãn trùng khớp với giá trị biểu thức và thi hành câu lệnh tương ứng nếu so khớp thành công. Nếu nhãn so khớp không được tìm thấy, chương trình sẽ tìm đến khỏi lệnh trong lựa chọn default, và nếu tìm thấy sẽ thực hiện câu lệnh tương ứng. Nếu không tìm thấy câu lệnh default, thì chương trình tiếp tục thi hành câu lệnh tiếp theo sau câu lệnh switch.

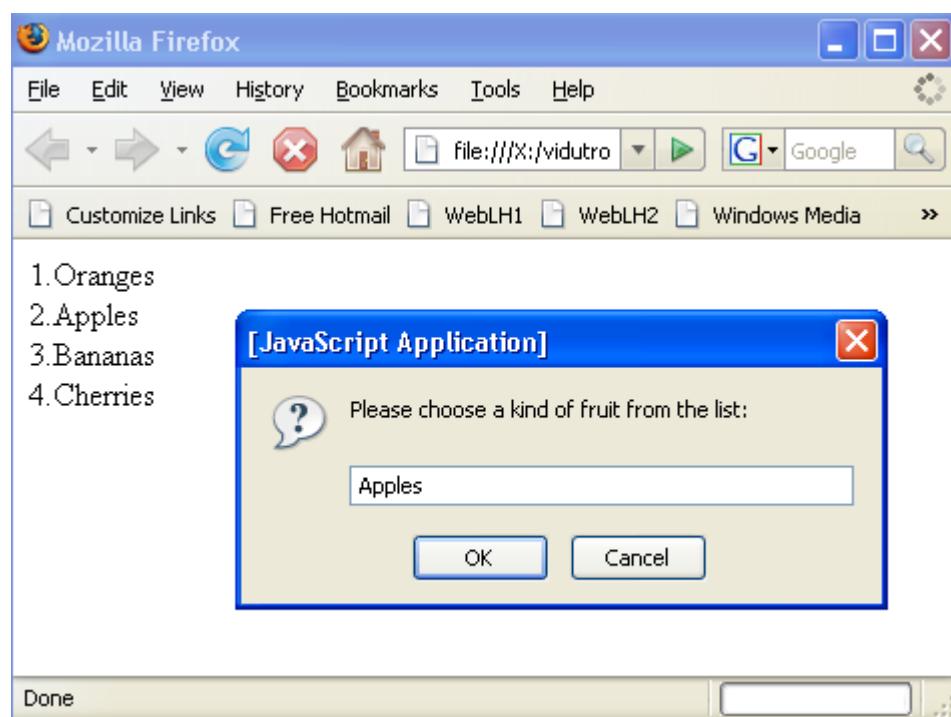
Câu lệnh tùy chọn break kết hợp với mỗi trường hợp đảm bảo rằng chương trình sẽ thoát khỏi lệnh switch khi câu lệnh so khớp được thi hành và tiếp tục thực thi câu lệnh tiếp theo câu lệnh switch. Nếu không sử dụng câu lệnh break thì chương trình vẫn tiếp tục thi hành lệnh kế tiếp trong câu lệnh switch.

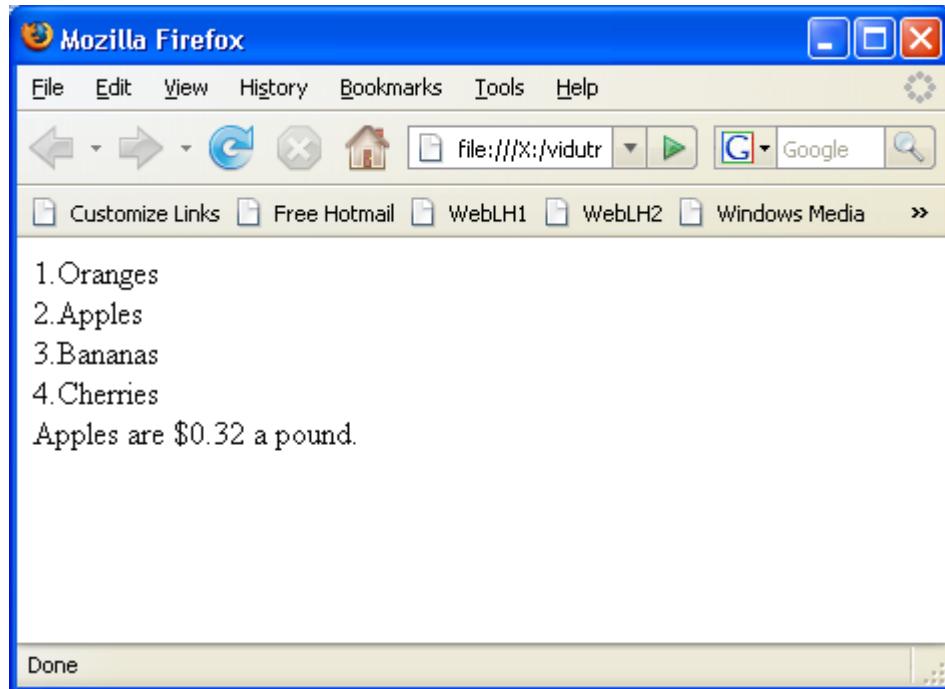
**Ví dụ 7.3:** Trong ví dụ sau, nếu nhập ước lượng đến “Bananas”, thì chương trình so khớp giá trị với trường hợp “Bananas” và thi hành câu lệnh được kết hợp. Khi bắt gặp break thì chương trình ngắt switch và thi hành câu lệnh theo sau switch. Nếu break được bỏ qua, thì câu lệnh cho trường hợp “Cherries” cũng sẽ được thi hành:

```
<HTML>
<HEAD>
<SCRIPT>
document.write("1.Oranges");
document.write("<br>2.Apples");
document.write("<br>3.Bananas");
document.write("<br>4.Cherries");
var exp=prompt ("Vui lòng hãy chọn một loại trái cây trong danh sách:
","");
switch (exp){
```

```
case "Oranges":  
    document.write("<br>Oranges are $0.59 a pound.");  
    break;  
case "Apples":  
    document.write("<br>Apples are $0.32 a pound.");  
    break;  
case "Bananas":  
    document.write("<br>Bananas are $0.48 a pound.");  
    break;  
case "Cherries":  
    document.write("<br>Cherries are $3.00 a pound.");  
    break;  
default:  
    document.write ("<br>Sorry, we have no this kind of fruit!!");  
}  
</SCRIPT>  
</HEAD>  
</HTML>
```

Kết quả:





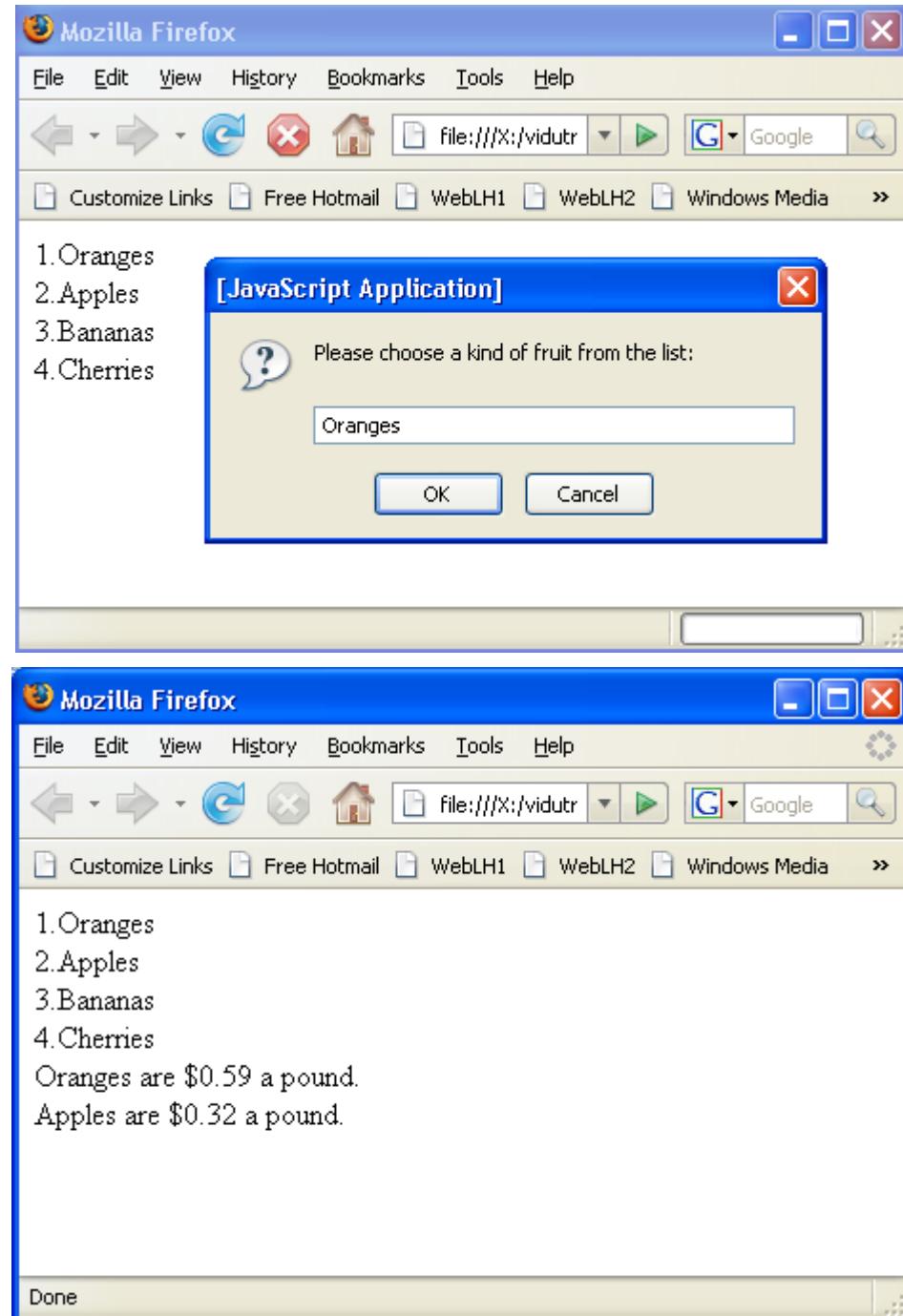
**Hình 7.3.1: Câu lệnh switch**

Ở đây chúng ta cần lưu ý về câu lệnh break trong mỗi case. Như đã nói ở trên, nếu không có break, chương trình sẽ tiếp tục thực hiện khối case khác. Ví dụ, trong đoạn mã trên, nếu ta không kết thúc khối case “Oranges” bằng câu lệnh break, thì chương trình sẽ tiếp tục thực hiện khối lệnh của case “Apples”.

Xét đoạn mã đang đ𝐞 cập:

```
switch (exp){
    case "Oranges":
        document.write("<br>Oranges are $0.59 a pound.");
    case "Apples":
        document.write("<br>Apples are $0.32 a pound.");
        break;
    case "Bananas":
        document.write("<br>Bananas are $0.48 a pound.");
        break;
    ...
}
```

**Kết quả:**



**Hình 7.3.2: Chú ý với sử dụng câu lệnh break**

### 7.3 Câu hỏi và bài tập

#### Câu hỏi:

1. Các câu lệnh trong JavaScript được kết thúc bởi dấu phẩy (,) \_\_\_\_\_ (Đúng/Sai)
2. Một câu lệnh duy nhất có thể nằm trên nhiều dòng. \_\_\_\_\_ (Đúng/Sai)
3. Nhiều câu lệnh không được viết trên một dòng duy nhất cho dù mỗi câu lệnh được phân tách bởi một dấu chấm phẩy (;). \_\_\_\_\_ (Đúng/Sai)
4. Bên trong một khối lệnh có thể có một khối lệnh khác hay không? \_\_\_\_\_ (Có/Không)

5. JavaScript cung cấp hai câu lệnh điều kiện là \_\_\_\_\_ và \_\_\_\_\_.
6. Một câu lệnh if có nhất thiết phải có thành phần else sau hay không?  
\_\_\_\_\_ (Có/Không)
7. Đối với câu lệnh if, chương trình sẽ kiểm tra nếu điều kiện sau theo sau if là \_\_\_\_\_ thì khối lệnh sau if sẽ được thực thi.
8. Đối với câu lệnh switch, nếu chương trình không tìm thấy một giá trị nào trong danh sách các case của nó, khối lệnh trong phần \_\_\_\_\_ sẽ được thực hiện.
9. Lệnh \_\_\_\_\_ dùng để thoát ra khỏi câu lệnh switch.

**Bài tập thực hành chương 7:**

- Viết chương trình cho phép người dùng nhập vào hai số songuyen1 và songuyen2, kiểm tra xem songuyen1 có chia hết cho songuyen2 không, hiện thông báo tương ứng.

Gợi ý: Thực hiện như sau:

- Nhập vào giá trị của 2 số (dùng prompt).
  - Lấy số dư của phép chia songuyen1 cho songuyen2.
  - Nếu số dư này bằng 0 thì in ra thông báo “songuyen1 chia het cho songuyen2”.
  - Nếu số dư này khác 0 thì in thông báo “songuyen1 khong chia het cho songuyen2”.
- Viết chương trình nhập vào ba con số, tìm số lớn nhất trong ba số này.
  - Viết chương trình cho phép người dùng nhập vào 1 năm, kiểm tra năm đó có phải là năm nhuận hay không.

Gợi ý: Năm nhuận là năm chia hết cho 4, ngoại trừ những năm chia hết cho 100 mà không chia hết cho 400. Ví dụ 1700, 1800, 1900 không phải là năm nhuận, các năm 1600, 2000 là các năm nhuận.

- Viết chương trình xếp loại học viên theo điểm số nguyên như sau: (dùng if..else)
  - Nhập điểm từ bàn phím (dùng prompt).
  - In ra thông báo xếp loại tương ứng với điểm như sau:
    - Nếu điểm là 9, 10 thì xếp loại giỏi.
    - Nếu điểm là 7, 8 thì xếp loại khá.
    - Nếu điểm là 5, 6 thì xếp loại trung bình.
    - Nếu điểm là 0, 1, 2, 3, 4 thì xếp loại yếu.
    - Nếu điểm <0 hoặc điểm>10 thì thông báo điểm nhập vào không hợp lệ.
- Viết lại chương trình ở bài 4 bằng cách sử dụng lệnh switch.

## CHƯƠNG 8

# CÂU LỆNH VÒNG LẶP

### 8.1 Các lệnh vòng lặp trong JavaScript

Vòng lặp là một tập hợp các lệnh thi hành lặp đi lặp lại cho đến khi một điều kiện cụ thể được xác định. 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.

JavaScript cung cấp các câu lệnh vòng lặp for, do..while, và while. Ngoài ra chúng ta có thể sử dụng các câu lệnh chuyển điều khiển bên trong các câu lệnh vòng lặp như break, continue và label (mặc dù label không phải là câu lệnh vòng lặp, nhưng nó được sử dụng thường xuyên với các câu lệnh vòng lặp)..

Ngoài ra, trong chương này chúng ta còn tìm hiểu về hai câu lệnh vòng lặp thao tác trên đối tượng đó là for..in và with.

#### *8.1.1 Câu lệnh 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.

Câu lệnh for bao gồm ba thành phần, được phân cách nhau bởi dấu chấm phẩy (;). Cả ba thành phần này đều không bắt buộc phải có, và chúng điều khiển việc thực hiện của vòng lặp for. Nếu có nhiều câu lệnh thực hiện trong thân của vòng lặp, chương trình phải sử dụng cặp dấu ngoặc mớc ({} ) để chứa các câu lệnh đó

Cú pháp lệnh như sau:

```
for([initialExpression];[condition];[incrementExpression]){
    statements;
}
```

Trong đó:

- initialExpression: Lệnh khởi tạo, được thực hiện duy nhất một lần và thường dùng để khởi tạo biến đếm.
- condition: điều kiện của vòng lặp.
- incrementExpression: Lệnh tăng, thay đổi giá trị biến đếm của vòng lặp.
- statements: Các lệnh bên trong vòng lặp.

Vòng lặp for thi hành như sau:

1. Khởi tạo biến thức initialExpression, nếu thành công thì vòng lặp được thi hành. Biến thức này thường dùng để khởi tạo một hoặc nhiều bộ đếm của vòng lặp, và cú pháp cho phép một biến thức có bất kỳ độ phức tạp nào. Biến thức này có thể cũng khai báo các biến.

2. Biểu thức condition được ước lượng. Nếu giá trị của condition là true, thì các câu lệnh của vòng lặp thi hành. Nếu giá trị của condition là false thì thoát khỏi vòng lặp. Nếu bỏ qua hoàn toàn biểu thức condition thì điều kiện luôn được thửa nhận là true.
3. Thực thi statements.
4. Cập nhật biểu thức incrementExpression, và trở về bước 2.

Ví dụ sau tính tổng các số từ 0 đến 9, sử dụng vòng lặp for.

```
tong = 0;
for (var i = 0; i<= 9; i++) {
    tong += i;
}
```

Trong vòng lặp for, ta có thể sử dụng nhiều biểu thức khởi tạo hay biểu thức thay đổi giá trị cho biến đếm bằng cách dùng toán tử dấu phẩy (đã học ở chương 2: Toán tử và biểu thức ).

Xét ví dụ trên, ta có thể khởi tạo giá trị cho biến tổng như là một thành phần của vòng lặp for thông qua toán tử dấu phẩy:

```
for (var i = 0, tong = 0; i<= 9; i++) {
    tong += i;
}
```

**Ví dụ 8.1:** Hàm sau đây có một câu lệnh for đếm số các mục được lựa chọn trong danh sách cuộn (một đối tượng Select có phép có nhiều sự lựa chọn). Câu lệnh for khai báo biến i và khởi tạo cho nó giá trị 0. Vòng lặp sẽ kiểm tra, nếu i nhỏ hơn số tùy chọn trong đối tượng Select, thì thi hành câu lệnh if thành công, và tăng i lên 1 sau khi thi hành xong lần lặp.

<HTML>

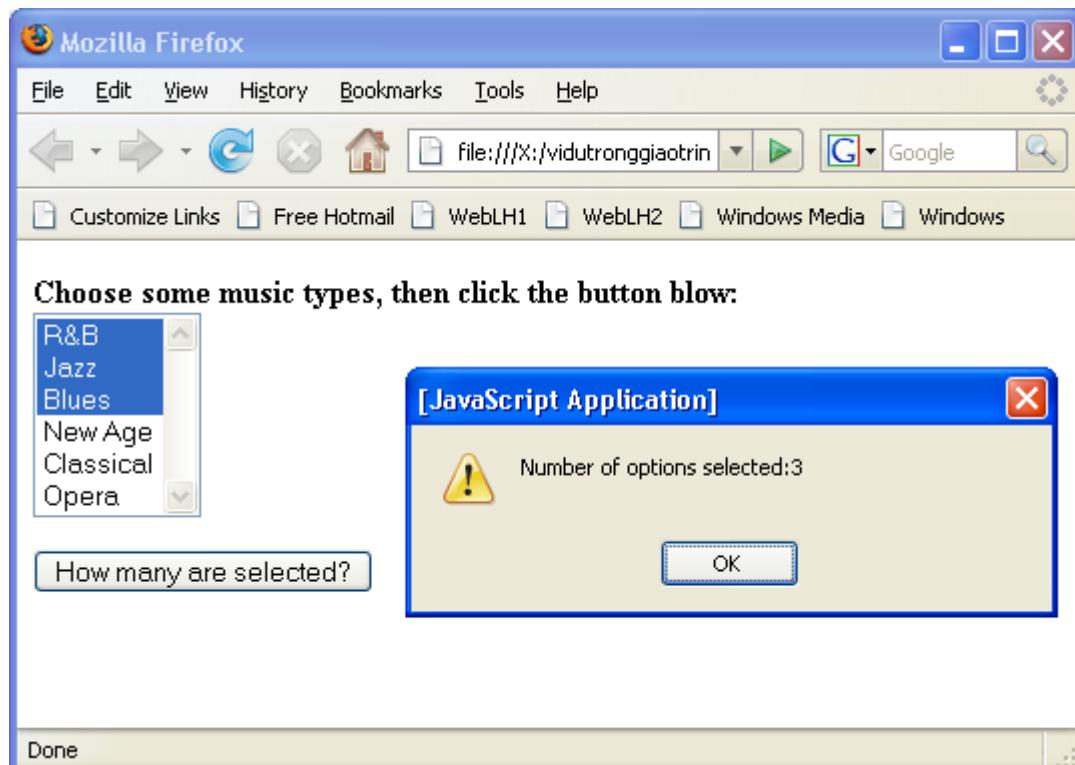
```
<HEAD>
<SCRIPT LANGUAGE="JavaScript">
    function howMany(selectObject){
        var numberSelected = 0;
        for(var i=0;i<selectObject.options.length; i++)
        {
            if(selectObject.options[i].selected == true)
                numberSelected++;
        }
        return numberSelected;
    }
</SCRIPT>
</HEAD>
```

```

<BODY>
    <FORM NAME = "selectForm">
        <P><B>Choose some music types, then click the button blow: </B>
        <BR><SELECT NAME ="musicTypes" MULTIPLE>
            <OPTION SELECTED> R&B
            <OPTION>Jazz
            <OPTION>Blues
            <OPTION>New Age
            <OPTION>Classical
            <OPTION>Opera
        </SELECT>
        <P><INPUT TYPE = "button" VALUE = "How many are selected?">
        onClick = "alert ('Number of options selected:' + howMany(document.selectForm.musicTypes))">
    </FORM>
</BODY>
</HTML>

```

**Kết quả:**



**Hình 8.1: Vòng lặp for**

### 8.1.2 Câu lệnh do..while

Câu lệnh do...while lặp cho tới khi một điều kiện cụ thể có giá trị là false.

Cú pháp lệnh như sau:

```
do {
    statement
} while (condition)
```

Trước hết, câu lệnh này thi hành statement một lần. Tại lúc kết thúc của mỗi lần thi hành vòng lặp, condition được kiểm tra: Nếu condition là true, thì câu lệnh tiếp tục được thi hành một lần nữa, ngược lại, nếu điều kiện là false, thì thi hành ngừng và điều khiển được chuyển tới câu lệnh kế tiếp câu lệnh do...while.

**Ví dụ 8.2:** Trong ví dụ sau, vòng lặp do...while làm đi làm lại cho đến khi biến i không còn nhỏ hơn 5 nữa.

```
do {
    i += 1;
    document.write (i);
} while (i<5)
```

### 8.1.3 Câu 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 ngoặc mỏng ({} ) để chứa các câu lệnh 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 nó kiểm tra điều kiện trước, và 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 lệnh như sau:

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

Nếu điều kiện là false, thì các câu lệnh trong vòng lặp dừng thi hành và điều khiển được chuyển tới câu lệnh sau vòng lặp.

Việc kiểm tra điều kiện xảy ra trước khi các câu lệnh trong vòng lặp được thi hành. Nếu điều kiện trả về là true, thì các câu lệnh trong vòng lặp được thi hành và điều kiện được kiểm tra lại một lần nữa. Nếu điều kiện trả về là false, thì dừng thi hành và điều khiển được chuyển tới câu lệnh kế tiếp câu lệnh while.

**Ví dụ 8.3:** Vòng lặp while sau đây lặp đi lặp lại miễn là n nhỏ hơn 3:

```
n = 0;
x = 0;
while (n <3){
    n++;
    x += n;
```

```

    }

```

Mỗi lần lặp lại, vòng lặp tăng n và gán giá trị đó cho x. Vì thế, x và n sẽ có các giá trị sau:

- Sau vòng lặp đầu tiên: n = 1 và x = 1
- Sau vòng lặp thứ 2: n = 2 và x = 3
- Sau vòng lặp thứ 3: n = 3 và x = 6

Sau khi kết thúc vòng lặp thứ 3, điều kiện  $n < 3$  không còn đúng, như vậy vòng lặp kết thúc.

Chúng ta lưu ý rằng phải đảm bảo điều kiện trong vòng lặp cuối cùng sẽ có giá trị là false, nếu không thì vòng lặp sẽ không bao giờ thoát, lúc này chúng ta nói vòng lặp vô tận.

**Ví dụ 8.4:** Các câu lệnh trong vòng lặp while sau đây thi hành mãi mãi bởi vì điều kiện không bao giờ có giá trị là false:

```

while (true) {
    alert ("Hello, word!")
}

```

## 8.2 Các lệnh chuyển điều khiển trong vòng lặp

### 8.2.1 Câu lệnh label

Một label bao gồm một câu lệnh với một danh hiệu cho phép chúng ta tham khảo tới nó ở một nơi khác trong chương trình của bạn. Ví dụ, bạn có thể sử dụng một nhãn để chỉ ra một vòng lặp, và sau đó sử dụng các câu lệnh break hoặc continue để chỉ ra một chương trình sẽ thoát khỏi vòng lặp hoặc tiếp tục thi hành nó.

Cú pháp của câu lệnh label như sau:

```

label:
    statement

```

Giá trị của label có thể là bất cứ danh hiệu nào của JavaScript nhưng không phải là từ khóa có sẵn của JavaScript.

statement có thể là bất cứ câu lệnh nào.

**Ví dụ 8.5 :** Trong ví dụ này, nhãn markLoop chỉ ra một vòng lặp while.

```

markLoop:
    while (theMark == true) {
        doSomething ();
    }

```

### 8.2.2 Câu lệnh break

Các vòng lặp for, while và do...while 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.

Chúng ta sử dụng câu lệnh break để thoát khỏi vòng lặp, câu lệnh switch hoặc câu lệnh label.

- Khi chúng ta sử dụng break mà không có một label, nó thoát khỏi vòng lặp while, do...while, for hoặc câu lệnh switch ngay lập tức và chuyển điều khiển tới câu lệnh sau.
- Khi chúng ta sử dụng break với một nhãn (label), nó nhảy tới câu lệnh được gán nhãn cụ thể.

Cú pháp của câu lệnh break như sau:

1. break
2. break label

Dạng thứ nhất của cú pháp thoát ngay ra khỏi vòng lặp hoặc câu lệnh switch, dạng thứ hai nhảy tới câu lệnh có label đính kèm.

**Ví dụ 8.6:** Ví dụ sau đây lặp đi lặp lại thông qua các phần tử trong một mảng cho tới khi nó tìm thấy chỉ số của một phần tử mà giá trị của nó là theValue:

```
for (i = 0; i < a.length; i++) {
    if (a[i] = theValue)
        break;
}
```

**Ví dụ 8.7:** Hàm sau có câu lệnh break chấm dứt vòng lặp while khi i là 3, và sau đó trả về giá trị  $3*x$ .

```
function testBreak (x) {
    var i = 0;
    while (i<6) {
        if (i == 3)
            break;
        i++;
    }
    return i*x;
}
```

### 8.2.3 Câu lệnh continue

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. Nó có thể được sử dụng để khởi động lại một câu lệnh while, do...while, for hoặc câu lệnh label.

- Khi chúng ta sử dụng câu lệnh continue mà không có label, thì nó dừng lần lặp hiện tại của câu lệnh while, do...while, hoặc for và tiếp tục thi hành vòng lặp ở lần lặp tiếp theo. Trái với câu lệnh break, continue không kết thúc sự thi hành của toàn bộ vòng lặp. Trong vòng lặp while, nó nhảy

ngược trở lại phần điều kiện. Trong vòng lặp for, nó nhảy tới phần incrementExpression.

- Khi chúng ta sử dụng continue với một label, thì nó tiếp tục với câu lệnh lặp được chỉ ra với label đó.

Cú pháp của câu lệnh continue như sau:

1. continue
2. continue label

**Ví dụ 8.8:** Ví dụ sau trình bày vòng lặp while với câu lệnh continue thi hành khi giá trị của i bằng 3.

```
i = 0;
n = 0;
while (i<5) {
    i++;
    if (i == 3)
        continue;
    n+=i;
}
```

**Ví dụ 8.9:** Trong ví dụ sau, câu lệnh được gắn nhãn checkiandj chứa một câu lệnh được gắn nhãn checkj. Nếu gặp phải continue, chương trình kết thúc lần lặp hiện tại của checkj và bắt đầu lần lặp tiếp theo. Mỗi lần gặp phải continue, checkj lặp lại cho tới khi điều kiện của nó trả về giá trị false. Khi giá trị false được trả về, phần còn lại của câu lệnh checkiandj được hoàn thành, và checkiandj lặp lại cho tới khi điều kiện của nó trả về false. Khi giá trị false được trả về, chương trình tiếp tục tại câu lệnh sau checkiandj.

Nếu câu lệnh continue có một nhãn checkiandj, chương trình sẽ tiếp tục tại đầu của câu lệnh checkiandj.

```
checkiandj:
while (i<4) {
    document.write (i + "<BR>");
    i +=1;
    checkj:
    while (j>4) {
        document.write (j + "<BR>");
        j -=1;
        if ((j%2) == 0)
            continue checkj;
        document.write (j + "is odd. <BR>");
    }
}
```

```

        document.write("i =" + i + "<BR>");
        document.write("j =" + j + "<BR>");
    }

```

### 8.3 Các lệnh thao tác trên đối tượng

JavaScript sử dụng các câu lệnh for...in và with để thao tác trên các đối tượng.

#### 8.3.1 Câu lệnh for...in

Câu lệnh for...in lặp đi lặp lại một biến chỉ định trên tất cả các thuộc tính của một đối tượng. Với mỗi thuộc tính riêng, JavaScript thực thi các câu lệnh có thể. Ví dụ chúng ta có thể sử dụng câu lệnh for...in để thực hiện một khối các câu lệnh cho mỗi phần tử của mảng.

Cú pháp lệnh như sau:

```

for (variable in object) {
    statements;
}

```

**Ví dụ 8.10:** Hàm sau đây có đối số là một đối tượng và tên của đối tượng. Sau đó nó lặp đi lặp lại trên toàn bộ thuộc tính của đối tượng và trả về một chuỗi liệt kê tên của các thuộc tính và giá trị của chúng.

```

function dump_props (obj, obj_name) {
    var result = "";
    for (var i in obj) {
        result += obj_name + "." + i + "=" + obj[i] + "<BR>"
    }
    result += "<HR>";
    return result;
}

```

Với một đối tượng car, các thuộc tính make và model, thì result sẽ là:

```

car.make = Ford
car.model = Mustang

```

**Ví dụ 8.11:** 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.

```

<HTML>
<HEAD>
    <TITLE> Datatype Example </TITLE>
    <SCRIPT LANGUAGE= "JavaScript">
        color = new Array ("red", "blue", "green");
        var record = "color";

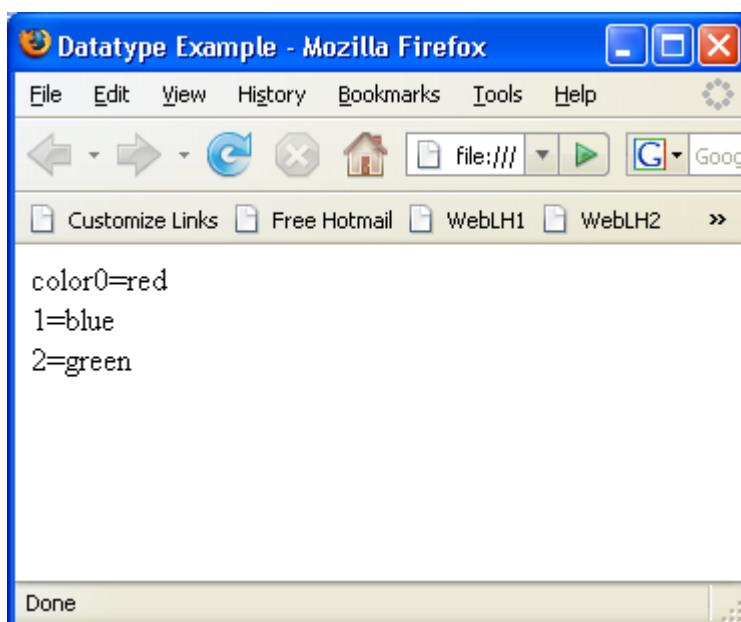
```

```

for (var prop in color){
    record += prop + "=" + color[prop] + "<BR>"
}
record += "<BR>"
document.write (record)
</SCRIPT>
</HEAD>
</HTML>

```

**Kết quả:**



**Hình 8.1: Câu lệnh for...in**

### 8.3.2 Câu lệnh with

Câu lệnh with thiết lập đối tượng mặc định cho một tập hợp các câu lệnh. JavaScript tìm kiếm bất cứ cái tên tuyệt đối nào nằm trong tập hợp các câu lệnh để xác định xem các tên này có là các thuộc tính của đối tượng mặc định hay không. Nếu một tên tuyệt đối so khớp với một thuộc tính, thì thuộc tính được sử dụng nằm trong câu lệnh, nếu không thì một biến cục bộ hoặc toàn cục được sử dụng.

Cú pháp lệnh như sau:

```

with (object) {
    statements;
}

```

**Ví dụ 8.12:** Câu lệnh with sau đây cho thấy đối tượng Math là đối tượng mặc định. Các câu lệnh theo sau câu lệnh with tham khảo tới thuộc tính PI và các phương thức cos và sin mà không chỉ rõ một đối tượng. JavaScript thừa nhận đối tượng Math cho các tham khảo này.

```
var a, x, y;
```

```

var r = 10
with (Math) {
    a = PI * r * r;
    x = r * cos(PI);
    y = r * sin(PI/2);
}

```

#### 8.4 Câu hỏi và bài tập

1. Khi sử dụng một vòng lặp, ta phải xác định số lần lặp cụ thể. \_\_\_\_\_ (Đúng/Sai)
2. Vòng lặp for sẽ thực hiện lặp đi lặp lại khối lệnh cho đến khi điều kiện là \_\_\_\_\_ (true/false)
3. Câu lệnh for bao gồm 3 thành phần, được phân cách nhau bởi dấu \_\_\_\_\_
4. Một vòng lặp for phải có đầy đủ 3 thành phần. \_\_\_\_\_ (Đúng/Sai)
5. Trong vòng lặp for, ta có thể sử dụng nhiều biến thức khởi tạo hay biến thức thay đổi giá trị cho biến đếm bằng cách dùng toán tử \_\_\_\_\_
6. Câu lệnh do...while lặp cho tới khi một điều kiện cụ thể có giá trị là false. \_\_\_\_\_ (Đúng/Sai)
7. Câu lệnh while được dùng để thực hiện một khối các câu lệnh chừng nào điều kiện còn là \_\_\_\_\_
8. Khác biệt chính giữa vòng lặp while và do...while là vòng lặp \_\_\_\_\_ có thể không được thực hiện một lần nào vì nó kiểm tra điều kiện trước.

#### Bài tập thực hành chương 8:

1. Viết chương trình tính tổng các số từ 1 đến 200.
2. Viết chương trình tính tích các số lẻ từ 1 đến 300.

3. Viết chương trình cho phép người dùng nhập vào một số. Kiểm tra số này có phải là số nguyên tố không.

*Gợi ý:* Số nguyên tố là số bắt đầu từ 2, chỉ chia hết cho 1 và cho chính nó.

4. Viết chương trình cho phép người dùng nhập vào hai số ( $x$  và  $n$ ). Tính  $x^n$ .
5. Cho người dùng nhập tên vào một hộp nhập, dữ liệu nhập vào không được trống. Nếu nhập trống phải cho nhập lại.
6. In ra tất cả các năm nhuần từ 1699 đến 2008.

*Gợi ý:* Xem cách kiểm tra năm nhuần ở bài tập chương 3.

## CHƯƠNG 9

# HÀM

### **9.1 Khái niệm và các thao tác trên hàm**

#### **9.1.1 Khái niệm về hàm**

Hàm là một trong những khái niệm cơ bản được xây dựng trong JavaScript. Một hàm trong JavaScript khá giống với một thủ tục hay chương trình con trong ngôn ngữ lập trình. Một hàm được định nghĩa là một tập các câu lệnh, thực hiện một nhiệm vụ cụ thể nào đó.

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

JavaScript hỗ trợ nhiều hàm định nghĩa sẵn mà chúng ta sẽ sử dụng trong các script. Ngoài ra, người dùng có thể tự định nghĩa các hàm khác để sử dụng.

Sau đây ta sẽ tìm hiểu một số thao tác trên hàm.

#### **9.1.2 Tạo hàm**

Thao tác này sử dụng cho các hàm tự định nghĩa. Để sử dụng một hàm, đầu tiên ta phải tạo ra, hay còn gọi là định nghĩa hàm, sau đó script có thể gọi nó. Định nghĩa một hàm là một quá trình khai báo tên của hàm và các lệnh sẽ được thực thi khi gọi hàm.

Các thành phần của một hàm gồm có:

- Từ khóa function.
- Tên hàm.
- Danh sách các đối số của hàm, nằm trong các dấu ngoặc đơn () và được phân cách bởi các dấu phẩy (,). Một hàm có thể không có đối số, nhưng chúng ta vẫn phải có cặp dấu ngoặc đơn () đặt sau tên hàm.
- Các câu lệnh JavaScript định nghĩa hàm, nằm trong cặp dấu ngoặc mỏng {} . Các câu lệnh trong một hàm có thể bao gồm các lệnh gọi đến các hàm khác đã định nghĩa trong ứng dụng hiện hành.

Cú pháp của một hàm như sau:

```
function functionName (argument1, argument2, ...)
{
    statements;
}
```

*Lưu ý một điều là trong JavaScript, các hàm không thể lồng nhau.  
Có nghĩa là một hàm không thể được định nghĩa bên trong thân  
một hàm khác.*

Ví dụ, đoạn mã nguồn sau định nghĩa một hàm đơn giản có tên là square:

```
function square (number) {
    return number * number;
}
```

Hàm square có một đối số gọi là number, có một câu lệnh trả về giá trị, giá trị trả về của hàm bằng đối số của hàm nhân với chính nó. Câu lệnh return được dùng để chỉ định giá trị trả về của hàm (xem trang 5.1.4: Câu lệnh return).

Tất cả các tham biến được truyền cho các hàm bằng *giá trị*, giá trị được truyền tới hàm; nhưng nếu hàm thay đổi giá trị của tham biến, thì sự thay đổi này không được phản ánh một cách toàn cục hoặc không phản ánh trong việc gọi hàm. Tuy nhiên, nếu ta truyền một đối tượng như một tham biến cho một hàm và hàm thay đổi các thuộc tính của đối tượng, thì sự thay đổi có thể thấy được bên ngoài của hàm, như được minh họa trong ví dụ sau:

```
function myFunc(theObject) {
    theObject.make = "Toyota"
}
mycar = {make:"Honda", model:"Accord", year:1998};
x = mycar.make; // Trả về Honda
myFunc(mycar); // Truyền đối tượng mycar cho hàm
y = mycar.make; // Trả về Toyota (thuộc tính bị thay đổi bởi hàm)
```

Một hàm có thể được định nghĩa dựa trên một điều kiện. Ví dụ, xét định nghĩa hàm sau:

```
if (num == 0)
{
    function myFunc (theObject) {
        theObject.make = "Toyota"
    }
}
```

Hàm myFunc chỉ được định nghĩa nếu biến num bằng 0. Nếu num khác 0, thì hàm không được định nghĩa và bất cứ sự cố gắng nào để thực hiện nó cũng sẽ thất bại.

Một hàm cũng có thể được định nghĩa bên trong một biểu thức, hàm này được gọi là biểu thức hàm. Diễn hình cho loại hàm này là hàm không cần có tên. Ví dụ, hàm square trên có thể được định nghĩa như sau:

```
const square = function (number) {return number * number};
```

Điều này thuận lợi khi truyền một hàm như một đối số cho một hàm khác. Ví dụ sau trình bày hàm map được định nghĩa và được gọi với một hàm không tên như là tham biến đầu tiên của nó.

```
function map(f,a) {
    var result = new Array;
```

```

for (var i = 0; i != a.length; i++)
    result[i] = f(a[i]);
return result;
}

```

Việc gọi:

```
map(function(x){return x * x * x },[0, 1, 2, 5, 10]);
```

Trả về giá trị: [0, 1, 8, 125, 1000].

### 9.1.3 Gọi hàm

Định nghĩa một hàm không có nghĩa là thi hành hàm đó. Định nghĩa hàm đơn giản là đặt tên cho hàm và chỉ ra những gì sẽ làm khi hàm được gọi. Việc gọi hàm thực tế là thực hiện các hành động cụ thể cùng với các tham biến xác định. Như vậy, 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, nếu chúng ta định nghĩa hàm square, ta có thể gọi nó như sau:

```
square (5)
```

Câu lệnh trên gọi hàm với đối số bằng 5. Hàm thực hiện các câu lệnh của nó và trả về giá trị là 25.

Các đối số của một hàm không bắt buộc phải là các chuỗi hay các số. Ta cũng có thể truyền các đối tượng cho một hàm.

Một hàm thậm chí có thể được đệ quy, hàm đệ quy là hàm có thể gọi lại chính nó.

Ví dụ, xét một hàm tính giai thừa của một số:

```

function factorial (n) {
    if ((n == 0) || (n = 1))
        return 1
    else {
        var result = (n * factorial(n -1));
        return result
    }
}

```

Chúng ta có thể tính giai thừa của các số từ 1 đến 5 như sau:

```

a = factorial(1) // Trả về 1
b = factorial(2) // Trả về 2
c = factorial(3) // Trả về 6
d = factorial(4) // Trả về 24
e = factorial(5) // Trả về 120

```

### 9.1.4 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;

hoặc:

return (value);

**Ví dụ:**

<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>

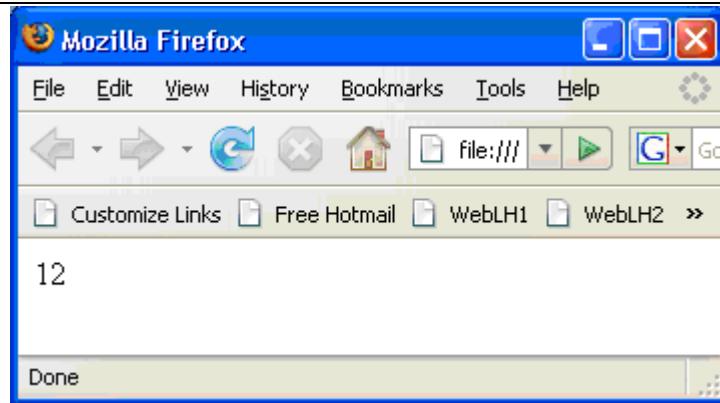
<BODY>

```
<script>
    testreturn(4);
</script>
```

</BODY>

</HTML>

**Kết quả:**



**Hình 9.1: 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. Quyền điều khiển ngay lập tức trả về cho câu lệnh đứng sau lệnh gọi hàm.

## 9.2 Một số hàm thông dụng được hỗ trợ bởi JavaScript

Như ta đã biết, JavaScript có nhiều hàm được định nghĩa sẵn mà khi cần chúng ta chỉ việc gọi. Ví dụ, nếu chúng ta muốn kiểm tra giá trị mà người dùng nhập vào có phải là một số hay không, chúng ta có thể sử dụng hàm isNaN (NaN: Not a Number) để thực hiện điều này.

Trong phần này chúng ta sẽ tìm hiểu về một số hàm thông dụng được hỗ trợ bởi JavaScript như sau:

- Hàm eval
- Hàm isFinite
- Hàm isNaN
- Hàm parseInt và parseFloat
- Hàm Number và String

### 9.2.1 Hàm eval

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

Cú pháp của hàm như sau:

`eval (string)`

Với string là chuỗi cần được đánh giá. Chuỗi này có thể là một biểu thức JavaScript, một câu lệnh, hay một nhóm các câu lệnh. Trong biểu thức có thể bao gồm các biến và thuộc tính của một đối tượng.

Nếu chuỗi đại diện cho một biểu thức thì hàm eval định giá trị biểu thức đó. Nếu đối số đại diện cho một hoặc nhiều câu lệnh JavaScript, thì hàm eval thực hiện các câu lệnh này. Không dùng hàm eval để định giá trị một biểu thức số học, vì JavaScript định giá trị các biểu thức số học một cách tự động.

### 9.2.2 Hàm isFinite

Hàm isFinite định giá trị một đối số để xác định xem nó có phải là một số hữu hạn hay không.

Cú pháp của hàm như sau:

```
isFinite(number)
```

Với number là số được định giá trị.

Nếu đối số là NaN, dương vô cùng hoặc âm vô cùng thì phương thức này trả về false, ngoài ra nó trả về true.

Đoạn mã nguồn sau kiểm tra đối số ClientInput để xác định xem nó có phải là số hữu hạn không:

```
if (isFinite(ClientInput) == true)
{
    /* các bước cụ thể*/
}
```

### **9.2.3 Hàm isNaN**

Hàm isNaN định giá trị một đối số để xác định xem nó có phải là “NaN” (Not a Number) hay không.

Cú pháp của hàm như sau:

```
isNaN(testValue)
```

Với testValue là giá trị bạn muốn định giá trị.

Các hàm parseInt và parseFloat trả về “NaN” khi chúng định giá trị một giá trị không phải là một số. Hàm isNaN trả về true nếu nó được truyền giá trị “NaN” và ngược lại là false.

Đoạn mã nguồn sau định giá trị floatValue để xác định xem nó có phải là một số hay không và sau đó gọi một thủ tục phù hợp:

```
floatValue = parseFloat (toFloat)
if (isNaN (floatValue)) {
    notFloat()
}
else {
    isFloat()
}
```

### **9.2.4 Các hàm parseInt và parseFloat**

Hai hàm parseInt và parseFloat trả về một giá trị số khi cho đối số là một chuỗi.

Cú pháp của hàm parseFloat là:

```
parseFloat(str)
```

Hàm parseFloat phân tích đối số của nó là chuỗi str, và cố gắng trả về một số dấu chấm động. Nếu nó gặp phải một ký tự khác với một dấu (+ hoặc -), một số (0-9), một dấu chấm thập phân, hoặc một số mũ, thì nó trả về giá trị cho đến vị trí đó và bỏ qua

ký tự đó và tất cả các ký tự theo sau. Nếu ký tự đầu tiên không thể được chuyển đổi thành một số, thì nó trả về “NaN”.

Cú pháp của hàm parseInt là:

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

Hàm parseInt phân tích đổi số đầu tiên của nó là chuỗi str và cố gắng trả về một số nguyên của cơ số radix chỉ định, được chỉ ra bởi đổi số tùy chọn thứ hai là radix.

Ví dụ, một cơ số mười được chỉ ra để chuyển đổi chuỗi thành một số thập phân, cơ số 8 – hệ bát phân để chuyển đổi chuỗi thành một số bát phân, cơ số 16 – thập lục phân để chuyển đổi chuỗi thành một số thập lục phân... Với các cơ số trên 10, các chữ cái của bảng ký tự chỉ ra các chữ số lớn hơn 9. Ví dụ, với các số thập lục phân (base 16), các chữ từ A đến F được sử dụng.

Nếu hàm parseInt gặp phải một ký tự không phải là một chữ số trong hệ cơ số chỉ định, thì nó sẽ bỏ qua số đó và tất cả các ký tự theo sau và trả về giá trị số nguyên được phân tích đến vị trí đó. Nếu ký tự đầu không thể được chuyển đổi thành một số trong hệ cơ số chỉ định, thì nó trả về “NaN”. Hàm parseInt rút ngắn chuỗi thành các giá trị số nguyên.

#### **9.2.5 Các hàm Number và String**

Các hàm Number và String cho phép bạn chuyển đổi một đối tượng thành một số hay thành một chuỗi.

Cú pháp của các hàm này là:

```
Number (objRef)
```

```
String (objRef)
```

Với objRef là một tham chiếu đối tượng.

Ví dụ sau chuyển đổi đối tượng Date thành một chuỗi có thể đọc được.

```
D = new Date (430054663215)
// trả về kết quả dưới đây
// "Thu Aug 18 18:37:43 UTC+0700 1983"
x = String (D)
```

### **9.3 Câu hỏi và bài tập**

1. Một hàm được định nghĩa là một tập các \_\_\_\_\_, thực hiện một nhiệm vụ cụ thể.
2. Các hàm bắt buộc phải có một hay nhiều tham số truyền vào và một giá trị trả về \_\_\_\_\_ (Đúng/Sai)
3. Trong JavaScript, người dùng chỉ có thể sử dụng các hàm định nghĩa sẵn chứ không được phép tự định nghĩa một hàm. \_\_\_\_\_ (Đúng/Sai)
4. Để định nghĩa một hàm thì ta phải bắt đầu bằng từ khóa \_\_\_\_\_
5. Danh sách các đối số của hàm, nằm trong các dấu \_\_\_\_\_ và được phân cách bởi các dấu \_\_\_\_\_
6. Nếu một hàm không có đối số, thì không cần cặp dấu ngoặc đơn () đặt sau tên hàm \_\_\_\_\_ (Đúng/Sai)

7. Trong JavaScript, các hàm không thể lồng nhau. Có nghĩa là một hàm không thể được định nghĩa bên trong thân một hàm khác \_\_\_\_\_(Đúng/Sai)
8. Tất cả các tham biến được truyền cho các hàm bằng \_\_\_\_\_
9. Hàm \_\_\_\_\_ là hàm có thể gọi lại chính nó.
10. Dùng lệnh return trong một hàm là không bắt buộc \_\_\_\_\_(Đúng/Sai)
11. Hàm \_\_\_\_\_ định giá trị một đối số để xác định xem nó có phải là một số hữu hạn hay không.
12. Hai hàm parseInt và parseFloat trả về một giá trị \_\_\_\_\_ khi cho đối số là một \_\_\_\_\_.
13. Các hàm Number và String cho phép bạn chuyển đổi một \_\_\_\_\_ thành một số hay thành một chuỗi.
14. Nếu đối số của hàm isNaN là một số thì giá trị trả về của hàm là \_\_\_\_\_(True/False).

**Bài tập thực hành chương 9:**

1. Hãy in ra tất cả các năm nhuần từ 1699 đến 2008 với định dạng: 5 năm in ra trên một dòng.

Yêu cầu: Viết hàm tự định nghĩa check (year) để kiểm tra một năm có phải là năm nhuần hay không.

2. Viết chương trình in tất cả các số nguyên tố từ 2 đến 300.

Yêu cầu: Viết hàm tự định nghĩa checknum (num) để kiểm tra một số có phải là số nguyên tố hay không.

3. Viết chương trình cho phép người dùng nhập vào ba số, kiểm tra xem ba số đó có phải là chiều dài ba cạnh của một tam giác hay không. Nếu đúng thì tính chu vi của tam giác này.

Yêu cầu: Viết hai hàm tự định nghĩa checknums() và cal() để kiểm tra ba số có phải là ba cạnh của tam giác hay không và để tính chu vi của tam giác đó.

## CHƯƠNG 10

# MẢNG

### **10.1 Khái niệm về mảng và các thao tác trên mảng trong JavaScript**

#### **10.1.1 Khái niệm về 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 tường minh. Nếu muốn sử dụng mảng, ta có thể sử dụng đối tượng Array sẵn có và các phương thức của nó để làm việc với các mảng trong ứng dụng của mình. Đối tượng Array có các phương thức để thao tác mảng theo nhiều cách khác nhau, như liên kết, đổi chiều, và sắp xếp chúng. Nó có một thuộc tính để xác định chiều dài mảng và các thuộc tính khác sử dụng với các biểu thức regular.

Một mảng được định nghĩa là một tập thứ tự các giá trị được tham khảo tới bằng tên và chỉ mục của nó. Ví dụ, bạn có thể có một mảng có tên là emp chứa tên của các nhân viên được đánh chỉ số bởi số hiệu của nhân viên. Như vậy emp[0] sẽ là nhân viên thứ nhất, emp[1] sẽ là nhân viên thứ hai...

#### **10.1.2 Tạo mảng**

Cú pháp sau đây dùng để tạo mảng:

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

hoặc:

```
arrayObjectName=new Array(arrayLength)
```

Trong đó:

- arrayObjectName là tên của đối tượng mới hoặc thuộc tính của đối tượng có sẵn. Khi sử dụng các thuộc tính và các phương thức của Array, thì arrayObjectName là tên của đối tượng mảng mới.
- element0, element1, ..., elementN là một danh sách các giá trị cho các phần tử của mảng. Khi sử dụng dạng này, mảng được khởi tạo với các giá trị cụ thể bằng các phần tử của nó, và thuộc tính length của mảng được thiết lập bằng số các đối số.
- arrayLength là độ dài khởi tạo của mảng. Ví dụ, mã nguồn sau đây tạo ra một mảng có năm phần tử:

```
billingMethod = new Array(5).
```

Các mảng chữ cũng là các đối tượng Array. Ví dụ sau là một mảng chữ:

```
coffees = ["French Roast", "Columbian", "Kona"]
```

#### **10.1.3 Gán giá trị cho các phần tử mảng**

Chúng ta có thể gán giá trị cho một mảng bằng cách gán các giá trị cho các phần tử của nó.

Ví dụ:

```
emp[0] = "Casey Jones"
emp[1] = "Phil Lesh"
emp[2] = "August West"
```

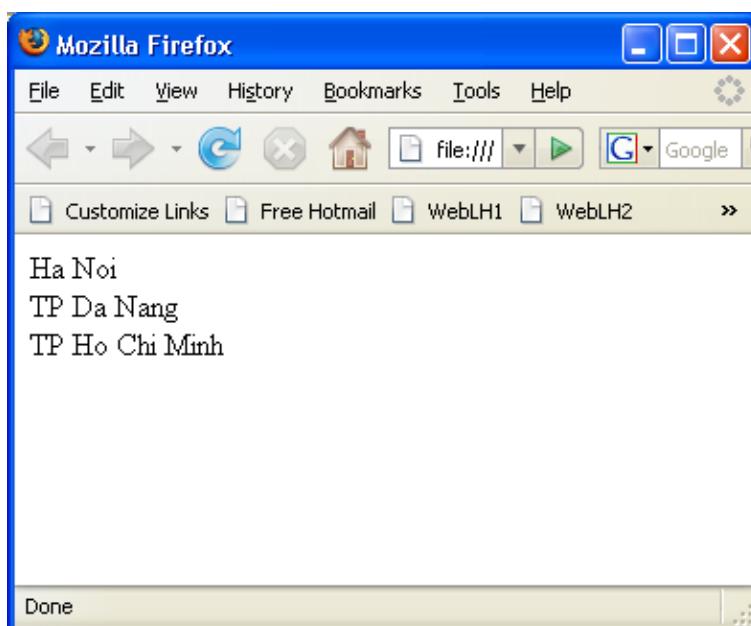
Ta cũng có thể gán giá trị cho một mảng ngay khi tạo ra nó:

```
myArray = new Array ("Hello", myVar, 3.14159)
```

### Ví dụ 10.1:

```
<HTML>
<HEAD>
<SCRIPT>
myArray = new Array(3);
myArray[0] = "Ha Noi";
myArray[1] = "TP Da Nang";
myArray[2] = "TP Ho Chi Minh";
document.writeln(myArray[0] + "<BR>");
document.writeln(myArray[1] + "<BR>");
document.writeln(myArray[2] + "<BR>");
</SCRIPT>
</HEAD>
</HTML>
```

Kết quả:



**Hình 10.1: Ví dụ mảng**

#### 10.1.4 Truy cập đến các phần tử mảng

Chúng ta truy cập tới các phần tử của mảng bằng cách sử dụng số thứ tự (hay chỉ số) của phần tử đó.

Ví dụ, giả sử ta định nghĩa mảng sau:

```
myArray = new Array ("Wind", "Rain", "Fire")
```

Sau đó ta sẽ tham khảo tới phần tử thứ nhất của mảng bằng cách dùng myArray[0] và phần tử thứ hai của mảng bằng myArray[1].

Ở đây lưu ý rằng: Chỉ mục của các phần tử bắt đầu bằng số 0, nhưng độ dài của mảng phản ánh số phần tử của mảng.

Ngoài cách dùng chỉ số của phần tử mảng, thì ta còn có dùng cách chỉ ra tên của phần tử để truy cập đến phần tử đó.

Ví dụ: myArray["Rain"].

#### 10.2 Các phương thức của mảng

Các phương thức của đối tượng Array được liệt kê và mô tả trong bảng sau:

**Bảng 10.1: Các phương thức của đối tượng mảng**

Phương thức	Mô tả
concat	Nối hai mảng và trả về một mảng mới.
join	Kết hợp tất cả các phần tử của một mảng thành một chuỗi.
pop	Gỡ bỏ phần tử cuối cùng của một mảng và trả về phần tử đó.
push	Bổ sung một hoặc nhiều phần tử vào cuối một mảng và trả về độ dài mới của mảng.
reverse	Hoán vị các phần tử của một mảng: Phần tử mảng đầu tiên trở thành phần tử cuối cùng và ngược lại, phần tử cuối cùng trở thành phần tử đầu tiên.
shift	Gỡ bỏ phần tử đầu tiên của một mảng và trả về phần tử đó.
slice	Trích một phần của một mảng và trả về một mảng mới.
splice	Bổ sung và/hoặc gỡ bỏ các phần tử của một mảng.
sort	Sắp xếp các phần tử của một mảng
toSource	Trả về một mảng chữ đại diện cho mảng chỉ định; ta có thể sử dụng giá trị này để tạo ra một mảng mới. Phương thức này đè lên phương thức Object.toSource.
toString	Trả về một chuỗi đại diện cho mảng và các phần tử của nó. Phương thức này đè lên phương thức Object.toString.
unshift	Bổ sung một hoặc nhiều phần tử vào phía trước của một

	mảng và trả về độ dài mới của mảng.
valueOf	Trả về giá trị nguyên thủy của mảng. Phương thức này đè lên phương thức Object.valueOf.

Ngoài ra, đối tượng này kế thừa các phương thức watch và unwatch từ đối tượng Object.

Dưới đây sẽ phân tích một số phương thức thông dụng trong bảng trên.

### 10.2.1 Phương thức concat

Cú pháp:

concat (arrayName2, arrayName3,..., arrayNameN)

Mô tả:

concat không làm thay đổi các mảng nguyên thủy, nhưng trả về một bản sao bao gồm các bản sao của các phần tử được liên kết từ các mảng nguyên thủy. Các phần tử của các mảng nguyên thủy được sao chép vào trong mảng mới như sau:

- Các tham khảo đối tượng (và không phải đối tượng thực sự): concat sao chép các tham khảo đối tượng vào trong mảng mới. Cả mảng nguyên thủy và mảng mới tham khảo đến cùng đối tượng. Nếu một đối tượng được tham khảo thay đổi, thì sự thay đổi thấy rõ trong cả các mảng mới và mảng nguyên thủy.
- Các chuỗi và các số (không phải các đối tượng String và Number): concat sao chép các chuỗi và các số vào trong mảng mới. Thay đổi chuỗi hoặc số trong một mảng không làm ảnh hưởng đến các mảng khác.

Nếu một phần tử mới được bổ sung vào một trong hai mảng, thì mảng kia không bị ảnh hưởng.

Ví dụ: Mã nguồn sau kết hợp hai mảng:

```
a = new Array ("a", "b", "c")
b = new Array (1,2,3)
ab = a.concat(b);
```

sẽ tạo ra mảng: ["a", "b", "c", 1, 2, 3].

Ví dụ: Mã nguồn sau kết hợp ba mảng:

```
num1 = [1,2,3]
num2 = [4,5,6]
num3 = [7,8,9]
nums = num1.concat(num2,num3)
```

sẽ tạo ra mảng: [1,2,3,4,5,6,7,8,9].

### 10.2.2 Phương thức join

Cú pháp:

join (separator)

Trong đó: separator chỉ ra một chuỗi để ngăn cách mỗi phần tử của một mảng. Dấu ngăn cách được chuyển đổi thành một chuỗi nếu cần thiết. Nếu bỏ qua, thì các phần tử mảng được ngăn cách bởi một dấu phẩy.

Mô tả:

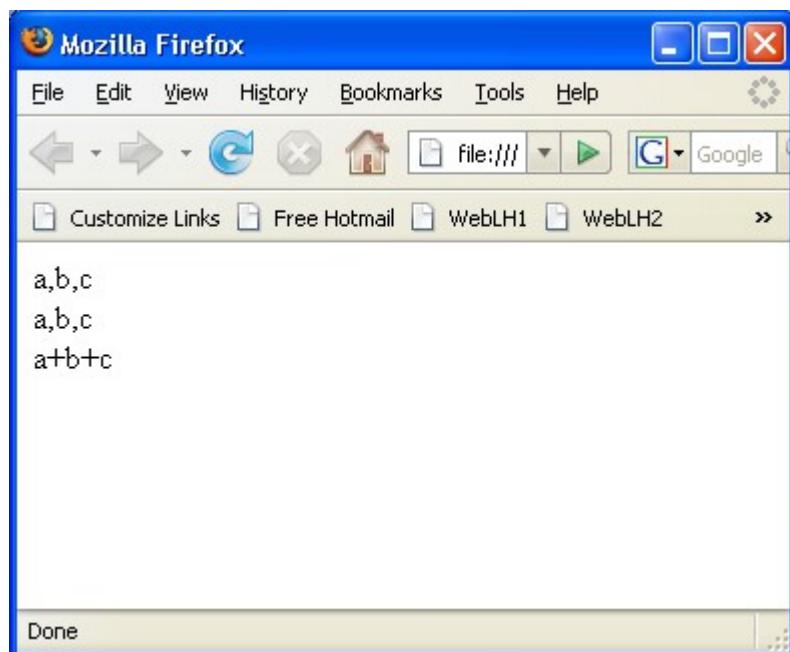
Các chuyển đổi chuỗi của tất cả các phần tử mảng được kết hợp thành một chuỗi.

Ví dụ sau tạo ra một mảng a với ba phần tử, sau đó kết hợp mảng ba lần: sử dụng dấu ngăn cách mặc định, sau đó đến một dấu phẩy và một dấu cách, rồi đến một dấu cộng.

### Ví dụ 10.2:

```
<HTML>
<HEAD>
<SCRIPT>
a = new Array ("a","b","c");
a1=a.join(); // gán "a,b,c" vào a1
a2=a.join(","); // gán "a,b,c" vào a2
a3=a.join("+"); // gán "a+b+c" vào a1
document.writeln(a1 + "<BR>");
document.writeln(a2 + "<BR>");
document.writeln(a3 + "<BR>");
</SCRIPT>
</HEAD>
</HTML>
```

Kết quả:



**Hình 10.2: Minh họa phương thức join**

### **10.2.3 Phương thức pop**

Cú pháp:

pop()

**Ví dụ :** Mã nguồn sau tạo ra mảng myFish gồm có bốn phần tử, sau đó gỡ bỏ phần tử cuối cùng của nó.

myFish = ["angel", "clown", "mandarin", "surgeon"] ;

popped = myFish.pop();

Chương trình sẽ gỡ bỏ phần tử cuối cùng của mảng myFish là “surgeon” và trả về phần tử này. Sau câu lệnh này, mảng myFish chỉ còn lại ba phần tử, chiều dài của mảng sẽ là 3.

### **10.2.4 Phương thức push**

Cú pháp:

push(element1, ..., elementN)

Trong đó: element1, ..., elementN là các phần tử được bổ sung vào cuối mảng.

Mô tả:

Phương thức push bổ sung một hoặc nhiều phần tử vào cuối một mảng và trả về độ dài mới của mảng.

**Ví dụ:** Mã nguồn sau tạo ra mảng myFish gồm có hai phần tử, sau đó bổ sung hai phần tử vào mảng. Sau khi mã nguồn thi hành, pushed chứa bốn phần tử.

myFish = ["angel", "clown"] ;

pushed = myFish.push("mandarin", "surgeon");

### **10.2.5 Phương thức reverse**

Cú pháp:

reverse()

Mô tả:

Phương thức reverse hoán vị các phần tử của đối tượng mảng được gọi.

**Ví dụ 10.3:** Ví dụ sau tạo ra một mảng myArray, gồm có ba phần tử, sau đó hoán vị mảng.

<HTML>

<HEAD>

<SCRIPT>

myArray = new Array(3);

myArray[0] = "One";

myArray[1] = "Two";

myArray[2] = "Three";

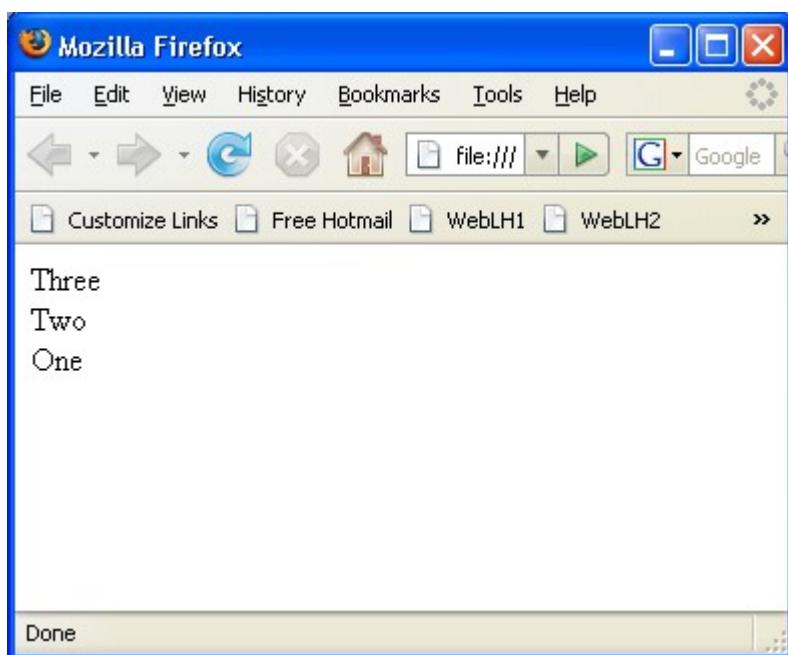
myArray.reverse();

```

        document.writeln(myArray[0] + "<BR>");
        document.writeln(myArray[1] + "<BR>");
        document.writeln(myArray[2] + "<BR>");
    </SCRIPT>
</HEAD>
</HTML>

```

**Kết quả:**



**Hình 10.3: Minh họa hàm reverse**

#### 10.2.6 Phương thức sort

Cú pháp:

sort (compareFunction)

Trong đó: compareFunction chỉ định một hàm định nghĩa thứ tự sắp xếp. Nếu bỏ qua, thì mảng được sắp xếp theo thứ tự từ điển (trong luật từ điển) theo như sự chuyển đổi chuỗi của mỗi phần tử.

Mô tả:

Nếu compareFunction không được cung cấp, thì các phần tử được sắp xếp bằng cách chuyển đổi chúng thành các chuỗi và so sánh các chuỗi theo thứ tự từ điển (không phải thứ tự số học). Ví dụ, “80” xếp trước “9” theo thứ tự từ điển, nhưng trong sắp xếp số học thì 9 xếp trước 80.

Nếu compareFunction được cung cấp, thì các phần tử mảng được sắp xếp tuân theo giá trị trả về của hàm compare. Nếu a và b là hai phần tử đang được so sánh thì:

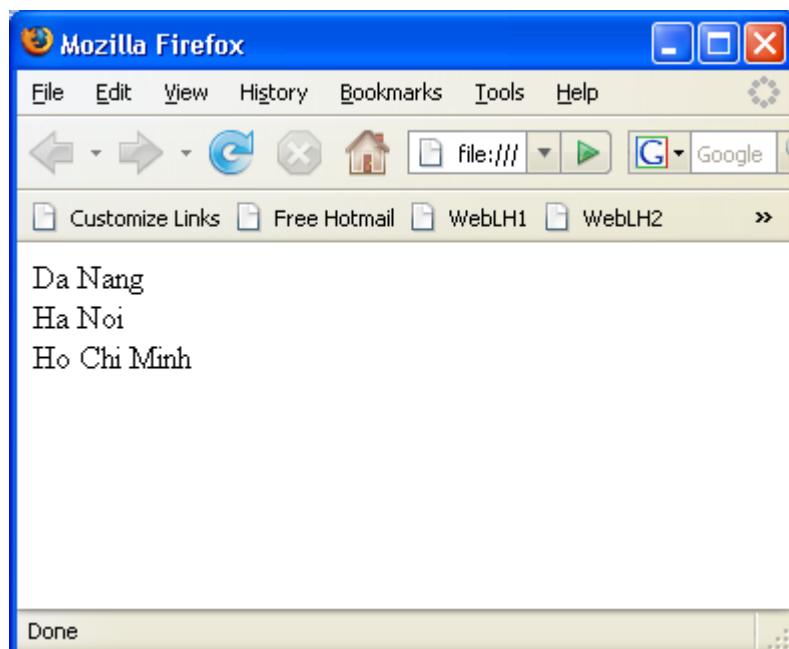
- Nếu compareFunction(a,b) nhỏ hơn 0, thì sắp xếp b có chỉ số nhỏ hơn a.
- Nếu compareFunction(a,b) trả về 0, thì a và b không thay đổi vai trò với nhau, nhưng được sắp xếp như tất cả các phần tử khác.
- Nếu compareFunction(a,b) lớn hơn 0, thì sắp xếp b có chỉ số lớn hơn a.

Ví dụ sau minh họa cho phương thức sort:

#### Ví dụ 10.4:

```
<HTML>
<HEAD>
<SCRIPT>
myArray = new Array(3);
myArray[0] = "Ha Noi";
myArray[1] = "Da Nang";
myArray[2] = "Ho Chi Minh";
myArray.sort();
document.writeln(myArray[0] + "<BR>");
document.writeln(myArray[1] + "<BR>");
document.writeln(myArray[2] + "<BR>");
</SCRIPT>
</HEAD>
</HTML>
```

Kết quả:



Hình 6.4: Các phần tử mảng được liệt kê sau khi đã sắp xếp

#### 10.3 Mảng hai 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.

#### Ví dụ 10.5:

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

&lt;HTML&gt;

&lt;HEAD&gt;

&lt;SCRIPT&gt;

```

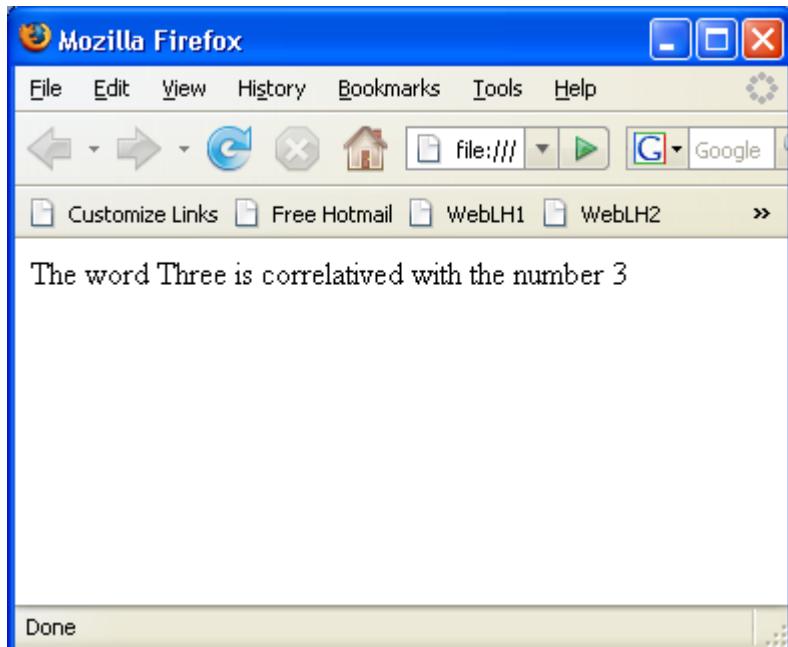
myArray = new Array(3,3);
myArray[0,0] = "One";
myArray[0,1] = 1;
myArray[1,0] = "Two";
myArray[1,1] = 2;
myArray[2,0] = "Three";
myArray[2,1] = 3;
document.write("The word " +myArray[1,0]);
document.write(" is correlated with the number " +myArray[1,1]);

```

&lt;/SCRIPT&gt;

&lt;/HEAD&gt;

&lt;/HTML&gt;

**Kết quả:****Hình 10.5 Ví dụ mảng hai chiều**

Mảng hai chiều còn được coi là mảng của các mảng một chiều. Có nghĩa là mỗi phần tử của mảng một chiều cũng là một mảng một chiều.

Ví dụ: Mã nguồn sau đây tạo ra một mảng hai chiều

```

a = new Array(4)
for (i=0; i<4; i++) {
    a[i] = new Array(4)
}

```

```

for (j=0; j<4; j++) {
    a[i][j] = "[" + i + "," + j + "]"
}

```

Ví dụ này tạo ra một mảng với các dòng như sau:

Row 0: [0,0][0,1][0,2][0,3]

Row 1: [1,0][1,1][1,2][1,3]

Row 2: [2,0][2,1][2,2][2,3]

Row 3: [3,0][3,1][3,2][3,3]

#### 10.4 Câu hỏi và bài tập

1. Mảng được dùng để lưu một tập hợp các biến có cùng \_\_\_\_\_
2. \_\_\_\_\_ của mảng dùng để phân biệt các biến này.
3. Trong JavaScript, chỉ số của mảng bắt đầu từ 0 \_\_\_\_\_ (Đúng/Sai)
4. Ta không thể gán giá trị cho một mảng ngay khi tạo ra \_\_\_\_\_ (Đúng/Sai)
5. \_\_\_\_\_ của mảng phản ánh số phần tử của mảng.
6. Phương thức \_\_\_\_\_ được dùng để sắp xếp các phần tử của một mảng.
7. Phương thức reverse được dùng để \_\_\_\_\_ các phần tử của một mảng.
8. Phương thức \_\_\_\_\_ nối hai mảng và trả về một mảng mới.
9. Phương thức \_\_\_\_\_ gỡ bỏ phần tử cuối cùng của một mảng và trả về phần tử đó.

**Bài tập thực hành chương 10:**

1. Nhập vào các giá trị cho một mảng 15 phần tử. Sau đó in ra các giá trị này.
2. Nhập một mảng 15 phần tử. Hãy in các giá trị của mảng sau khi sắp xếp các giá trị này theo thứ tự tăng dần.

## CHƯƠNG 11

# CÁC ĐỐI TƯỢNG CƠ BẢN CỦA JAVASCRIPT

JavaScript được thiết kế trên mô hình nền tảng đối tượng đơn giản. Một đối tượng là một gói dữ liệu toàn diện. Các thuộc tính (là các biến hoặc các đối tượng khác của JavaScript) dùng để định nghĩa đối tượng và các phương thức (là các hàm kết hợp với đối tượng) tác động tới dữ liệu đều nằm trong đối tượng.

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

objectName.propertyName

Trong đó, cả tên đối tượng (objectName) và tên thuộc tính (propertyName) đều phân biệt chữ hoa và chữ thường. Ta định nghĩa một thuộc tính bằng cách gán cho nó một giá trị.

Ví dụ: Một chiếc xe hơi là một đối tượng. Các thuộc tính của chiếc xe hơi là hãng xe (make), kiểu dáng (model) và màu sắc của xe (color). Hầu hết các chiếc xe hơi đều có một vài phương thức chung như go(), brake(), reverse().

```
myCar.make = "Ford";  
myCar.model = "Mustang";  
myCar.color = "black";
```

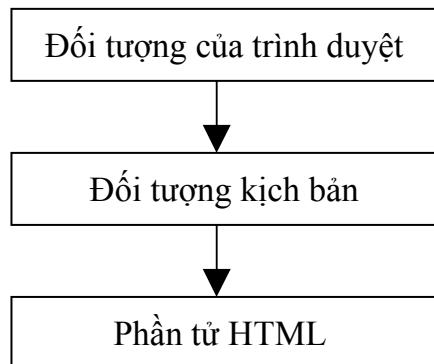
Để truy cập đến các phương thức của một đối tượng, chúng ta phải chỉ ra tên đối tượng và phương thức yêu cầu:

objectName.method()

Khi tạo ra một trang Web, chúng ta có thể chèn:

- Các đối tượng của trình duyệt
- Các đối tượng xây dựng sẵn của ngôn ngữ kịch bản được sử dụng (JavaScript)
- Các phần tử HTML

Đĩ 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 phản tử trong trang. Các đối tượng trình duyệt chẳng hạn như văn bản (document), cửa sổ (window), khung (frame), vị trí (location),... nằm trên cùng của cây phân cấp đối tượng. Sau đó là các đối tượng xây dựng sẵn của JavaScript. Các phản tử HTML nằm ở sau cùng và chính là các thẻ HTML tạo nên văn bản hiện hành.

Trong chương này, chúng ta tìm hiểu về một số đối tượng xây dựng sẵn trong JavaScript:

- *Math*
- *String*
- *Date*

## 11.1 Đối tượng Math

### 11.1.1 Mô tả

Đối tượng Math là một đối tượng được xây dựng sẵn có các phương thức và các thuộc tính cho các hàm và các hằng số toán học.

Math là một đối tượng JavaScript cấp cao. Chúng ta có thể tự động truy xuất nó mà không cần sử dụng một hàm dựng hay gọi một phương thức nào.

Tất cả các thuộc tính và phương thức của đối tượng Math đều tĩnh. Chúng ta tham khảo đến hằng số PI dưới dạng Math.PI và ta gọi hàm sin dưới dạng Math.sin(x), với x là đối số của phương thức. Các hằng số được định nghĩa với độ chính xác cao nhất của các số thực trong JavaScript.

Có thể sử dụng câu lệnh `with` khi một đoạn mã nguồn sử dụng một số hằng số và phương thức Math, vì thế chúng ta không cần phải lặp lại “Math” nhiều lần.

Ví dụ:

Thay vì phải viết:

```

a = Math.PI * r * r
b = r * Math.sin(x)
c = r * Math.cos(x)
  
```

Ta có thể dùng `with`:

`with (Math)`

```

{
    a = PI * r * r
    b= r * sin(x)
    c = r * cos(x)
}

```

### 11.1.2 Các thuộc tính của đối tượng Math

Các thuộc tính của đối tượng Math được tổng kết trong bảng sau:

**Bảng 7.1: Các thuộc tính của đối tượng Math**

Thuộc tính	Mô tả
E	Hằng số Euler và cơ số của các logarithm tự nhiên, xấp xỉ 2.718.
LN2	Logarithm tự nhiên của 2, xấp xỉ 0.693.
LN10	Logarithm tự nhiên của 10, xấp xỉ 2.302.
LOG2E	Logarithm cơ số 2 của E, xấp xỉ 1.442.
LOG10E	Logarithm cơ số 10 của E, xấp xỉ 0.434.
PI	Tỉ lệ của chu vi một đường tròn với đường kính của nó, xấp xỉ 3.14159.
SQRT1_2	Căn bậc hai của $\frac{1}{2}$ , tương đương 1 trên căn bậc hai của 2, xấp xỉ 0.707.
SQRT2	Căn bậc hai của 2, xấp xỉ 1.414.

#### Ví dụ 11.1:

Ví dụ sau tính diện tích của một đường tròn với bán kính do người dùng nhập từ bàn phím:

```

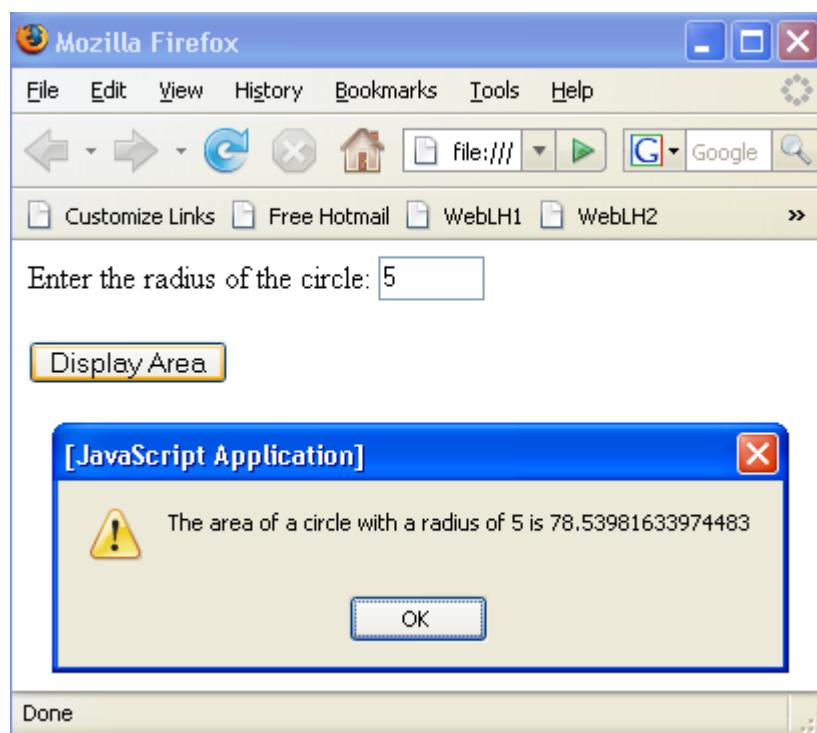
<HTML>
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
function doCal (x)
{
    var a;
    a = Math.PI * x * x;
    alert ("The area of a circle with a radius of " + x + " is " + a);
}
</SCRIPT>
</HEAD>
<BODY>
<FORM>

```

Enter the radius of the circle:

```
<INPUT TYPE = TEXT size = 5 name = "rad">
<BR><BR>
<INPUT TYPE = button value = "Display Area" onclick =
"doCal(rad.value)">
</FORM>
</BODY>
</HTML>
```

**Kết quả:**



**Hình 11.2: Kết quả tính diện tích của đường tròn với bán kính bằng 5**

### 11.1.3 Các phương thức của đối tượng Math

Các phương thức của đối tượng Math được tổng kết trong bảng sau:

**Bảng 7.2: Các phương thức của đối tượng Math**

Phương thức	Mô tả
abs	Trả về giá trị tuyệt đối của một số.
acos	Trả về arccos (theo radian) của một số.
asin	Trả về arcsin (theo radian) của một số.
atan	Trả về arctang (theo radian) của một số.
atan2	Trả về arctang của thương số của các đối số của nó.
ceil	Trả về số nguyên nhỏ nhất lớn hơn hoặc bằng một số.
cos	Trả về cos của một số.

exp	Trả về $E^{\text{number}}$ , với number là đối số, và E là hằng số Euler, cơ số của logarithm tự nhiên.
floor	Trả về số nguyên lớn nhất nhỏ hơn hoặc bằng một số.
log	Trả về logarithm tự nhiên (cơ số E) của một số.
max	Trả về số lớn hơn trong hai số.
min	Trả về số nhỏ hơn trong hai số.
pow	Trả về cơ số lũy thừa số mũ, đó là baseexponent.
random	Trả về một số giả ngẫu nhiên giữa 0 và 1.
round	Trả về giá trị của một số được làm tròn đến số nguyên gần nhất.
sin	Trả về sin của một số.
sqrt	Trả về căn bậc hai của một số
tan	Trả về tan của một số.

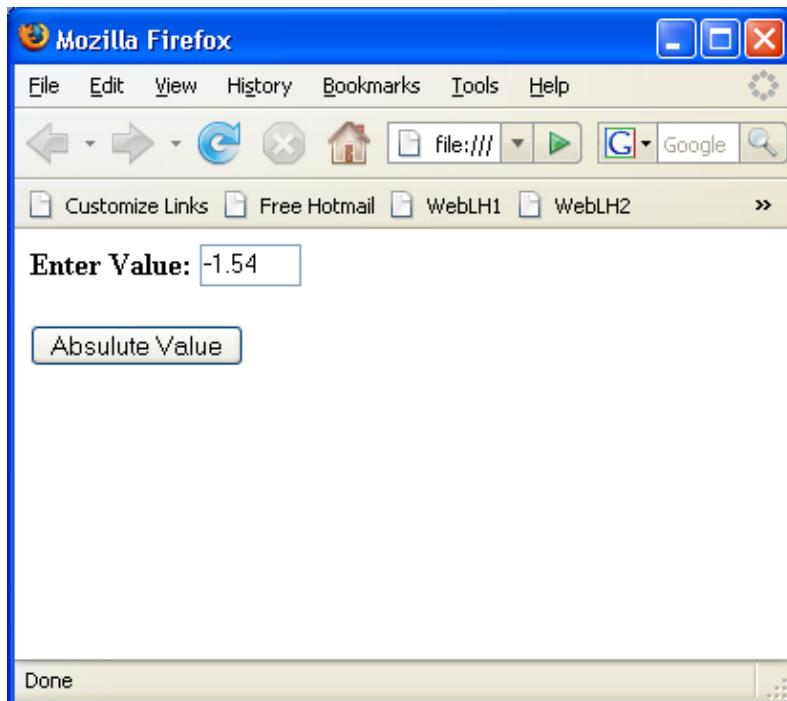
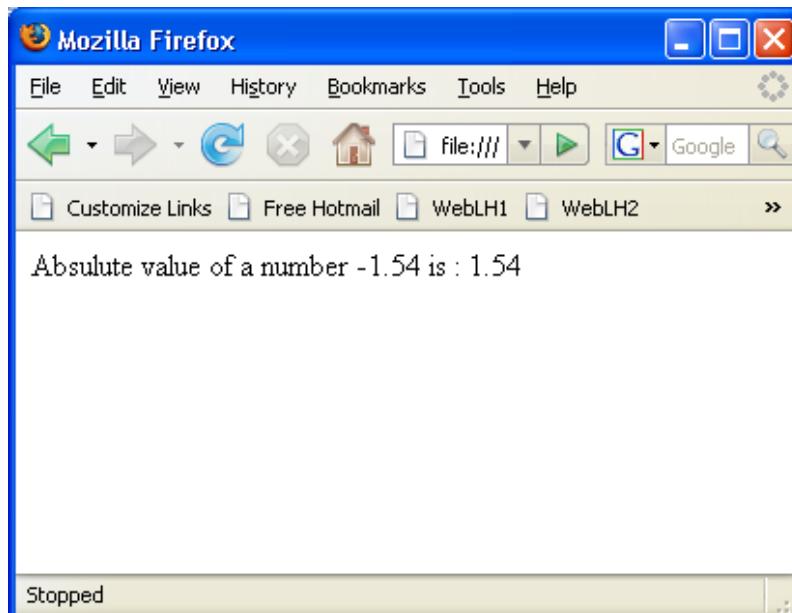
**Ví dụ 11.2:**

Ví dụ sau trả về giá trị tuyệt đối của một số:

```
<HTML>
<HEAD>
<SCRIPT LANGUAGE = "JavaScript">
function show (value)
{
    with (Math)
    {
        document.write("Absolute value of a number " + value + " is : "
+abs(eval(value)));
    }
}
</SCRIPT>
</HEAD>
<BODY>
<FORM>
<B>Enter Value:</B>
<INPUT TYPE = TEXT size = 5 name = "absofvalue">
<BR><BR>
<INPUT TYPE = button value = "Absolute Value" onclick = "show
(this.form.absofvalue.value)">
</FORM>
```

&lt;/BODY&gt;

&lt;/HTML&gt;

**Kết quả:****Hình 11.3.1: Nhập giá trị****Hình 11.3.2: Kết quả trả về trị tuyệt đối của giá trị đó**

## 11.2 Đối tượng String

### 11.2.1 Mô tả

Đối tượng String được dùng để thao tác và làm việc với chuỗi văn bản. Chúng ta có thể tách nó thành các chuỗi con và biến đổi chuỗi đó thành các chuỗi chữ hoa hoặc chữ thường trong một chương trình.

Cú pháp tổng quát để truy cập đến một thuộc tính hoặc một phương thức của một đối tượng String như sau:

stringName.propertyName

và:

stringName.methodName

Ở đây chúng ta lưu ý đừng nhầm lẫn giữa một chuỗi chữ với một đối tượng String, bởi vì đối tượng String là một trình bao bọc xung quanh kiểu dữ liệu chuỗi nguyên thủy.

Ví dụ, mã nguồn sau đây tạo ra một chuỗi chữ s1 và cũng tạo ra đối tượng String s2:

s1 = "foo" // tạo ra một chuỗi ký tự

s2 = new String ("foo") // tạo ra một đối tượng String.

Ta có thể gọi bất cứ phương thức nào của đối tượng String trên giá trị chuỗi chữ - JavaScript tự động chuyển đổi chuỗi chữ thành một đối tượng String tạm thời, gọi phương thức, sau đó loại bỏ đối tượng String tạm thời. Ta cũng có thể sử dụng thuộc tính String.length với một chuỗi chữ.

Chúng ta nên sử dụng chuỗi chữ nếu không cần sử dụng đối tượng String một cách cụ thể, vì các đối tượng String có thể tác động khác thường.

Xét ví dụ sau:

s1 = "2 + 2" // tạo ra một giá trị cho chuỗi ký tự

s2 = new String ("2 + 2") // tạo ra một đối tượng String

eval(s1) // trả về số 4

eval(s2) // trả về chuỗi "2 + 2"

### ***11.2.2 Các thuộc tính của đối tượng String***

Đối tượng String có một thuộc tính duy nhất là thuộc tính length. Thuộc tính này cho biết số ký tự trong một chuỗi.

Ví dụ, mã nguồn sau sẽ gán giá trị 13 cho biến strlen, vì chuỗi myString có chứa 13 ký tự:

myString = "Hello, World!"

strlen = myString.length

Thuộc tính length của một đối tượng String (hay một chuỗi) sẽ được tự động cập nhật (thay đổi giá trị) khi chuỗi thay đổi, và người dùng không được quyền thiết đặt giá trị này.

### ***11.2.3 Các phương thức của đối tượng String***

Đối tượng String có hai kiểu phương thức: một kiểu trả về sự biến đổi trên bản thân một chuỗi, chẳng hạn như phương thức substring() và toUpperCase() hay toLowerCase(), và kiểu kia trả về một chuỗi có dạng HTML, chẳng hạn như phương thức bold() và link().

Ví dụ, nếu sử dụng chuỗi myString đã định nghĩa ở trên, cả hai câu lệnh myString.toUpperCase() và câu lệnh “hello, world!”.toUpperCase() đều sẽ trả về chuỗi “HELLO, WORLD!”.

Phương thức substring có hai đối số và trả về một chuỗi con của chuỗi giữa hai đối số. Sử dụng ví dụ trên, myString.substring(4,9) trả về chuỗi “o, Wo”.

Như đã nói ở trên, đối tượng String cũng có một số phương thức tự động định dạng HTML, chẳng hạn như bold để tạo ra văn bản kiểu chữ đậm và link để tạo ra siêu liên kết.

Ví dụ, chúng ta có thể tạo ra một siêu liên kết tới một URL với phương thức link như sau:

```
myString.link("http://www.helloworld.com")
```

Bảng sau đây tóm tắt các phương thức thông dụng của đối tượng String:

**Bảng 7.3: Các phương thức của đối tượng String**

Phương thức	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
concat	Nối hai chuỗi và trả về chuỗi mới
fontcolor	Xác định màu của font chữ
italics	Hiển thị chuỗi ở dạng in nghiêng
link	Tạo ra siêu liên kết HTML
small	Giảm kích thước của chuỗi văn bản
strike	Hiển thị chuỗi có đường gạch ngang nằm giữa (ví dụ: <code>strikethrough</code> )
sub	Hiển thị văn bản dưới dạng chỉ số dưới
substring, substr	Trả về chuỗi con cụ thể của một chuỗi, bằng cách chỉ ra chỉ số đầu và chỉ số cuối, hoặc chỉ số đầu và độ dài chuỗi con.
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 minh họa một vài phương thức của đối tượng String và công dụng của chúng:

### Ví dụ 11.3:

```
<HTML>
```

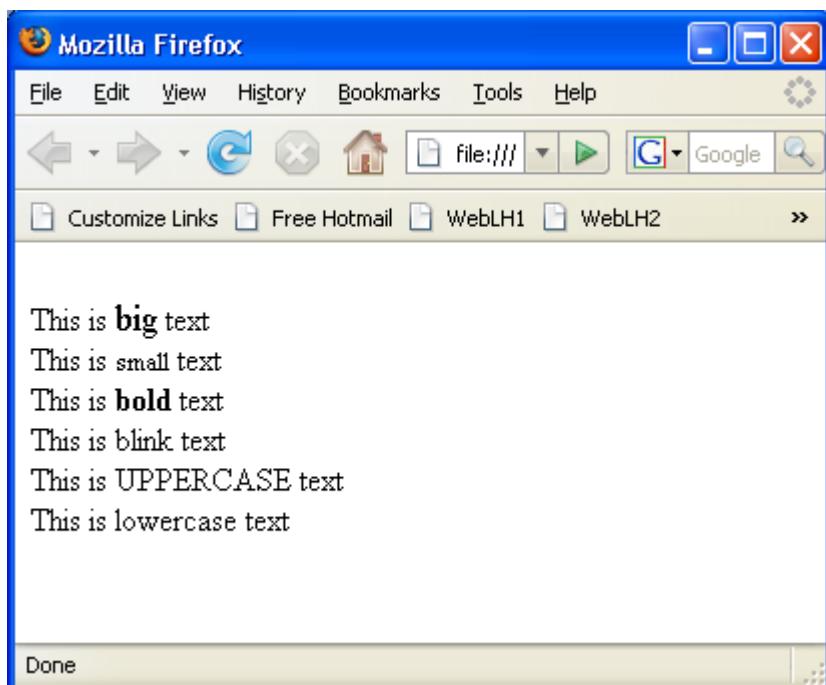
```
<HEAD>
```

```

<SCRIPT LANGUAGE = "JavaScript">
    var a = "big";
    var b = "small";
    var c = "bold";
    var d = "blink";
    var e = "UpperCase";
    var f = "LowerCase";
    document.write ("<BR>This is "+ a.big() + " text");
    document.write ("<BR>This is "+ b.small() + " text");
    document.write ("<BR>This is "+ c.bold() + " text");
    document.write ("<BR>This is "+ d.blink() + " text");
    document.write ("<BR>This is "+ e.toUpperCase() + " text");
    document.write ("<BR>This is "+ f.toLowerCase() + " text");
</SCRIPT>
</HEAD>
</HTML>

```

**Kết quả:**



**Hình 11.4: Minh họa một số phương thức của đối tượng String**

#### 11.2.4 Tìm kiếm trong một chuỗi

Chúng ta có thể sử dụng hàm search() của đối tượng String để tìm kiếm sự xuất hiện của một đoạn văn bản trong chuỗi. Tham số cho hàm này là chuỗi mà ta cần tìm. Hàm search() trả lại chỉ số vị trí của chuỗi tìm kiếm. Nếu không tìm thấy, hàm sẽ trả về giá trị -1.

Ví dụ dưới đây minh họa công dụng của hàm này:

#### Ví dụ 11.4:

<HTML>

<HEAD>

```
<title>Tim kiem trong chuoi </title>
<script language=javascript>
str1 = "Đây là kết thúc của một dòng.";
document.write(str1);
document.write("<BR>");
document.write("Vị trí của từ 'kết' là " +str1.search('kết'));
</script>
```

</HEAD>

</HTML>

#### Kết quả:

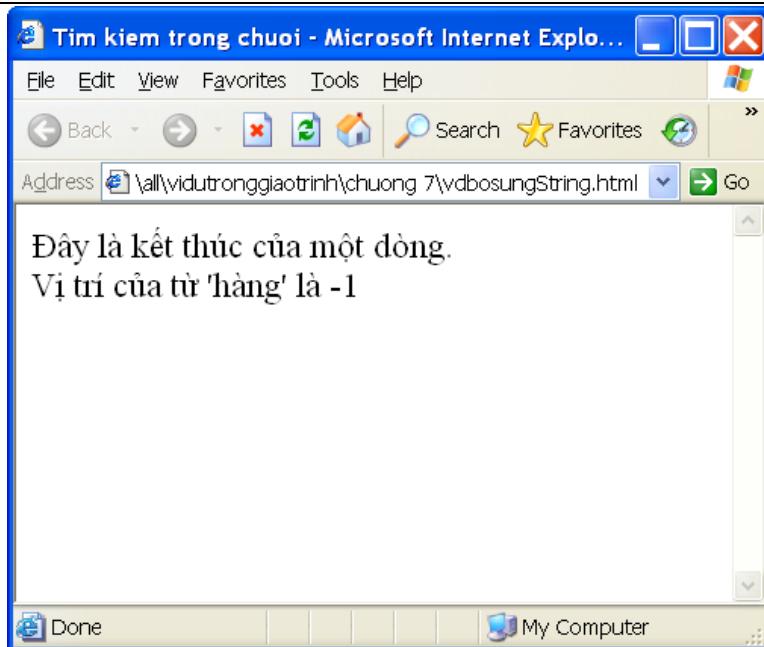


**Hình 11.5.1: Minh họa hàm search()**

Đoạn mã sau thay đổi tham số của hàm search() để tìm một chuỗi con không có trong chuỗi:

```
str1 = "Đây là kết thúc của một dòng.";
document.write(str1);
document.write("<BR>");
document.write( " Vị trí của từ 'hàng' là " +str1.search('hàng'));
```

#### Kết quả:



**Hình 11.5.2: Minh họa hàm search() khi không tìm ra chuỗi con**

### 11.2.5 Định vị các ký tự trong một chuỗi

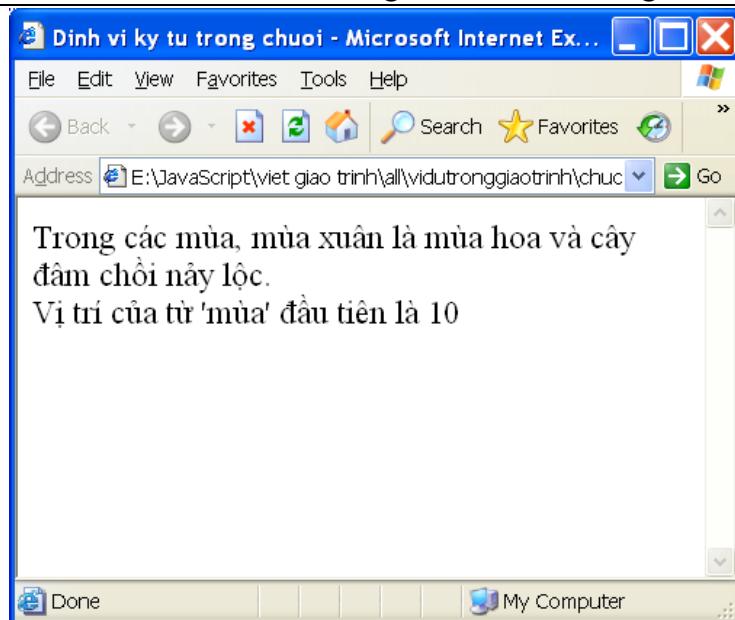
Chúng ta có thể dùng hàm indexOf() (tương tự như search()) để xác định vị trí xuất hiện đầu tiên của một chuỗi con hoặc một ký tự trong một chuỗi, ngoài ra ta còn có thể xác định nơi mà indexOf() bắt đầu thực hiện việc tìm kiếm. Ta cũng có thể tìm bắt đầu từ cuối chuỗi bằng cách dùng hàm lastIndexOf(). Nếu cung cấp tham số thứ hai cho hàm này, nó sẽ tìm kiếm ngược về đầu chuỗi bắt đầu tại vị trí được xác định bởi tham số thứ hai.

Ta xét các ví dụ sau để hiểu rõ hơn về hai hàm này:

**Ví dụ 11.5:** Sử dụng hàm indexOf() để xác định vị trí xuất hiện đầu tiên của chuỗi con ‘mùa’ trong chuỗi “Trong các mùa, mùa xuân là mùa hoa và cây đâm chồi nảy lộc”

```
<HTML>
<HEAD>
    <title>Dinh vi ky tu trong chuoi </title>
    <script language=javascript>
        str1 = "Trong các mùa, mùa xuân là mùa hoa và cây đâm chồi
nảy lộc.";
        document.write(str1);
        document.write("<BR>");
        document.write("Vị trí của từ 'mùa' đầu tiên là "
+str1.indexOf('mùa'));
    </script>
</HEAD>
</HTML>
```

**Kết quả:**

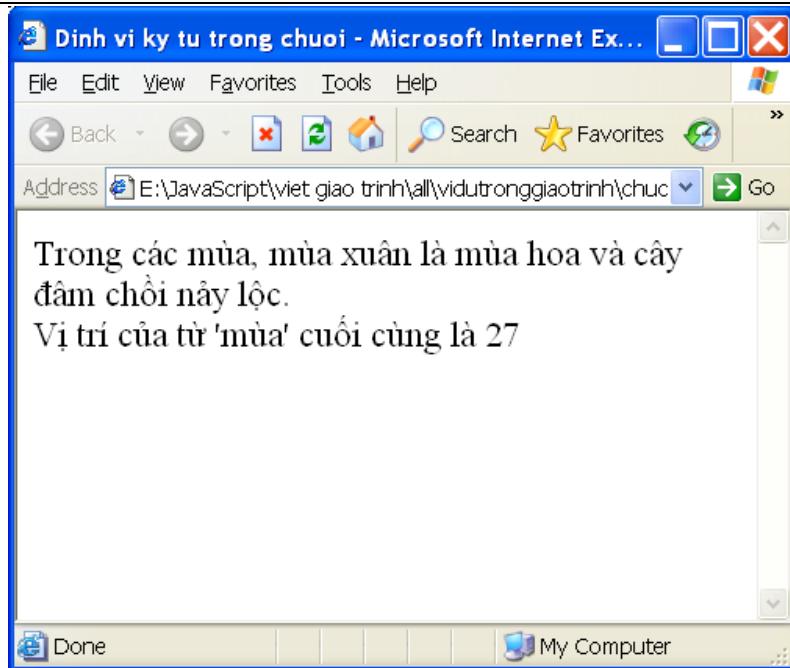


**Hình 11.6: Minh họa hàm indexOf()**

**Ví dụ 11.6:** Sử dụng hàm lastIndexOf() để xác định vị trí xuất hiện cuối cùng của chuỗi con ‘mùa’ trong chuỗi “Trong các mùa, mùa xuân là mùa hoa và cây đâm chồi này lộc”

```
<HTML>
<HEAD>
    <title>Dinh vi ky tu trong chuoi </title>
    <script language=javascript>
        str1 = "Trong các mùa, mùa xuân là mùa hoa và cây đâm chồi
        này lộc.";
        document.write(str1);
        document.write("<BR>");
        document.write("Vị trí của từ 'mùa' cuối cùng là "
        +str1.lastIndexOf('mùa')));
    </script>
</HEAD>
</HTML>
```

**Kết quả:**



**Hình 11.7: Minh họa hàm lastIndexOf()**

### 11.3 Đối tượng Date

#### 11.3.1 Mô tả

JavaScript không có kiểu dữ liệu ngày tháng. Tuy nhiên, chúng ta có thể sử dụng đối tượng Date và các phương thức của nó để làm việc với ngày tháng, thời gian trong các ứng dụng. Đối tượng Date có nhiều phương thức cho việc thiết lập, nhận và thao tác ngày tháng. Nó không có bất kỳ thuộc tính nào.

JavaScript xử lý ngày tháng tương tự Java. Hai ngôn ngữ này có nhiều phương thức ngày tháng giống nhau, và cả hai ngôn ngữ lưu trữ ngày tháng bằng số miligiây kể từ 00:00:00, ngày 1 tháng 1 năm 1970.

Phạm vi của đối tượng Date là từ -100,000,000 ngày tới 100,000,000 ngày liên quan tới ngày 01 tháng 1 năm 1970 UTC.

Để tạo ra một đối tượng Date, ta dùng cú pháp sau:

```
dateObjectName = new Date ([parameters])
```

Ở đây dateObjectName là tên của đối tượng Date sẽ được tạo ra, nó có thể là đối tượng mới hoặc thuộc tính của một đối tượng sẵn có.

Các parameters trong cú pháp sẵn có có thể là:

- Không có đối số: tạo ra ngày tháng và thời gian của một ngày.

Ví dụ: today = new Date()

- Một chuỗi mô tả ngày tháng có dạng như sau: “Tháng ngày, năm giờ:phút:giây”.

Ví dụ: Xmas95 = new Date(“December 25,1995 13:30:00”).

Nếu bỏ qua giờ, phút và giây, thì giá trị sẽ được thiết lập là 0.

- Một tập hợp các giá trị số nguyên cho năm, tháng, ngày.

Ví dụ: Xmas95 = new Date (1995,11,25)

Một tập hợp các giá trị số nguyên cho năm, tháng, ngày, giờ, phút và giây.

Ví dụ: Xmas95 = new Date(1995,11,25,9,30,0).

### **11.3.2 Các nhóm phương thức của đối tượng Date**

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

**Bảng 7.4: Các nhóm phương thức của đối tượng Date**

Nhóm phương thức	Mô tả
set	Gồm những phương thức được 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.

Với các phương thức “get” và “set”, ta có thể nhận và thiết lập giây, phút, giờ, ngày của tháng, ngày của tuần, các tháng và các năm riêng biệt. Có phương thức getDay trả về ngày của tuần, nhưng không có phương thức setDay tương ứng, bởi vì ngày của tuần được thiết lập tự động. Các phương thức sử dụng các số nguyên để mô tả các giá trị này như sau:

- Giây và phút: 0 đến 59.
- Giờ: 0 đến 23.
- Ngày (trong tuần): 0 (chủ nhật) đến 6 (thứ 7).
- Ngày (trong tháng): 1 đến 31.
- Tháng: 0 (tháng 1) đến 11 (tháng 12).
- Năm: từ 1900 trở đi.

Ví dụ, giả sử ta định nghĩa ngày tháng sau:

Xmas95 = new Date("December 25, 1995")

Thì Xmas95.getMonth() sẽ trả về giá trị 11, và Xmas95.getFullYear() sẽ trả về năm 1995.

Phản tiếp theo chúng ta sẽ tìm hiểu về các phương thức trong từng nhóm phương thức của đối tượng Date.

### **11.3.3 Các phương thức của đối tượng Date**

#### **11.3.3.1 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 .
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).

#### 11.3.3.2 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 giờ cho đối tượng Date (1-12).
setYear	Thiết lập năm cho đối tượng Date, năm phải lớn hơn 1900 (năm (-) 1900).

#### 11.3.3.3 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.

#### 11.3.3.4 Nhóm phương thức parse và 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 11.7:

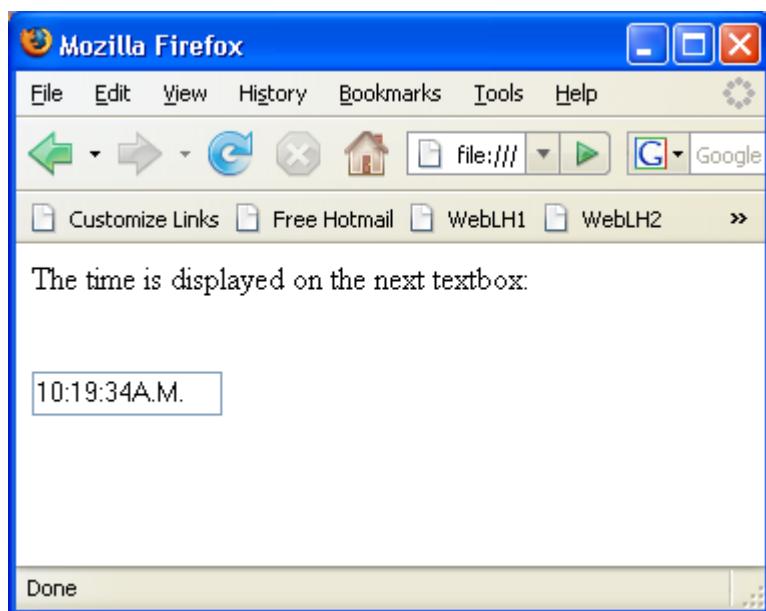
```
<HTML>
<HEAD>
<script>
function dispTime() {
    var time = new Date()
    ...
}
```

```

        var hour = time.getHours()
        var min = time.getMinutes()
        var sec = time.getSeconds()
        var temp = "" + ((hour>12) ? hour -12:hour)
        temp +=((min <10)? ":0" :":") + min
        temp +=((sec <10)? ":0" :":") + sec
        temp +=(hour>=12)? "P.M." :"A.M."
        document.MyPage.time.value = temp
    }
</script>
</HEAD>
<BODY onload="disptime()">
<P> The time is displayed on the next textbox:
<BR><BR>
<FORM Name = "MyPage">
    <INPUT Type = "text" Name="time" size =12 value = "" >
</FORM>
</BODY>
</HTML>

```

**Kết quả:**



**Hình 11.8 Minh họa đối tượng Date**

#### 11.4 Câu hỏi và bài tập

- Để truy cập đến các thuộc tính của đối tượng, chúng ta phải chỉ ra \_\_\_\_\_ đối tượng và \_\_\_\_\_ của nó.

2. Đối tượng String có một thuộc tính duy nhất là thuộc tính \_\_\_\_\_.  
Thuộc tính này cho biết \_\_\_\_\_ trong một chuỗi.
3. Người dùng được quyền thiết đặt giá trị cho thuộc tính length \_\_\_\_\_.  
(Đúng/Sai)
4. Nhóm phương thức set gồm những phương thức được dùng để \_\_\_\_\_ các giá trị thời gian.
5. Nhóm phương thức get gồm những phương thức được dùng để \_\_\_\_\_ các giá trị thời gian.
6. Nhóm phương thức parse và UTC gồm những phương thức được dùng để \_\_\_\_\_ các chuỗi.
7. Phương thức getDate sẽ trả về ngày trong tuần từ đối tượng Date (0-6).  
\_\_\_\_\_ (Đúng/Sai)
8. Phương thức setDay được dùng để thiết lập ngày trong tuần cho đối tượng Date. \_\_\_\_\_ (Đúng/Sai)

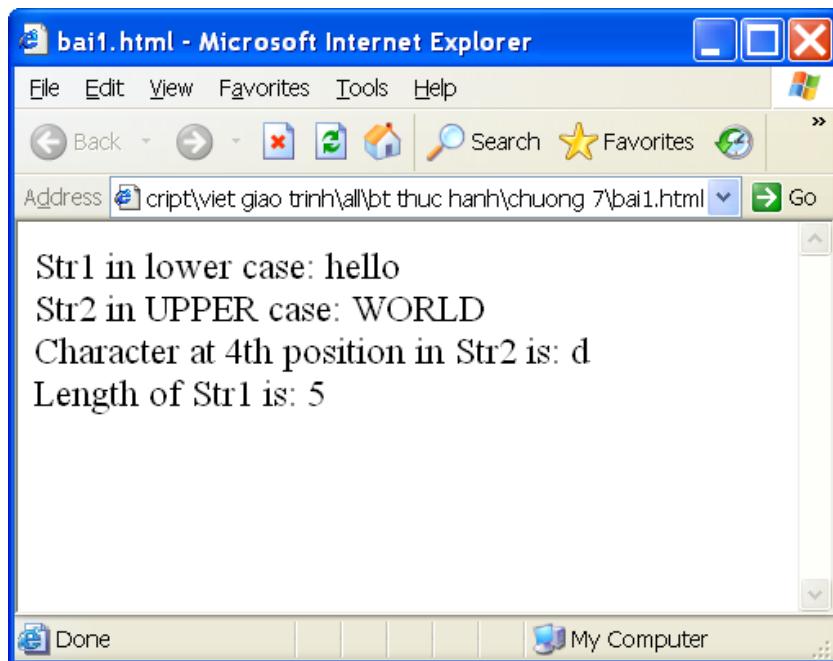
**Bài tập thực hành chương 11:**

1. Cho dòng khai báo sau:

```
var str1 = new String ("HELLO");
```

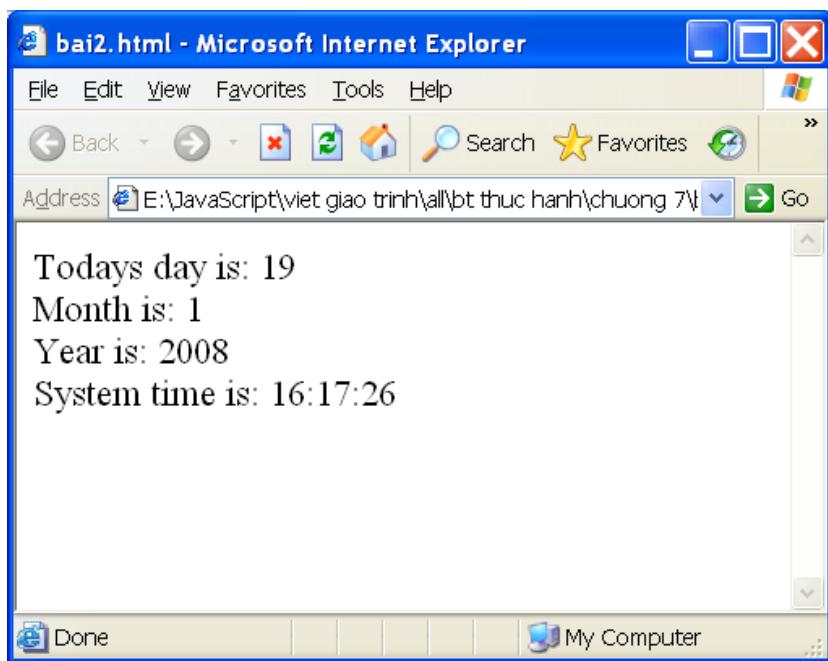
```
var str2 = new String ("world");
```

Hãy viết chương trình đầy đủ để thực hiện các công việc như trong hình sau:



Ghi ý: dùng các phương thức của đối tượng String.

2. Viết chương trình lấy các thông số ngày, tháng, năm và giờ của hệ thống. Kết quả chạy chương trình sẽ như trong hình sau:



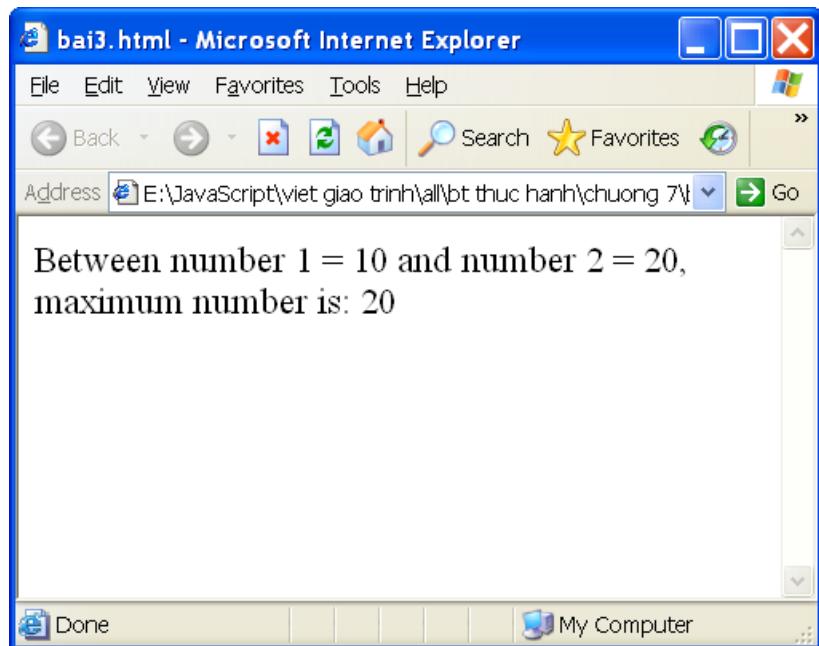
Ghi ý: dùng các phương thức của đối tượng Date.

3. Cho dòng khai báo sau:

num1=10

num2=20

Viết chương trình để hiển thị dòng chữ như trong hình: (sử dụng phương thức của hàm Math)



4. Nhập vào một số là bán kính của đường tròn. Hãy tính chu vi và bán kính của đường tròn đó.
5. In ra câu chào “Chào buổi sáng. Bây giờ là ....” nếu giờ hệ thống < 12. Ngược lại nếu giờ >12 thì in câu “Chào buổi chiều. Bây giờ là ....”

## CHƯƠNG 12

# XỬ LÝ FORM VÀ CÁC SỰ KIỆN CHO CÁC PHẦN TỬ TRÊN FORM

## 12.1 Giới thiệu về đối tượng form

### 12.1.1 Mô tả đối tượng

Đối tượng form (biểu mẫu) cho phép người sử dụng nhập vào văn bản và tạo ra các lựa chọn từ các phần tử form như các hộp chọn (checkbox), các nút bấm radio và các danh sách lựa chọn (selection lists). Ta cũng có thể sử dụng một biểu mẫu để gửi dữ liệu đến một server.

Đối tượng form được tạo ra bởi thẻ FORM của HTML. Công cụ thi hành JavaScript tạo ra một đối tượng form cho mỗi thẻ FORM trong tài liệu. Ta truy xuất các đối tượng FORM thông qua thuộc tính document.forms và thông qua các thuộc tính chỉ định của đối tượng đó.

Để định nghĩa một biểu mẫu, ta sử dụng cú pháp HTML chuẩn với sự bổ sung các trình xử lý sự kiện của JavaScript. Nếu ta cung cấp một giá trị cho thuộc tính NAME, thì ta có thể sử dụng giá trị đó để chỉ mục trong mảng forms. Ngoài ra, đối tượng document được kết hợp có một thuộc tính chỉ định cho mỗi biểu mẫu chỉ định.

Mỗi biểu mẫu trong một tài liệu là một đối tượng riêng biệt. Ta có thể tham khảo đến các phần tử của một biểu mẫu trong mã nguồn của mình bằng cách sử dụng tên của phần tử (từ thuộc tính NAME) hoặc mảng Form.elements. Mảng elements chứa một mục nhập cho mỗi phần tử (như một đối tượng Checkbox, Radio hoặc Text chẳng hạn) trong một biểu mẫu.

Nếu nhiều đối tượng trên cùng biểu mẫu có cùng thuộc tính NAME, thì một mảng tên đã cho được tự động tạo ra. Mỗi phần tử trong mảng đại diện cho một đối tượng Form riêng biệt. Các phần tử được chỉ mục theo thứ tự gốc bắt đầu từ 0. Ví dụ, nếu hai phần tử Text và một phần tử Textarea trên cùng biểu mẫu có thuộc tính NAME của chúng được thiết lập là “myField”, thì một mảng với các phần tử myField[0], myField[1] và myField[2] được tạo ra. Ta cần biết trạng thái này trong mã nguồn của mình và biết myField tham khảo đến một phần tử duy nhất hay đến một mảng các phần tử.

### 12.1.2 Các thuộc tính và phương thức của đối tượng form

Các thuộc tính của đối tượng form được trình bày trong bảng sau:

**Bảng 9.1: Các thuộc tính của đối tượng form**

Thuộc tính	Mô tả
action	Thuộc tính này chỉ định vị trí của script sẽ được dùng để xử lý form được hoàn thành và gửi (submit)
elements	Một mảng phản ánh tất cả các phần tử trong một biểu mẫu.
encoding	Phản ánh thuộc tính ENCTYPE

length	Phản ánh số các phần tử trên một biểu mẫu.
method	Thuộc tính này chỉ định phương thức mà dữ liệu sẽ được gửi đến server.
name	Phản ánh thuộc tính NAME.
target	Phản ánh thuộc tính TARGET.

Các phương thức của đối tượng form được trình bày trong bảng sau:

**Bảng 9.2: Các phương thức của đối tượng form**

Phương thức	Mô tả
handleEvent	Gọi trình xử lý cho sự kiện được chỉ định.
reset	Mô phỏng việc kích chuột trên một nút bấm reset cho biểu mẫu gọi.
submit	Đệ trình một biểu mẫu.

### Ví dụ 12.1:

Ví dụ sau mô tả một form đơn giản với hai nút *Submit* và *Reset*. Khi người dùng nhấn vào nút *Submit* thì sẽ liên kết đến một file khác (file *Simple.html*) và hiển thị giao diện của file này. Đây là hai nút mà ta thường thấy trong các form, thường được dùng để gửi thông tin đã nhập đi (*Submit*), hoặc để xóa các thông tin đã nhập để nhập lại (*Reset*).

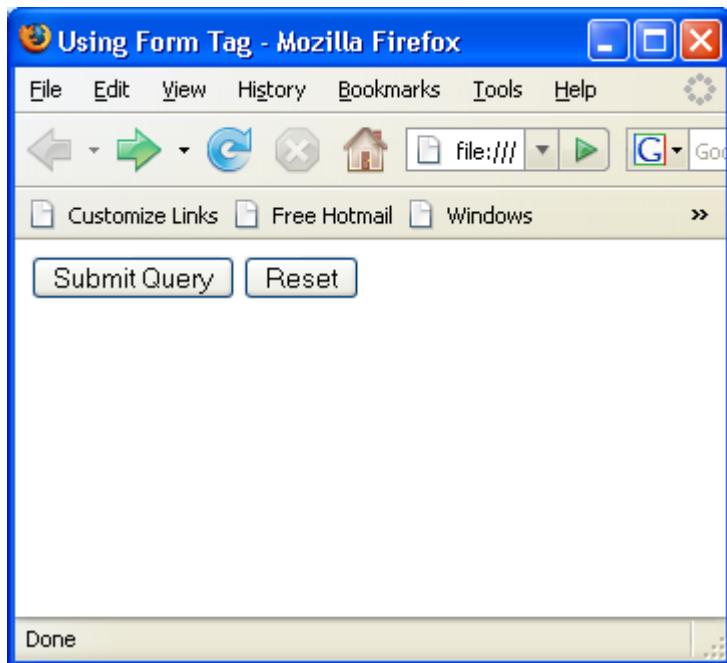
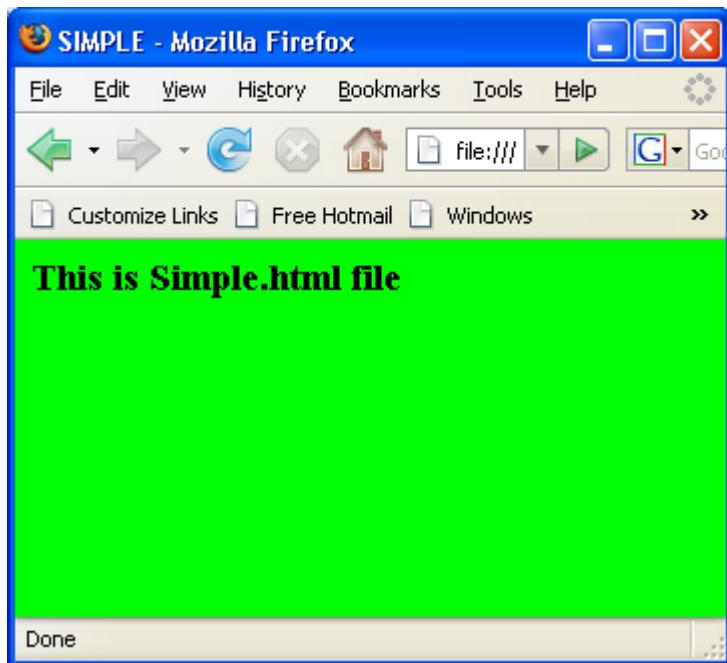
```
<HTML>
<HEAD>
    <TITLE> Using Form Tag </TITLE>
</HEAD>
<BODY BGCOLOR = "#FFFFFF">
    <FORM ACTION = "Simple.html">
        <INPUT TYPE = "submit" NAME= "Submit">
        <INPUT TYPE = "reset" NAME= "Reset">
    </FORM>
</BODY>
</HTML>
```

*File Simple.html:*

```
<HTML>
<HEAD>
    <TITLE> SIMPLE </TITLE>
</HEAD>
<BODY BGCOLOR = "#00FF00">
    <H3> This is Simple.html file </H3>
```

&lt;/BODY&gt;

&lt;/HTML&gt;

**Kết quả:****Hình 12.1.1: Kết quả ví dụ 9.1****Hình 9.1.2: Sau khi nhấn nút Submit**

## 12.2 Xử lý sự kiện trong JavaScript

### 12.2.1 Khái niệm về sự kiện và trình xử lý sự kiện

Các sự kiện là hành động do người dùng tác động sinh ra. Chúng ta có thể làm cho trang web dễ tương tác hơn bằng cách tạo ra các trình xử lý sự kiện đáp ứng các sự kiện do người dùng tạo ra. Trong phần này, chúng ta sẽ tìm hiểu về các sự kiện mà trình duyệt hỗ trợ và cách tạo ra các trình xử lý sự kiện cho các sự kiện này.

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 – như kích chuột vào một nút (button) – hoặc do hệ thống tạo ra – như 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 hỗ 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.

### **12.2.2 Các sự kiện JavaScript phổ biến**

Sau đây là một số sự kiện JavaScript phổ biến 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 9.2:**

Ví dụ sau minh họa sự kiện onClick. Trong ví dụ này, người dùng sẽ nhập một biểu thức vào ô textbox *expr*, sau khi bấm nút *Calculate* thì trên màn hình sẽ xuất hiện một hộp thoại yêu cầu người dùng xác nhận có thực hiện biểu thức vừa nhập hay không. Nếu đồng ý thì kết quả của biểu thức sẽ được hiển thị, nếu không thì sẽ hiển thị thông báo “Please come back again”.

<HTML>

&lt;HEAD&gt;

```
<TITLE> onClick example </TITLE>
<SCRIPT LANGUAGE = "JavaScript">
function compute(form)
{
if(confirm ("Are you sure?"))
{
form.ketqua.value = eval(form.expr.value)
}
else
{
alert("Please come back again.")
}
}
</SCRIPT>
```

&lt;/HEAD&gt;

&lt;BODY&gt;

&lt;FORM&gt;

Enter the expression:

&lt;INPUT TYPE="text" NAME ="expr" SIZE =15&gt;

&lt;BR&gt;

&lt;BR&gt;

&lt;INPUT TYPE = "button" VALUE = "Calculate" onClick=
"compute(this.form)"&gt;

&lt;BR&gt;

&lt;BR&gt;

&lt;BR&gt;

The result is:

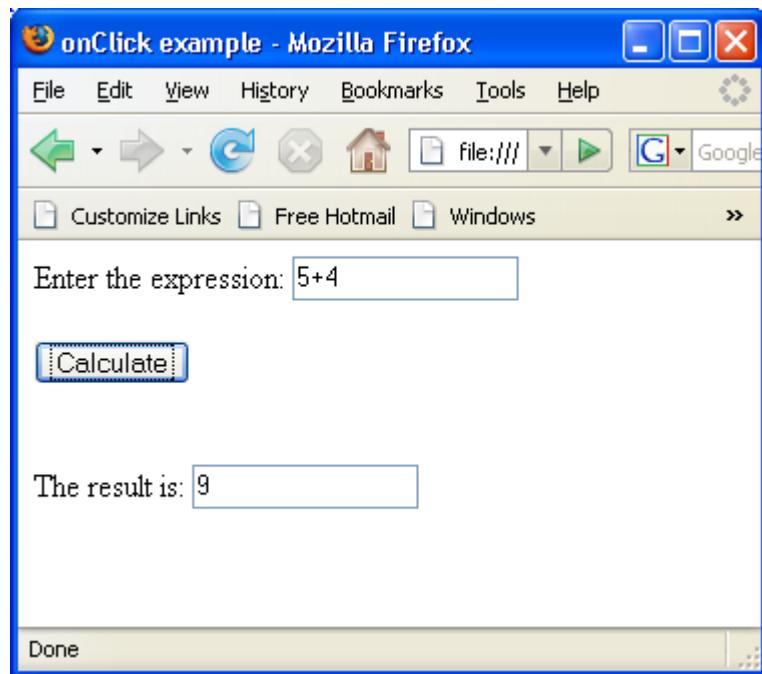
&lt;INPUT TYPE="text" NAME ="ketqua" SIZE =15&gt;

&lt;BR&gt;

&lt;/FORM&gt;

&lt;/BODY&gt;

&lt;/HTML&gt;

**Kết quả:****Hình 12.2: Minh họa sự kiện onClick**

- **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 lựa chọn trong danh sách chọn lựa thay đổi. Tuy nhiên, sự kiện onChange không được tạo ra khi một radio button hoặc một checkbox được nhập. 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 12.3:**

Ví dụ sau sẽ xử lý ký tự do người dùng nhập vào. Nếu ký tự nhập vào là một con số, thì trên màn hình sẽ hiển thị thông báo “Thank you！”, nếu không phải thì sẽ hiển thị “Please enter a numeric value”.

```
<HTML>
<HEAD>
    <TITLE> onChange example </TITLE>
    <SCRIPT LANGUAGE = "JavaScript">
        function checkNum(num)
        {
            if (num == "")
            {
                alert ("Please enter a number");
                return false;
            }
        }
    </SCRIPT>
</HEAD>
<BODY>
    <FORM>
        Enter a number: <INPUT type="text" name="num">
        <INPUT type="button" value="Check" onclick="checkNum(num.value)">
    </FORM>
</BODY>

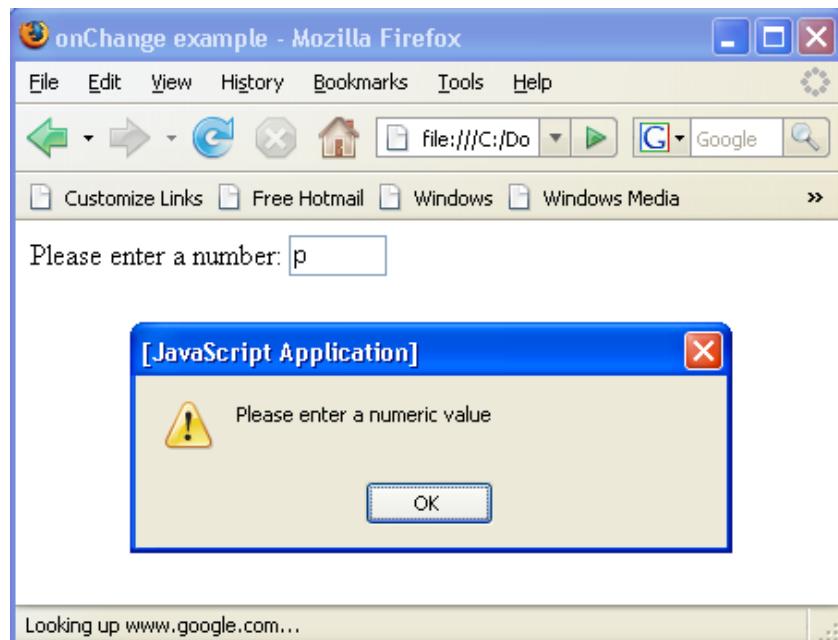
```

```

        }
        if (isNaN(num))
        {
            alert ("Please enter a numeric value");
            return false;
        }
        else
        {
            alert ("Thank you!");
        }
    }
</SCRIPT>
</HEAD>
<BODY bgColor = white>
<FORM>
    Please enter a number:
    <INPUT TYPE = text size = 5 onChange = "checkNum(this.value)">
</FORM>
</BODY>
</HTML>

```

**Kết quả:** Nếu giá trị nhập vào không phải là một số:



**Hình 12.3.1: Minh họa sự kiện onChange**

Nếu giá trị nhập vào là một số:



**Hình 12.3.2: Minh họa sự kiện onChange**

- **onFocus**

Sự kiện onFocus được gửi đi bất khi nào một phần tử form trở thành phần tử hiện thời. Chỉ khi một phần tử nhận được tiêu điểm nó mới nhận được dữ liệu nhập 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 đến các phần tử trên form).

- **onBlur**

onBlur ngược với onFocus. 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.

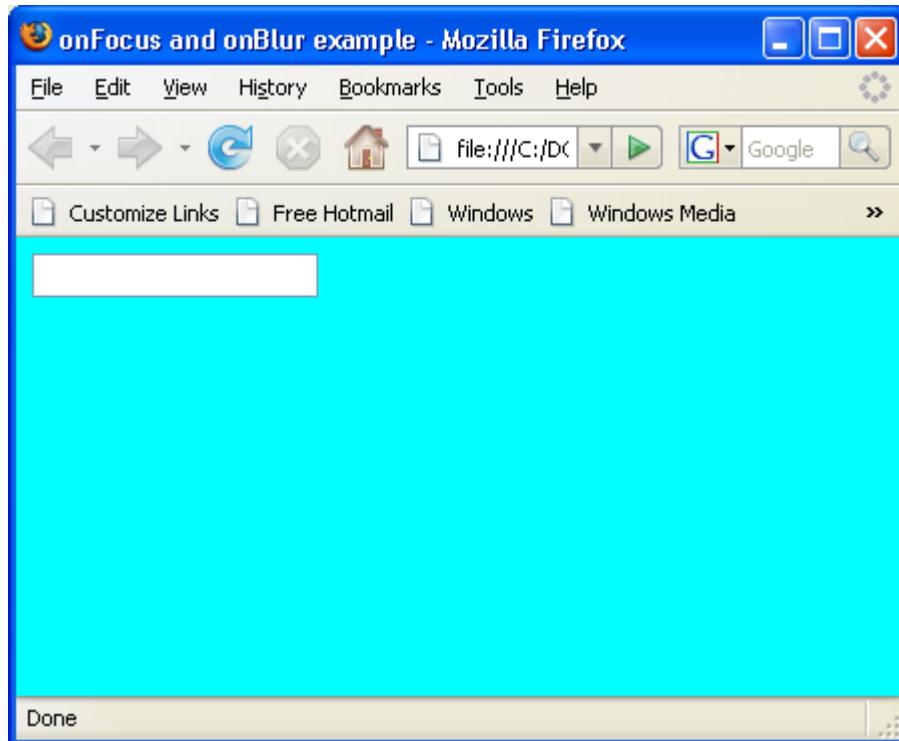
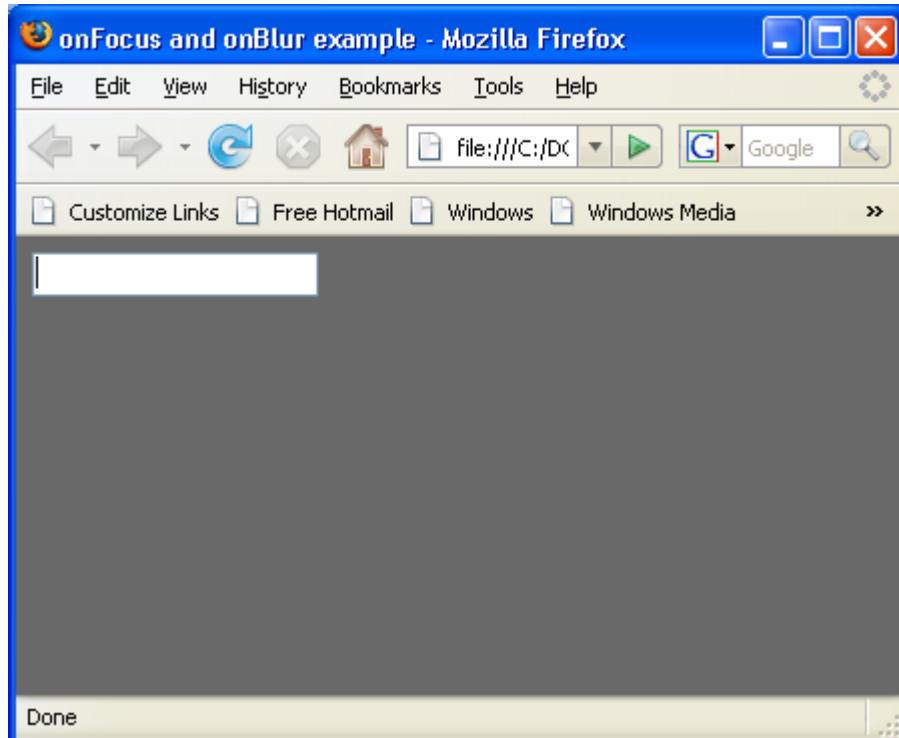
**Ví dụ 12.4:**

Ví dụ sau minh họa sự kiện onFocus và onBlur. Khi textbox nhận được focus, màu nền sẽ chuyển sang DIMGRAY, khi mất focus (blur), màu nền sẽ chuyển sang AQUA.

```
<HTML>
<HEAD>
    <TITLE> onFocus and onBlur example </TITLE>
</HEAD>
<BODY bgColor = "lavender">
    <FORM>
        <INPUT TYPE = text name= text1
            onBlur = "(document.bgColor='aqua')"
            onFocus= "(document.bgColor='dimgray')">
    </FORM>
```

&lt;/BODY&gt;

&lt;/HTML&gt;

**Kết quả:****Hình 12.4.1: Khi textbox mất focus (blur)****Hình 12.4.2: Khi textbox nhận focus**

- **onMouseOver**

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

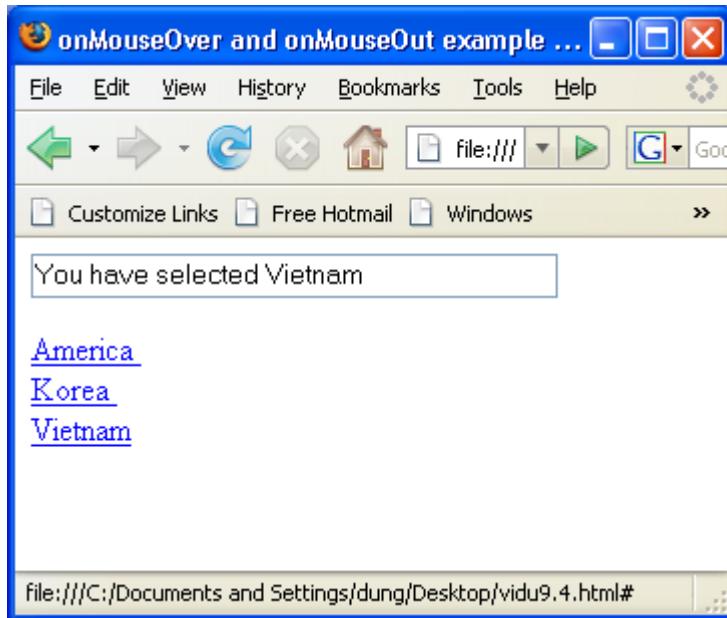
**Ví dụ 12.5:**

Ví dụ sau minh họa sự kiện onMouseOver. Trong ví dụ này, khi người dùng di chuyển con trỏ chuột lên một phần tử, thì sự kiện onMouseOver sẽ được tạo ra, lúc này trên ô textbox sẽ hiển thị một nội dung tương ứng với phần tử được lựa chọn. Có nghĩa là, khi con trỏ chuột được di chuyển đến phần tử Vietnam, thì trên ô textbox sẽ hiển thị câu “You have selected Vietnam”. Tương tự đối với các phần tử America và Korea.

```
<HTML>
<HEAD>
    <TITLE> onMouseOver and example </TITLE>
    <SCRIPT LANGUAGE = "JavaScript">
        var num=0
        function showLink(num)
        {
            if (num==1)
            {
                document.forms[0].elements[0].value=      "You      have
selected America";
            }
            if (num==2)
            {
                document.forms[0].elements[0].value=      "You      have
selected Korea";
            }
            if (num==3)
            {
                document.forms[0].elements[0].value=      "You      have
selected Vietnam";
            }
        }
    </SCRIPT>
</HEAD>
<BODY>
    <FORM>
        <INPUT TYPE="text" SIZE =40>
    </FORM>
    <a href = "#" onMouseOver = "showLink(1)"> America </a><br>
    <a href = "#" onMouseOver = "showLink(2)"> Korea </a><br>
    <a href = "#" onMouseOver = "showLink(3)"> Vietnam </a><br>
```

&lt;/BODY&gt;

&lt;/HTML&gt;

**Kết quả:****Hình 12.5: Minh họa sự kiện onMouseOver**

- **onMouseOut**

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

- **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 12.6:**

Đoạn mã sau minh họa sự kiện này, dùng trong thẻ BODY.

&lt;HTML&gt;

&lt;HEAD&gt;

&lt;TITLE&gt; Hello &lt;/TITLE&gt;

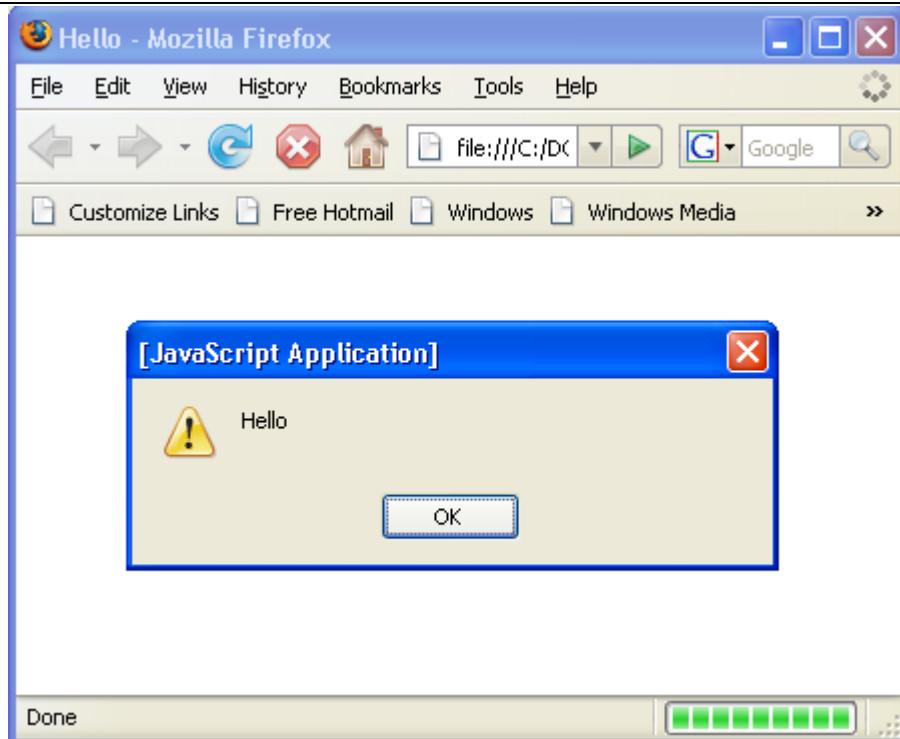
&lt;/HEAD&gt;

&lt;BODY onLoad="alert('Hello')"&gt;

&lt;/BODY&gt;

&lt;/HTML&gt;

**Kết quả:**



**Hình 12.6: Minh họa sự kiện onLoad**

- **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 các đối tượng button, document, và link.

**Ví dụ 12.7:**

Trong ví dụ sau, khi người dùng nhấp chuột vào nút *Change* thì màu nền sẽ chuyển sang màu *aqua*, và khi người dùng thả chuột thì màu nền sẽ là *limegreen*.

```
<HTML>
<HEAD>
    <TITLE> onMouseDown-onMouseUp example </TITLE>
</HEAD>
<BODY bgColor = "lavender">
    <FORM>
        <INPUT TYPE = button name= butt1 value="Change Color"
```

```
onMouseDown = "(document.bgColor='aqua')"
```

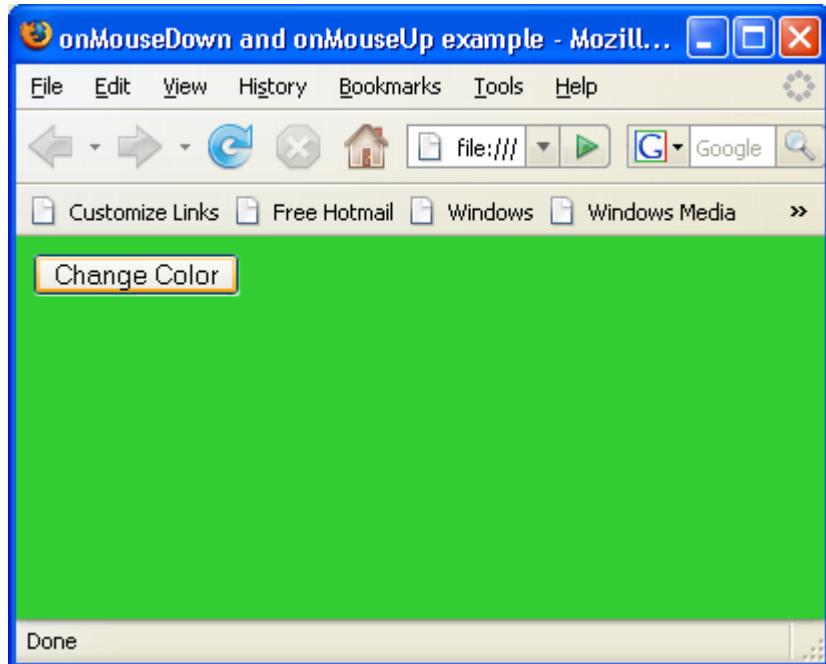
```
onMouseUp= "(document.bgColor='limegreen')">
```

```
</FORM>
```

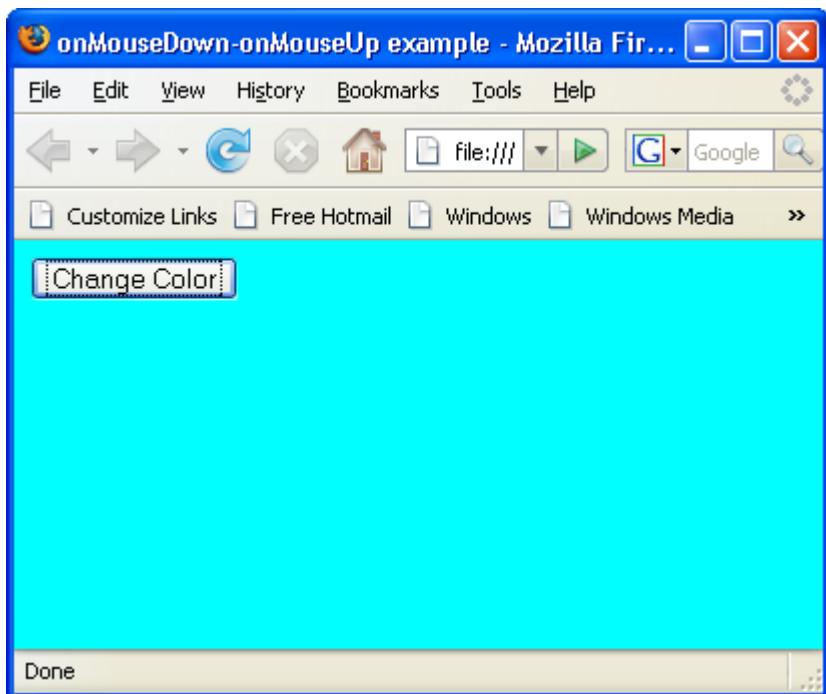
```
</BODY>
```

```
</HTML>
```

**Kết quả:**



**Hình 12.7.1: Minh họa sự kiện onMouseDown**



**Hình 12.7.2: Minh họa sự kiện onMouseUp**

- **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.

### 12.2.3 Làm việc với trình xử lý sự kiện

Khi một sự kiện được khởi tạo, chúng ta có thể tạo ra 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.

Ví dụ:

```
<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.

#### 12.2.3.1 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 See!"
      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.

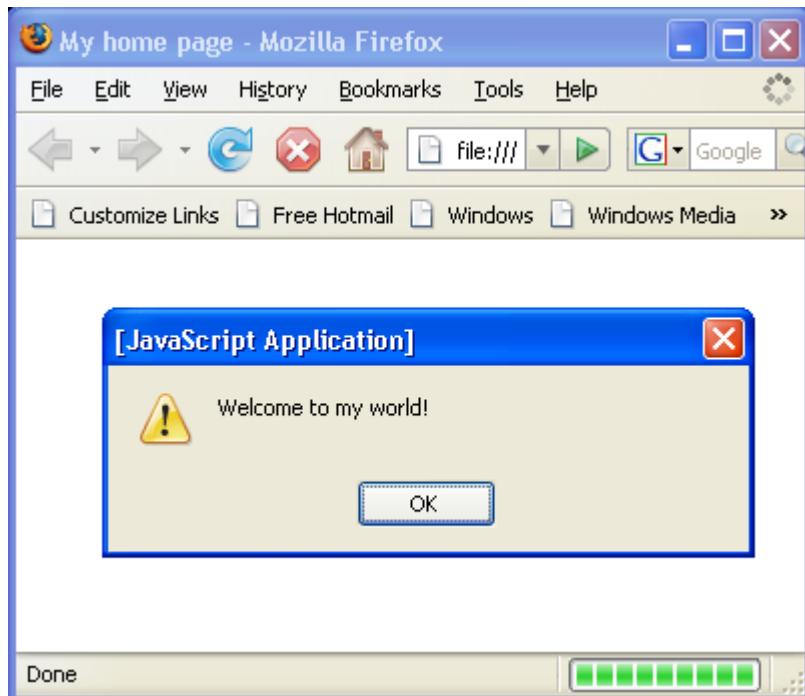
Câu lệnh này gán hàm greeting() cho trình xử lý sự kiện onLoad của window. Thuộc tính trình xử lý sự kiện được tham chiếu đến hàm greeting chứ không phải lời gọi đến hàm greeting().

**Ví dụ 12.8:**

```
<HTML>
  <HEAD>
    <TITLE> My home page </TITLE>
    <SCRIPT LANGUAGE = "JavaScript">
      function greeting()
      {
        alert ("Welcome to my world!");
      }
    </SCRIPT>
  </HEAD>
  <BODY onLoad="greeting()">
```

&lt;/BODY&gt;

&lt;/HTML&gt;

**Kết quả:****Hình 12.8: Kết quả ví dụ 9.8****12.2.3.2 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 12.9:**

&lt;HTML&gt;

&lt;HEAD&gt;

```
<TITLE> My home page </TITLE>
```

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

```
function greeting()
```

```
{
```

```
    alert ("Welcome to my world!");
```

```
}
```

```
    window.onLoad = greeting();
```

```
</SCRIPT>
```

&lt;/HEAD&gt;

&lt;/HTML&gt;

Kết quả sẽ tương tự như hình 9.8. Điểm mạnh của kỹ thuật này là tính linh hoạt. Có nghĩa là chúng ta có thể thay đổi nhanh chóng các trình xử lý sự kiện khi được yêu cầu.

### 12.3 Sử dụng sự kiện cho các thành phần trên form

Trong phần này chúng ta sẽ thảo luận về các phần tử trên form và các sự kiện trên các phần tử này.

#### 12.3.1 Đổi tượng Textfield (Trường văn bản)

Các Textfield nhận biết các sự kiện onBlur, onFocus, và onChange.

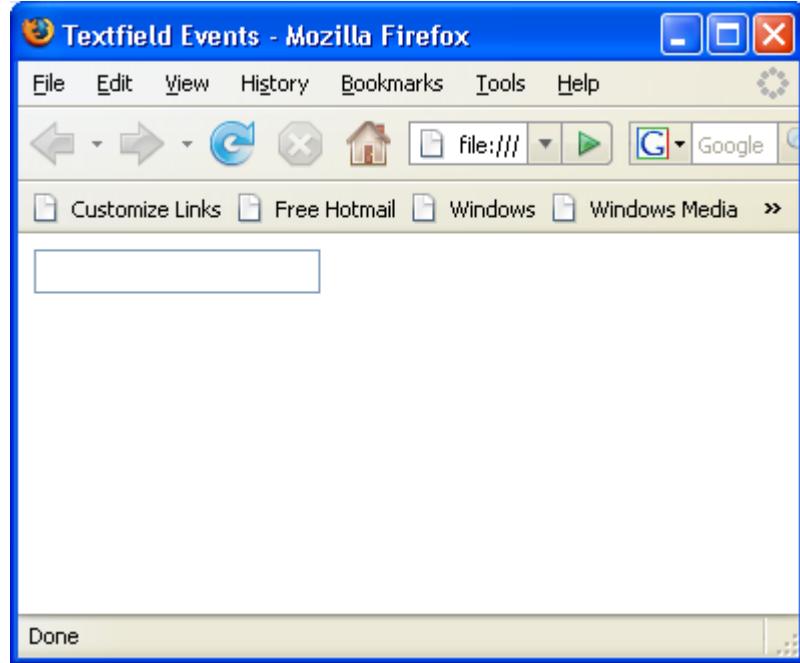
- Sự kiện onFocus xảy ra khi người dùng nhấp chuột vào trường text.
- Sự kiện onBlur xảy ra khi người dùng di chuyển ra khỏi trường text bằng cách nhấp chuột bên ngoài nó hoặc nhấn phím “tab”.
- Sự kiện onChange xảy ra khi người dùng có sự thay đổi bên trong trường text và sau đó di chuyển ra khỏi trường văn bản.

#### Ví dụ 12.10:

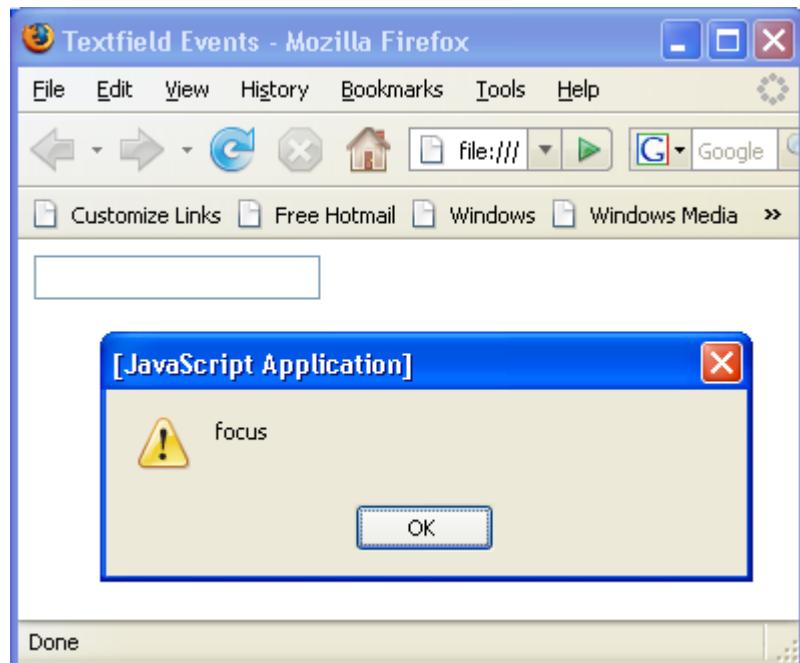
Ví dụ dưới đây minh họa các sự kiện nói trên. Trong ví dụ này, khi người dùng nhấp chuột bên trong trường text, sự kiện onFocus sẽ xảy ra. Khi người dùng có sự thay đổi ở văn bản sau đó di chuyển ra khỏi vùng văn bản hiện thời, sự kiện onChange sẽ xảy ra.

```
<HTML>
<HEAD>
    <TITLE> Textfield Events </TITLE>
    <SCRIPT LANGUAGE = "JavaScript">
        function writeIt(value)
        {
            alert (value);
        }
    </SCRIPT>
</HEAD>
<BODY BGCOLOR = "#FFFFFF">
    <FORM NAME = "myfm">
        <INPUT TYPE="text" NAME="first_text"
        onFocus="writeIt('focus');"
        onChange="writeIt('change');">
    </FORM>
</BODY>
</HTML>
```

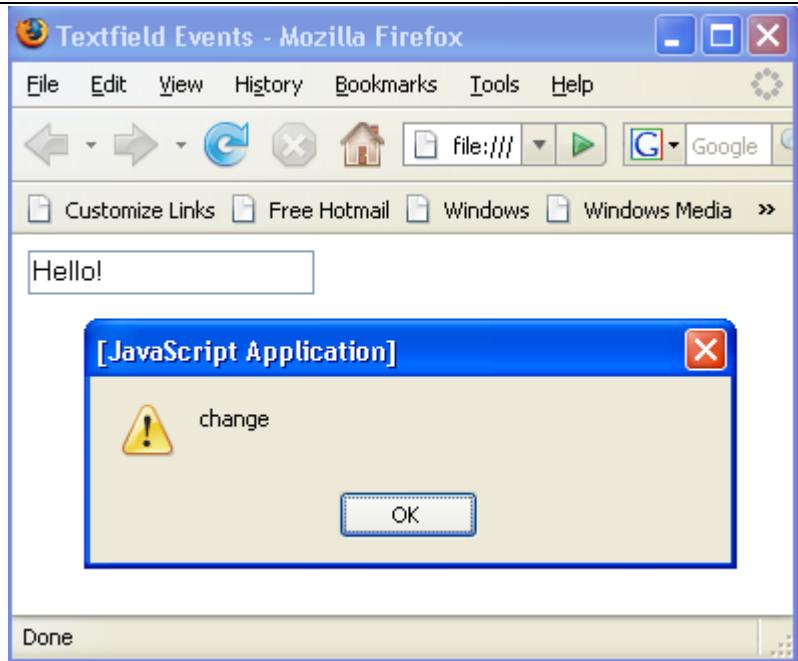
#### Kết quả:



Hình 12.9.1: Kết quả ví dụ 9.10



Hình 12.9.2: Khi nhấp chuột vào textbox



**Hình 12.9.3: Khi nhấp chuột ra khỏi textbox**

(Nếu nội dung trong textbox thay đổi )

### 12.3.2 Đổi tương Command Button

Sự kiện onClick của một command button xảy ra khi người dùng nhấp chuột vào command button đó.

Ví dụ dưới đây sẽ minh họa cách sử dụng sự kiện onClick trên đối tượng command button. Trong ví dụ này, khi người dùng nhấp chuột vào nút *Copy*, sự kiện onClick xảy ra và phần văn bản ở trường text thứ nhất sẽ được sao chép sang trường text thứ hai.

#### Ví dụ 12.11:

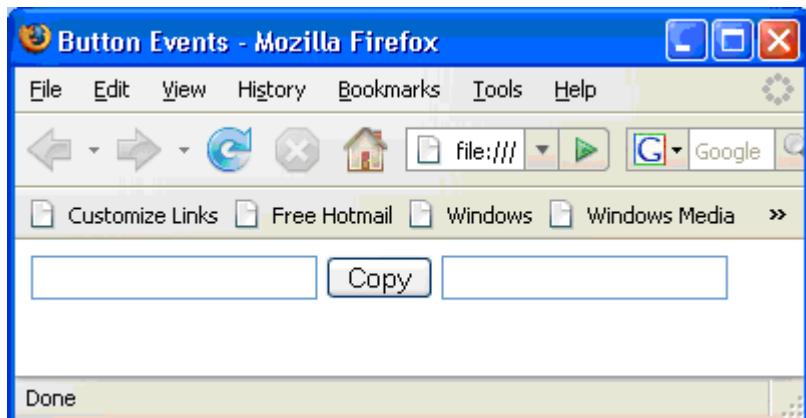
```
<HTML>
<HEAD>
    <TITLE> Button Events </TITLE>
    <SCRIPT LANGUAGE = "JavaScript">
        function writeIt(value)
        {
            myfm.second_text.value = value;;
        }
    </SCRIPT>
</HEAD>
<BODY BGCOLOR = "#FFFFFF">
<FORM NAME = "myfm">
    <INPUT TYPE="text" NAME="first_text">
    <INPUT TYPE="button" VALUE="Copy">
```

```

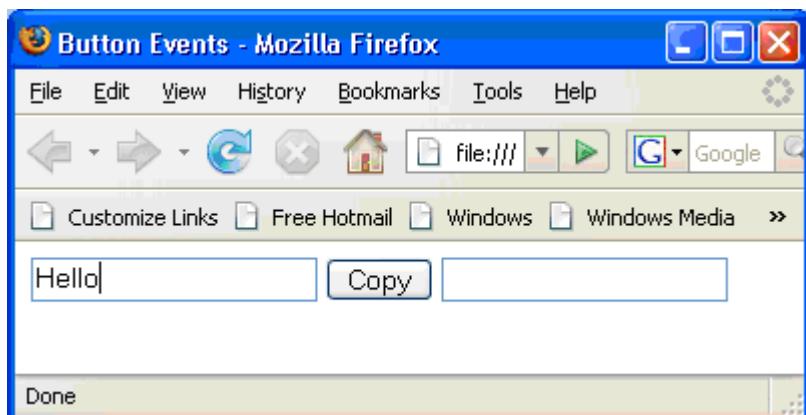
        onClick="writeIt(myfm.first_text.value);">
<INPUT TYPE="text" NAME="second_text">
</FORM>
</BODY>
</HTML>

```

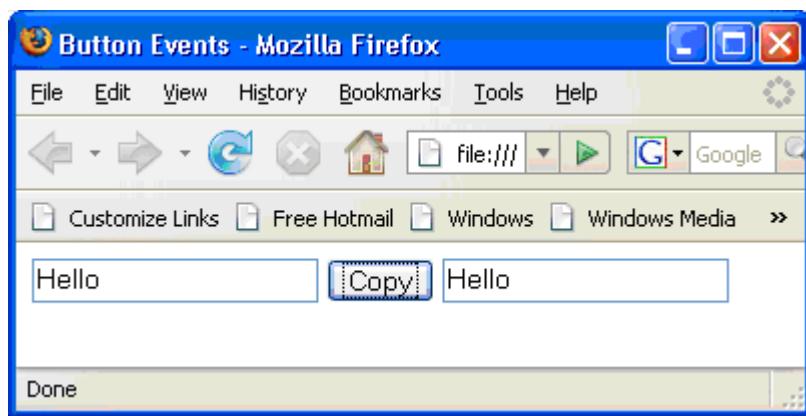
**Kết quả:**



Hình 12.10.1: Kết quả ví dụ 9.10



Hình 12.10.2: Sau khi nhập văn bản vào trường text đầu tiên



Hình 12.10.3: Sau khi nhấp vào nút Copy

### 12.3.3 Đối tượng Checkbox

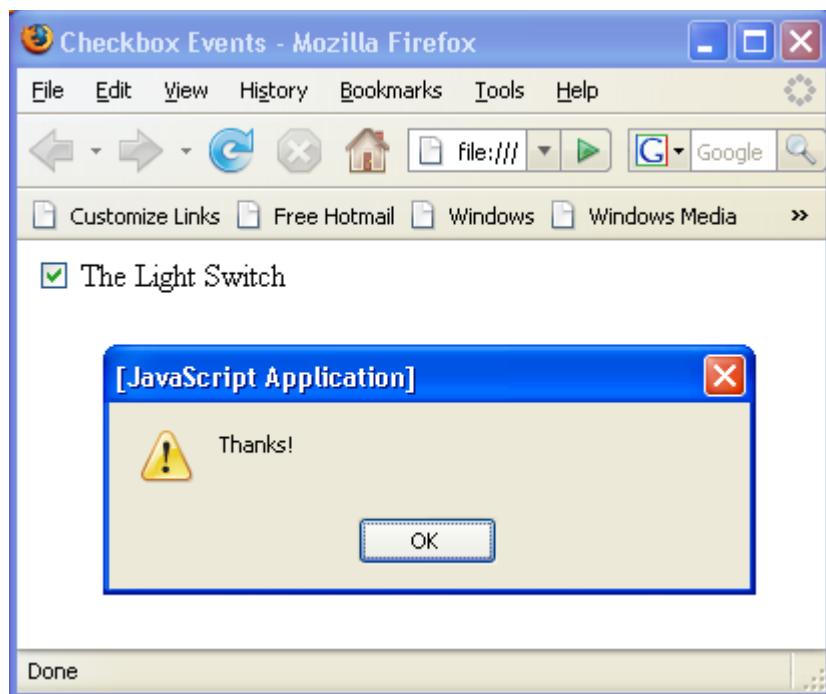
Checkbox là một đối tượng của form hoạt động theo cơ chế bật-tắt. Điều này có nghĩa là Checkbox có thể được check hoặc không. Cũng như button, checkbox cũng

Trong ví dụ sau, trên form sẽ có một ô chọn checkbox, khi người dùng kích chọn vào checkbox này (tạm hiểu là khi “bật đèn”) thì màu nền sẽ chuyển sang màu trắng, đồng thời kèm theo thông báo “Thanks!”, ngược lại, khi người dùng bỏ chọn (tạm hiểu là “tắt đèn”) thì màu nền sẽ là màu đen và hiển thị thông báo “Hey! Turn that back on!”

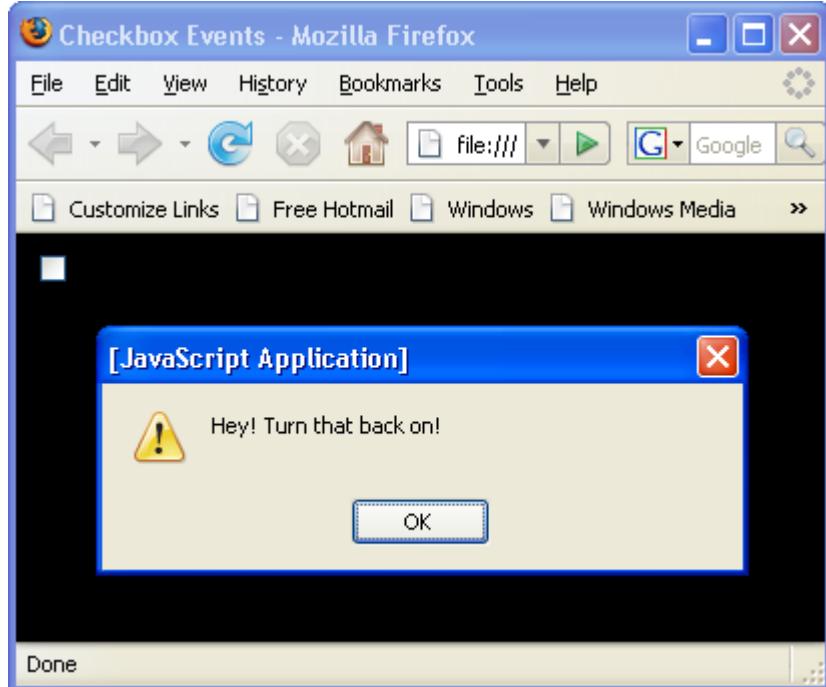
### Ví dụ 12.12:

```
<HTML>
<HEAD>
    <TITLE> Checkbox Events </TITLE>
    <SCRIPT LANGUAGE = "JavaScript">
        function switchLight()
        {
            var the_box=window.document.form_2.check_1;
            var the_switch = "";
            if (the_box.checked == false)
            {
                document.bgColor='black';
                alert ("Hey! Turn that back on!");
            }
            else
            {
                document.bgColor='white';
                alert ("Thanks!");
            }
        }
    </SCRIPT>
</HEAD>
<BODY BGCOLOR = "#FFFFFF">
    <FORM NAME = "form_2">
        <INPUT TYPE="checkbox" NAME="check_1"
        onClick="switchLight();">
        The Light Switch
    </FORM>
</BODY>
</HTML>
```

**Kết quả:**



**Hình 12.11.1: Khi nhấp chọn vào ô checkbox**



**Hình 12.11.2: Khi bỏ chọn checkbox**

#### 12.3.4 Đối tượng radio

Mã JavaScript của sự kiện onClick trên nút radio tương tự như đối với checkbox, chúng chỉ khác nhau trong cách dùng trên form. Khi chúng ta để một checkbox ở chế độ tắt (bật) ta có thể bật lại (tắt đi). Tuy nhiên đối với các nút radio thì khác, một khi đã được bật, thì tắt cả các radio khác đều ở chế độ tắt, ta không thể thay đổi trạng thái của radio này bằng cách nhấp vào nó như đối với checkbox. Trạng thái này của các nút giữ nguyên cho đến khi một radio khác được bật. Lúc này, chỉ có radio mới được bật là ở trạng thái bật còn các radio khác đều ở chế độ tắt.

Ví dụ sau minh họa đối tượng radio. Cũng tương tự như ví dụ 9.11, nhưng trong ví dụ này, checkbox được thay bằng hai nút radio, một nút là “bật đèn” (Light on) và một nút là “tắt đèn” (Light off). Với chức năng tương tự, khi chọn “Light on” thì màu nền sẽ là màu trắng, và thông báo kèm theo sẽ là “Thanks！”, ngược lại, nếu chọn “Light off” thì màu nền sẽ chuyển thành màu đen, và sẽ hiển thị thông báo “Hey! Turn that back on！”.

### Ví dụ 12.13:

```

<HTML>
<HEAD>
    <TITLE> Option Button Events </TITLE>
    <SCRIPT LANGUAGE = "JavaScript">
        function offButton()
        {
            var the_box=window.document.form_1.radio_1;
            if (the_box.checked == true)
            {
                window.document.form_1.radio_2.checked
                = false;
                document.bgColor='black';
                alert ("Hey! Turn that back on!");
            }
        }
        function onButton()
        {
            var the_box=window.document.form_1.radio_2;
            if (the_box.checked == true)
            {
                window.document.form_1.radio_1.checked
                = false;
                document.bgColor='white';
                alert ("Thanks!");
            }
        }
    </SCRIPT>
</HEAD>
<BODY BGCOLOR = "#FFFFFF" TEXT ="red">
    <FORM NAME = "form_1">
        <INPUT TYPE="radio" NAME="radio_1"

```

```
Light off
```

```
<INPUT TYPE="radio" NAME="radio_2"
onClick="onButton();" checked>
```

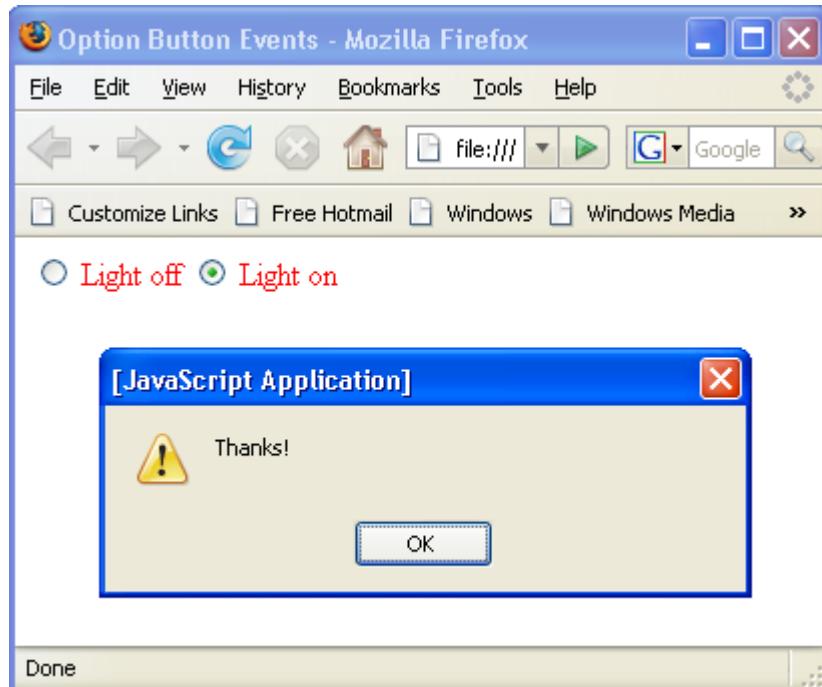
```
Light on
```

```
</FORM>
```

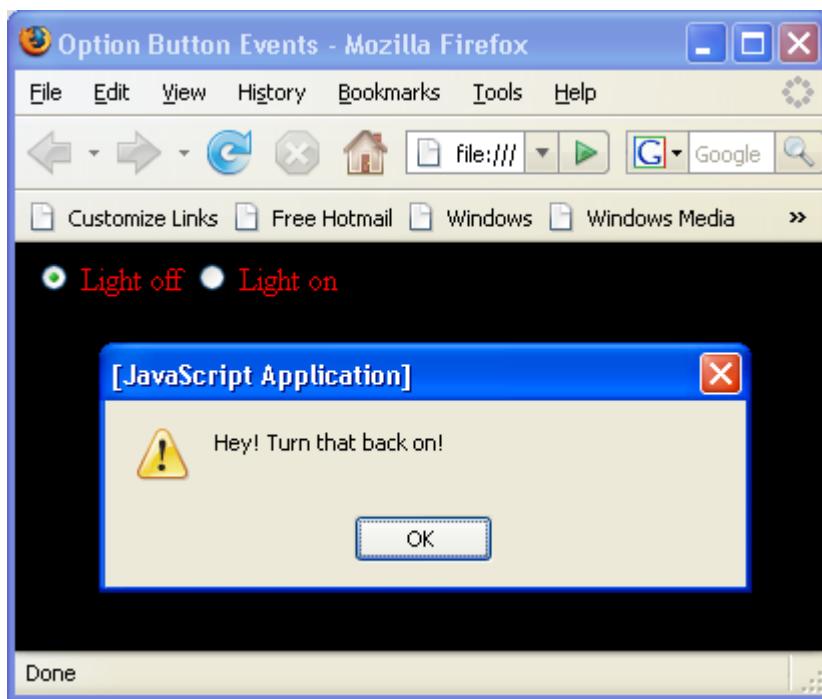
```
</BODY>
```

```
</HTML>
```

**Kết quả:**



Hình 12.12.1: Khi chọn Light on



**Hình 12.12.2: Khi chọn Light off****12.3.5 Đổi tượng ComboBox (lựa chọn)**

Đổi tượng ComboBox xuất hiện trong form HTML giống như một danh sách đồ xuống hoặc danh sách cuộn của các tùy chọn.

Danh sách tùy chọn được mô tả giữ hai thẻ <SELECT> và </SELECT> bằng cách sử dụng thẻ <OPTION>

ComboBox hỗ trợ các sự kiện onBlur, onFocus, và onChange.

**12.3.6 Kiểm tra tính hợp lệ của nội dung các trường trên form**

Chương trình dưới đây là một ví dụ về việc kiểm tra tính hợp lệ của nội dung các trường trên form trước khi chuyển nó cho server để tiếp tục xử lý. Trong quá trình kiểm tra tính hợp lệ của nội dung các trường trên form, người lập trình phải kiểm tra từng trường để bảo đảm rằng không có trường nào bị bỏ trống hoặc chứa các thông tin không hợp lệ.

**Ví dụ 12.14:**

```
<HTML>
<HEAD>
    <TITLE> Form events </TITLE>
    <SCRIPT LANGUAGE = "JavaScript">
        function validateFirstName()
        {
            var str=form1.fname.value;
            if (str.length == 0)
            {
                alert ("The first name cannot be
empty");
                return false;
            }
            return true;
        }
        function validateLastName()
        {
            var str=form1.lname.value;
            if (str.length == 0)
            {
                alert ("The last name cannot be
empty");
                return false;
            }
        }
    </SCRIPT>
</HEAD>
<BODY>
    <FORM NAME="form1">
        First Name: <INPUT TYPE="text" NAME="fname">
        Last Name: <INPUT TYPE="text" NAME="lname">
        <INPUT TYPE="button" VALUE="Submit" onClick="validateFirstName(); validateLastName();">
    </FORM>
</BODY>

```

```
return true;
```

```
}
```

```
function validateEmail()
```

```
{
```

```
    var str=form1.email.value;
```

```
    if (str.length == 0)
```

```
{
```

```
        alert ("The Email field cannot be  
empty");
```

```
        return false;
```

```
}
```

```
}
```

```
function proccessForm()
```

```
{
```

```
    disp = open("", "result")
```

```
    disp.document.write("<TITLE> Result Page  
</TITLE>"+"<PRE>")
```

```
    disp.document.write("<H2 ALIGN='CENTER'>" +  
"Thanks for signing in " +  
</H2>"+"<HR>"+"<BR><BR>")
```

```
    disp.document.write("First name \t\t:" +  
form1.fname.value+"<BR>")
```

```
    disp.document.write("Last name \t\t:" +  
form1.lname.value+"<BR>")
```

```
    disp.document.write("Email \t\t\t:" +  
form1.email.value+"<BR>")
```

```
    disp.document.write("Your comments \t\t:" +  
form1.comment.value+"<BR>")
```

```
    disp.document.write("<PRE>")
```

```
    if (disp.confirm("Is this information  
correct"))
```

```
        disp.close()
```

```
}
```

```
</SCRIPT>
```

```
</HEAD>
```

```
<BODY BGCOLOR = "#FFFFFF">
```

```
    <H2 ALIGN = "CENTER"> Handling Form  
Events</H2><HR>
```

```
    <FORM NAME = "form1">
```

```

<P> First name: <INPUT TYPE="text" NAME="fname"
size=10 onBlur = "validateFirstName()">

    Last name: <INPUT TYPE="text" NAME="lname"
size=15    onBlur = "validateLastName()"></P>

    <P> Email: <INPUT TYPE="text" NAME="email"
size=10    onBlur = "validateEmail()">

    Comments: <TEXTAREA NAME="comment" rows=4
cols=30>Enter your comments </TEXTAREA></P>

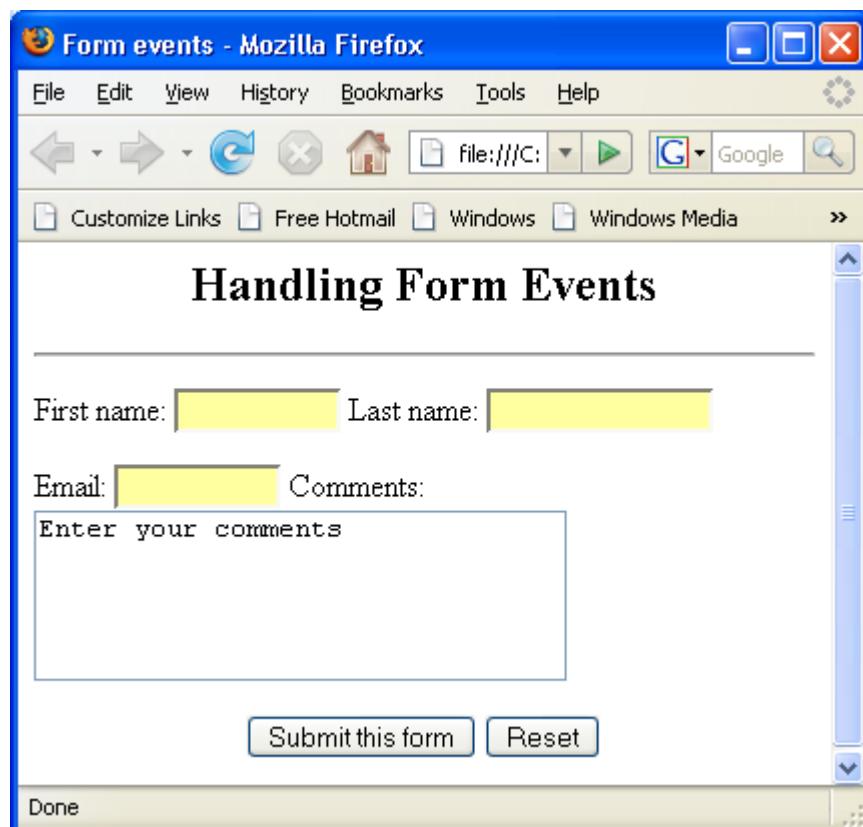
    <P ALIGN="CENTER"><INPUT TYPE = "button"
VALUE="Submit this form" onClick =
"processForm()">

        <INPUT TYPE= "reset"></P>
    </FORM>

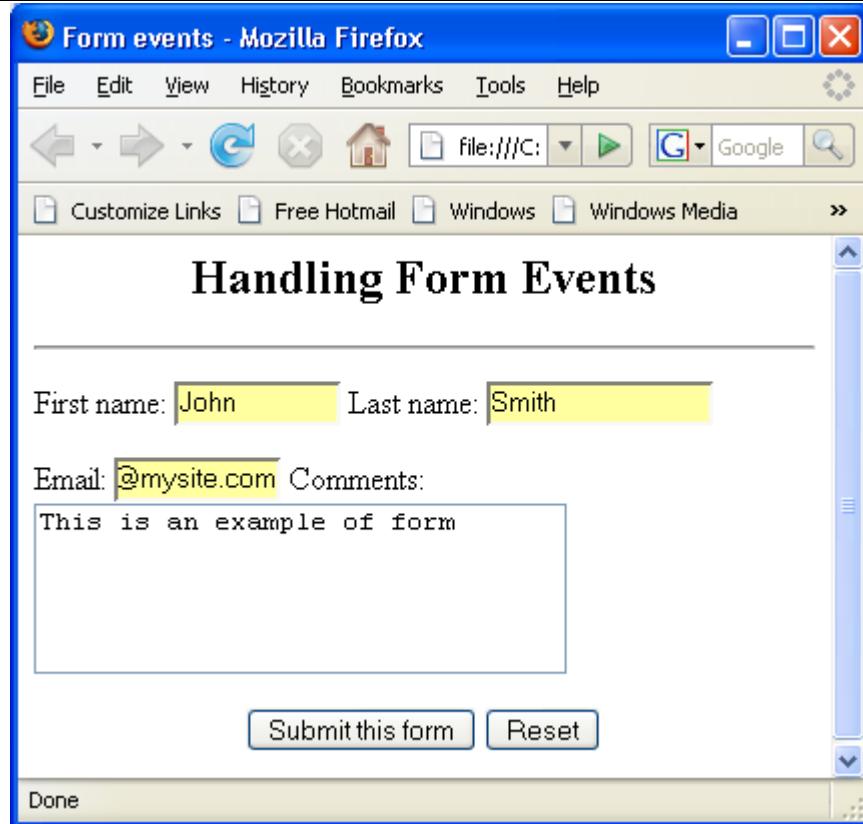
</BODY>
</HTML>

```

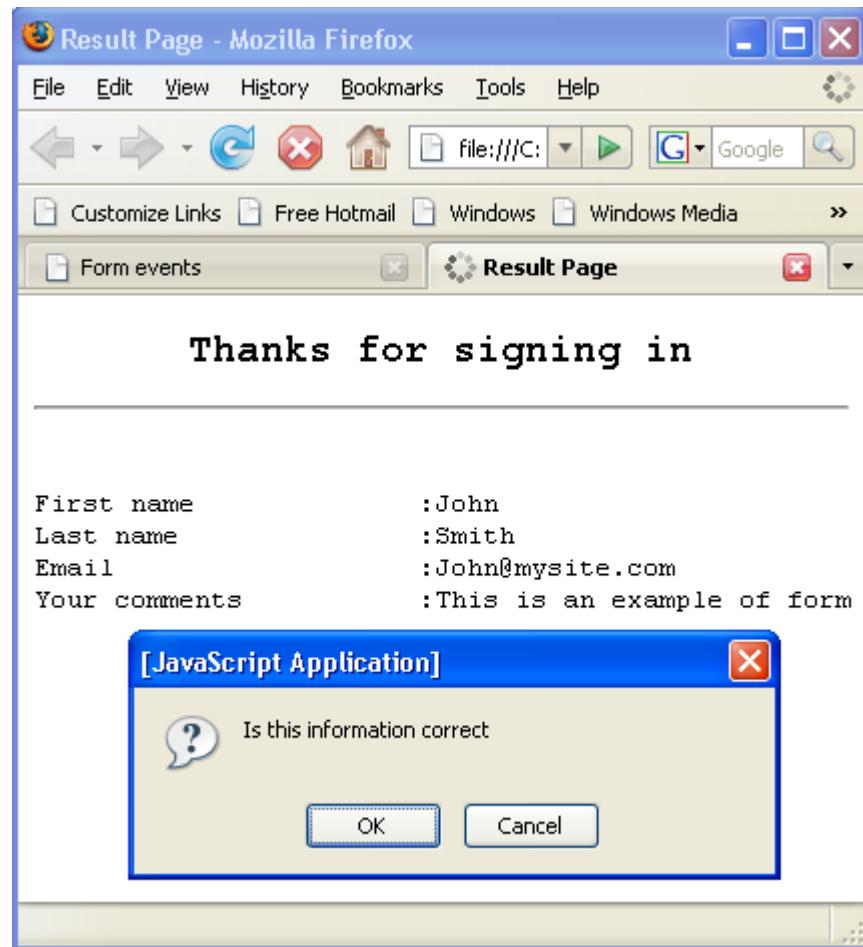
**Kết quả:**



**Hình 12.10.1: Kết quả ví dụ 9.11**



Hình 12.10.2: Sau khi nhập thông tin vào form



Hình 12.10.3: Sau khi bấm nút “Submit this form”

Chúng ta xét tiếp ví dụ sau, ví dụ này là đoạn chương trình kiểm tra tính đúng đắn của một địa chỉ email được nhập vào qua Textbox.

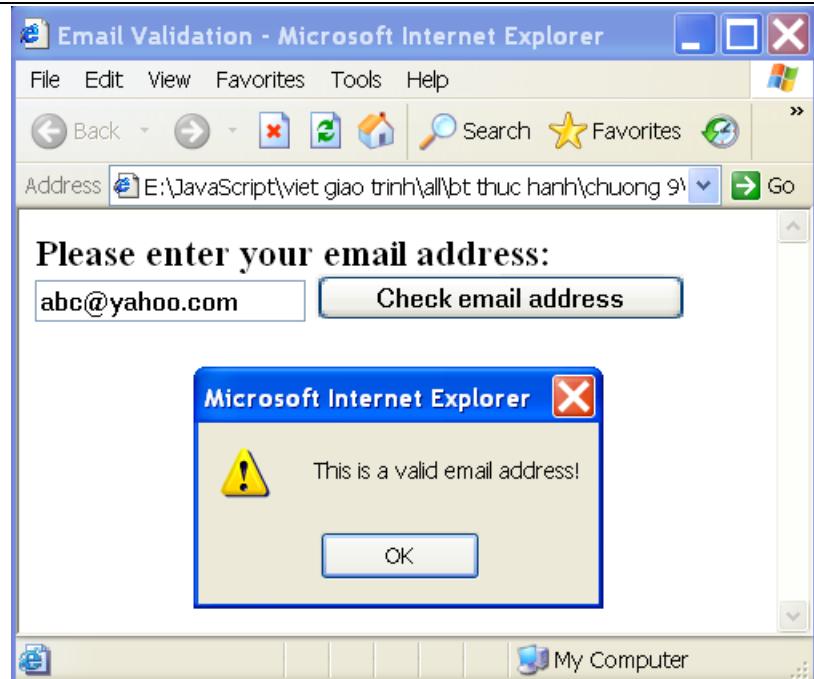
### Ví dụ 12.15:

```

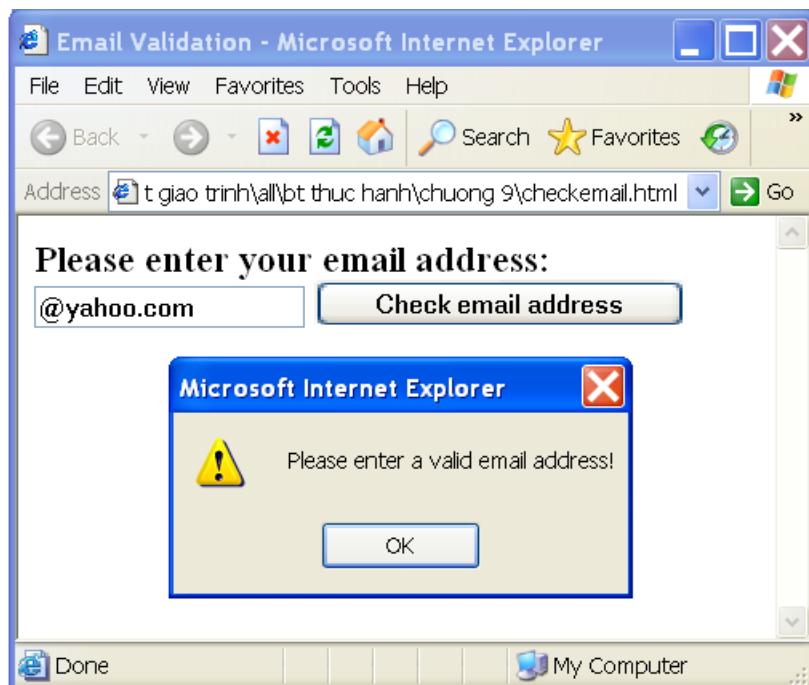
<HTML>
<HEAD>
    <TITLE> Email Validation </TITLE>
    <script language=Javascript>
        function IsEmailValid (Formname,ElemName)
        {
            var EmailOk = true;
            var Temp = ElemName;
            var AtSym = Temp.value.indexOf ('@');
            var Period =Temp.value.lastIndexOf('.');
            var Space = Temp.value.indexOf(' ');
            var Length = Temp.length -1;
            if ((AtSym <1)||(Period<=AtSym+1)||(Period ==Length)|| (Space
                >=0))
            {
                EmailOk=false
                alert('Please enter a valid email address!')
                Temp.focus()
            }
            else
                alert('This is a valid email address!')
            return EmailOk
        }
    </script>
</HEAD>
<BODY>
    <form name="frm">
        <b> Please enter your email address: <input type = "text" name =
        "text1">
        <input type= "button" value = "Check email address" onClick =
        "IsEmailValid('frm', frm.text1);">
    </form>
</BODY>
</HTML>

```

### Kết quả:



Hình 12.11.1: Một địa chỉ email hợp lệ



Hình 12.11.1: Kiểm tra địa chỉ email không hợp lệ, yêu cầu người dùng nhập lại

#### 12.4 Câu hỏi và bài tập

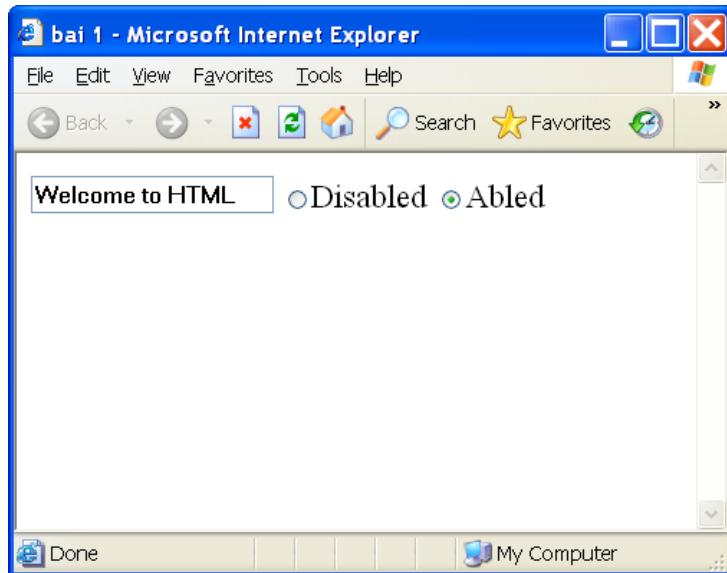
1. Sự kiện onChange có được tạo ra khi một radio button hoặc một checkbox được nhấp hay không ? \_\_\_\_\_ (Có/Không )
2. Sự kiện \_\_\_\_\_ được tạo ra bắt cứ khi nào con trỏ chuột di chuyển lên trên một phần tử.
3. Sự kiện \_\_\_\_\_ được tạo ra bắt cứ khi nào con trỏ chuột di chuyển ra khỏi phần tử đó.
4. Sự kiện \_\_\_\_\_ được kích hoạt khi hành động nhấp chuột xảy ra.
5. Sự kiện \_\_\_\_\_ được kích hoạt khi hành động nhả chuột xảy ra.

6. Khi một sự kiện được khởi tạo, chúng ta có thể tạo ra một đoạn mã JavaScript để đáp ứng lại sự kiện. Đoạn mã này được gọi là \_\_\_\_\_.
7. Các Textfield nhận biết các sự kiện onBlur, onFocus, và onChange. Trong đó:
  - Sự kiện \_\_\_\_\_ xảy ra khi người dùng có sự thay đổi bên trong trường text và sau đó di chuyển ra khỏi trường văn bản.
  - Sự kiện \_\_\_\_\_ xảy ra khi người dùng di chuyển ra khỏi trường text bằng cách nhấp chuột bên ngoài nó hoặc nhấn phím “tab”.
  - Sự kiện \_\_\_\_\_ xảy ra khi người dùng nhấp chuột vào trường text.
8. Trong một form, ta có thể chọn nhiều nút radio trong cùng một nhóm.  
\_\_\_\_\_ (Đúng/Sai)
9. Đối tượng \_\_\_\_\_ xuất hiện trong form HTML giống như một danh sách sổ xuống hoặc danh sách cuộn của các tùy chọn.

**Bài tập thực hành chương 12:**

1. Hãy viết một trang web, gồm có một textbox và hai nút radio như hình dưới

Yêu cầu: Viết một hàm JavaScript, khi kích vào nút radio “Disabled” thì giá trị trong textbox ẩn, và khi chọn nút radio “abled” thì cho textbox soạn thảo được.



2. Hãy viết một trang web, tương tự bài 1 gồm có một button và một nút CheckBox như hình dưới. Khi kích vào CheckBox thì cho button này ẩn đi.
3. Hãy viết một trang web, chèn một textbox, gõ nội dung vào textbox. Khi textBox mất focus (tiêu điểm) thì sẽ có một hộp thông báo hiển thị nội dung của textbox.
4. Hãy viết một trang web, chèn một textbox và một button. Nếu không nhập nội dung vào textbox thì button này sẽ bị ẩn (thuộc tính disabled của button bằng true), khi nội dung được nhập vào textbox thì button này sẽ xuất hiện. Bấm vào button thì sẽ xuất hiện một thông báo nội dung vừa nhập trong textbox.
5. Thiết kế một form gồm có hai textbox và một nút lệnh button, textbox1 để người dùng nhập username, textbox2 để người dùng nhập password (khi người dùng gõ vào textbox này thì trên textbox chỉ hiển thị các dấu \* thay vì nội dung đang nhập). Sau khi nhập nội dung vào hai textbox này, nhấn vào nút lệnh thì username và password của người dùng sẽ được hiển thị trong một thông báo.
6. Làm lại ví dụ 9.14 trong chương này, yêu cầu các trường không được để trống, và email nhập vào phải là một địa chỉ email hợp lệ. Nếu email không hợp lệ thì chỉ buộc người dùng nhập lại địa chỉ email (Các trường khác không bắt buộc nhưng vẫn có thể thay đổi nội dung được).

## TÀI LIỆU THAM KHẢO

- [1] *Căn bản thiết kế Web*, Nhà xuất bản thông kê.
- [2] **Nguyễn Viết Linh** (2002), *Hướng dẫn lập trình và tham khảo toàn diện JavaScript*, NXB Thanh Niên, Bến Tre.
- [3] **Nguyễn Trường Sinh** (2006), *Học nhanh JavaScript bằng hình ảnh*, NXB Lao động xã hội.
- [4] **Nguyễn Trường Sinh** (2005), *Thiết kế Web động với JavaScript*, NXB Lao động xã hội
- [5] **Lê Minh Trí** (2000), *JavaScript*, NXB Trẻ và Công ty văn hóa Phương Nam, TP Hồ Chí Minh.
- [6] *Thiết kế trang Web cá nhân bằng ngôn ngữ HTML*, Nhà Xuất bản thông kê
- [7] **Thu Nhi**, *Thiết kế trang Web với HTML*, NXB Trẻ
- [8] **Trung tâm đào tạo lập trình viên quốc tế Softech – Aptech**, *Giáo trình HTML và JavaScript*.
  
- [9] **JavaScript 2.0: The Complete Reference, Second Edition**  
by Thomas Powell and Fritz Schneider.  
McGraw-Hill/Osborne © 2004
- [10] **JavaScript & DHTML Cookbook**  
By [Danny Goodman](#)  
Publisher: O'Reilly
- [11] **JavaScript: The Definitive Guide, 4th Edition**  
By David Flanagan  
Publisher: O'Reilly
- [12] **Learning JavaScript**  
By Shelley Powers  
Publisher: O'Reilly