

## MỤC LỤC

DANH MỤC HÌNH VẼ.....	5
THUẬT NGỮ VIẾT TẮT .....	8
LỜI NÓI ĐẦU .....	1
CHƯƠNG 1: TỔNG QUAN VỀ MẠNG INTERNET .....	2
1.1. Lịch sử Web (World Wide Web - WWW) .....	2
1.2. Cơ bản về trình duyệt Web.....	2
1.3. Web server (máy chủ Web).....	3
1.4. File Transfer Protocol – FTP.....	4
1.5. Giao thức TCP/IP .....	4
1.6. Trang Web (Webpage).....	5
1.7. Hypertext Transfer Protocol - HTTP .....	5
1.8. Hyperlink.....	7
CHƯƠNG 2: NGÔN NGỮ ĐÁNH DẤU SIÊU VĂN BẢN HTML .....	8
2.1. Tổng quan về HTML.....	8
2.2. Cấu tạo thẻ của html và cấu trúc trang HTML.....	8
2.2.1. Cấu trúc thẻ HTML.....	8
2.2.2. Cấu trúc tài liệu HTML.....	9
2.2.3. Ý nghĩa của các thẻ trong cấu trúc trang Web .....	10
2.3. Các thẻ trình bày trang web.....	11
2.3.1. Thẻ ngắt dòng và khai báo đoạn .....	11
2.3.2. Các thẻ đề mục.....	11
2.4. Các thẻ định dạng kí tự .....	12
2.5. Thẻ Body và các thuộc tính.....	13
2.6. Các thẻ tạo danh sách .....	13
2.7. Chèn ảnh, âm thanh và video .....	15
2.7.1. <i>Cú pháp thẻ IMG chèn hình ảnh vào trang web</i> .....	15
2.7.2. <i>Cú pháp thẻ EMBED chèn hình âm thanh và video vào trang web</i> .....	16
2.7.3. <i>Span &amp; div</i> .....	17
2.8. Siêu liên kết.....	17
2.9. Bảng .....	18
2.9.1. Các thẻ dùng tạo bảng.....	18
2.9.2. Thẻ Table và thuộc tính .....	19
2.9.3. Thẻ TR và thuộc tính .....	19
2.9.4. Thẻ TD và thuộc tính .....	19
2.9.5. Thẻ TH và thuộc tính .....	20
2.10. Phân khung cho trang web.....	21

2.11.	Mẫu biểu (FORM).....	22
2.11.1.	Giới thiệu mẫu biểu.....	22
2.11.2.	Các phần tử nhập HTML .....	22
2.11.3.	Ví dụ về biểu mẫu: .....	26
CHƯƠNG 3: NGÔN NGỮ CSS.....		27
3.1.	Giới thiệu.....	27
3.2.	Một số quy ước về cách viết CSS .....	27
3.2.1.	<i>Cú pháp CSS</i> .....	27
3.2.2.	<i>Đơn vị CSS</i> .....	30
3.2.3.	<i>Vị trí đặt CSS</i> .....	31
3.2.4.	<i>Sự ưu tiên</i> .....	32
3.3.	Màu chữ và màu nền .....	34
3.4.	Font chữ.....	36
3.5.	Text.....	38
3.6.	Pseudo-classes for Links .....	39
3.7.	Span & div.....	40
3.8.	Box Model.....	41
3.9.	Margin & padding .....	42
3.9.1.	Thuộc tính margin.....	42
3.9.2.	Thuộc tính padding .....	43
3.10.	Border .....	44
3.11.	Height & width .....	44
3.11.1.	Thuộc tính width .....	44
3.11.2.	<i>Thuộc tính height</i> .....	46
3.12.	Float & clear .....	48
3.12.1.	Thuộc tính float .....	48
3.12.2.	Thuộc tính clear.....	49
3.13.	Position .....	50
3.14.1.	<i>Absolute position</i> .....	51
3.14.2.	<i>Relative position</i> .....	52
CHƯƠNG 4: NGÔN NGỮ KỊCH BẢN JAVASCRIPT.....		53
4.1.	Nhập môn JavaScript .....	53
4.1.1.	<i>Sử dụng một file nguồn JavaScript</i> .....	53
4.1.2.	<i>Hiển thị một dòng text</i> .....	54
4.1.3.	<i>Giao tiếp với người sử dụng</i> .....	55
4.2.	Biến trong Javascript.....	57
4.2.1.	<i>Biến và phân loại biến</i> .....	57

4.2.2.	<i>Kiểu dữ liệu</i> .....	57
4.3.	Xây dựng các biểu thức trong Javascript .....	58
4.3.1.	<i>Định nghĩa và phân loại biểu thức</i> .....	58
4.3.2.	<i>Các toán tử</i> .....	58
4.4.	Các lệnh.....	60
4.4.1.	<i>Câu lệnh điều kiện</i> .....	60
4.4.2.	<i>Câu lệnh lặp</i> .....	62
4.4.3.	<i>Các câu lệnh thao tác trên đối tượng</i> .....	64
4.5.	Các hàm(Functions) .....	66
4.6.	Các hàm có sẵn.....	66
4.7.	Mảng(Array).....	68
4.8.	Các đối tượng trong JavaScript .....	70
4.8.1.	<i>Đối tượng window</i> .....	70
4.8.2.	<i>Đối tượng navigator</i> .....	72
4.8.3.	<i>Đối tượng location</i> .....	73
4.8.4.	<i>Đối tượng history</i> .....	73
4.8.5.	<i>Đối tượng document</i> .....	73
4.8.6.	<i>Đối tượng forms</i> .....	75
4.8.7.	<i>Đối tượng Math</i> .....	76
4.8.8.	<i>Đối tượng Date</i> .....	77
4.8.9.	<i>Đối tượng String</i> .....	78
CHƯƠNG 5:	NGÔN NGỮ HTML5 VÀ CSS3 .....	80
5.1.	Những khái niệm đầu tiên về HTML5 .....	80
5.1.1.	<i>Định nghĩa về HTML5</i> .....	80
5.1.2.	<i>Cú pháp của HTML5</i> .....	80
5.1.3.	<i>Các thành phần mới của HTML5</i> .....	80
5.1.4.	<i>HTML5 API (giao diện lập trình ứng dụng) và công nghệ hỗ trợ</i> .....	81
5.2.	Khởi tạo, làm việc với mã HTML5 và thành phần Form.....	81
5.2.1.	<i>Sử dụng ngôn ngữ đánh dấu HTML5</i> .....	81
5.2.2.	<i>Làm việc với các phần tử nội dung(content) của HTML5</i> .....	83
5.2.3.	<i>Tổng quan về các thành phần form mới của HTML5</i> .....	83
5.2.4.	<i>Làm việc với thành phần form mới trong HTML5</i> .....	84
5.3.	Làm việc với Javascript và query của HTML5 .....	86
5.4.	Làm việc với các thành phần Video, Audio và Canvas của HTML5 .....	90
5.5.	Làm việc CSS3.....	92
5.6.	Làm việc với thành phần mở rộng của CSS3 .....	96
TÀI LIỆU THAM KHẢO	.....	98



## DANH MỤC HÌNH VẼ

<i>Hình 1. Các trình duyệt web .....</i>	<b>2</b>
<i>Hình 2. Quá trình trao đổi dữ liệu với Web Server .....</i>	<b>3</b>
<i>Hình 3. Kiến trúc TCP/IP .....</i>	<b>4</b>
<i>Hình 4. Website và Webpage .....</i>	<b>5</b>
<i>Hình 5. HTTP và HTTPS.....</i>	<b>6</b>
<i>Hình 6. Tổng quan về HTML .....</i>	<b>8</b>
<i>Hình 7. Thuộc tính của thẻ body .....</i>	<b>11</b>
<i>Hình 8. Thiết lập tiêu đề và căn chỉnh .....</i>	<b>12</b>
<i>Hình 9. Sử dụng danh sách .....</i>	<b>15</b>
<i>Hình 10. Chèn ảnh, âm thanh và video .....</i>	<b>16</b>
<i>Hình 11. Siêu liên kết .....</i>	<b>18</b>
<i>Hình 12. Ví dụ về bảng .....</i>	<b>18</b>
<i>Hình 14 Ví dụ về form .....</i>	<b>26</b>

Hình 16. Model Box

*Hình 17. Kết quả thực hiện*

.....	42
<i>Hình 18. Mô hình hóa bằng thuộc tính margin</i>	
Hình 19 Ví dụ về float trong CSS	
Hình 20 Ví dụ về thuộc tính clear trong CSS .....	50
Hình 21. Ví dụ về Absolute position	
Hình 22. Ví dụ về relative position	
Hình 23. Hộp alert .....	55
Hình 24: Hiển thị cửa sổ nhập tên	
Hình 25. Kết hợp phương thức write và prompt .....	56
Hình 26. Kiểu dữ liệu .....	57
Hình 27. Kiểu dữ liệu null .....	58
Hình 28. Câu lệnh switch ... case .....	62
Hình 29: Kết quả của lệnh for...loop .....	62
Hình 30. Câu lệnh continue .....	64
Hình 31 Câu lệnh for ... in .....	64
Hình 32. Kết quả của ví dụ lệnh New .....	65
Hình 33: Kết quả của ví dụ lệnh with.....	66
Hình 34 Ví dụ hàm Eval	
Hình 35: Ví dụ parseInt .....	67
Hình 36: Ví dụ mảng .....	68
Hình 37. Ví dụ về thực hiện thuộc tính và phương thức của mảng.....	69
Hình 38: Minh hoạ cho đối tượng cửa sổ	
Hình 39: Ví dụ đối tượng Navigator .....	72
Hình 40. Ví dụ về đối tượng history.....	73
Hình 41. Kiểm soát dữ liệu số .....	76
Hình 42. Cách làm việc của form.....	83
Hình 43. Thành phần datalist	
Hình 44. Thành phần fieldset .....	86
Hình 45. Ví dụ về jquery	

Hình 46. Thao tác JQuery với HTML

Hình 47. Thao tác JQuery với CSS với thuộc tính background

Hình 48. Thuộc tính border-radius

Hình 49. Ví dụ về thuộc tính border-radius..... 94

Hình 50. Thuộc tính transform ..... 94

Hình 51. Thuộc tính transition

## **THUẬT NGỮ VIẾT TẮT**

HTML – HyperText Markup Language

WWW – World Wide Web

CSS – Cascading Style Sheet

FTP – File Transfer Protocol

HTTP – HyperText Transfer Protocol

ASP – Active Server Page

PHP - Hypertext Preprocessor

HTML5 - HyperText Markup Language version 5

CSS3 - Cascading Style Sheet version 3

IIS – Internet Information Service

JSP – Java Server Page

TCP/IP - Internet protocol suite (TCP/IP protocol suite)

URL – Uniform Resource Locator



## LỜI NÓI ĐẦU

Công nghệ World Wide Web giúp đưa thông tin mong muốn lên mạng Internet cho mọi người cùng xem thông qua các trang web. Đây là cách thức mà con người liên lạc, nghiên cứu thông tin, giải trí được sử dụng phổ biến nhất hiện nay. Vậy một trang web được xây dựng, tạo ra và duy trì như thế nào? Làm sao để chúng ta có thể đưa thật nhiều thông tin lên trên mạng Internet cho cả thế giới xem? Các vấn đề này sẽ được trình bày trong cuốn giáo trình “Thiết kế Web” để giúp bạn hiểu về cách thức xây dựng, trình bày một trang web. Và đây cũng là nền tảng cơ bản để bạn nghiên cứu chuyên sâu với các ngôn ngữ lập trình web nhằm tạo ra các trang web động.

Cấu trúc của giáo trình này bao gồm các 5 phần chính

- Chương 1: TỔNG QUAN VỀ MẠNG INTERNET
- Chương 2: NGÔN NGỮ ĐÁNH DẤU SIÊU VĂN BẢN HTML
- Chương 3: NGÔN NGỮ CSS
- Chương 4: NGÔN NGỮ KỊCH BẢN JAVASCRIPT
- Chương 5: NGÔN NGỮ HTML5 VÀ CSS3

Trong quá trình xây dựng chắc chắn giáo trình vẫn còn nhiều thiếu sót. Mong các bạn đọc góp ý, phê bình hoặc những nhận xét riêng của bạn về giáo trình này với mục tiêu nhằm nâng cao chất lượng của giáo trình. Chúng tôi xin cảm ơn và hứa sẽ tiếp thu để hoàn thiện tài liệu ngày càng tốt hơn.

Mọi thông tin xin liên hệ theo địa chỉ:

- Bộ môn Kỹ thuật và mạng máy tính - Khoa Công nghệ thông tin – Trường Đại học Công nghiệp Hà nội
- Hoặc địa chỉ e-mail: [phunt@fit-hau.edu.vn](mailto:phunt@fit-hau.edu.vn)

Chúc các bạn thành công!

### Nhóm tác giả

ThS. Nguyễn Trung Phú

ThS. Trần Phương Nhung

ThS. Đỗ Thị Minh Nguyệt

# CHƯƠNG 1: TỔNG QUAN VỀ MẠNG INTERNET

Hình thành năm 1969, từ 1 dự án của Bộ quốc phòng Mỹ, tên là mạng ARPAnet của Ban quản lý dự án quốc phòng. Đây là một mạng thử nghiệm phục vụ các nghiên cứu quốc phòng, một trong các mục đích là xây dựng một mạng máy tính có khả năng chịu đựng được các sự cố, mạng cũng cho phép một máy tính bất kỳ trên mạng liên lạc với mọi máy tính khác.

## 1.1. Lịch sử Web (World Wide Web - WWW)

World Wide Web, gọi tắt là Web hoặc WWW, mạng lưới toàn cầu là một không gian thông tin toàn cầu mà mọi người có thể truy nhập (đọc và viết) qua các máy tính nối với mạng Internet. Thuật ngữ này thường được hiểu nhầm là từ đồng nghĩa với chính thuật ngữ Internet. Nhưng Web thực ra chỉ là một trong các dịch vụ chạy trên Internet, chẳng hạn như dịch vụ thư điện tử. Web được phát minh và đưa vào sử dụng vào khoảng năm 1990, 1991 bởi viện sĩ Viện Hàn lâm Anh Tim Berners-Lee và Robert Cailliau (Bỉ) tại CERN, Geneva, Switzerland. Ngày 30 tháng 4 năm 1993, CERN thông báo rằng World Wide Web sẽ được miễn phí để sử dụng cho bất cứ ai.

World Wide Web là mạng lưới nguồn thông tin cho phép ta khai thác thông qua một số công cụ, chương trình hoạt động dưới các giao thức mạng. World Wide Web là công cụ, phương tiện hay đúng hơn là một dịch vụ của Internet.

Các tài liệu trên World Wide Web được lưu trữ trong một hệ thống siêu văn bản (hypertext), đặt tại các máy tính trong mạng Internet. Người dùng phải sử dụng một chương trình được gọi là trình duyệt web (web browser) để xem siêu văn bản. Chương trình này sẽ nhận thông tin (documents) tại ô địa chỉ (address) do người sử dụng yêu cầu (thông tin trong ô địa chỉ được gọi là tên miền (domain name), rồi sau đó chương trình sẽ tự động gửi thông tin đến máy chủ web (web server) và hiển thị trên màn hình máy tính của người xem. Người dùng có thể theo các liên kết siêu văn bản (hyperlink) trên mỗi trang web để nối với các tài liệu khác hoặc gửi thông tin phản hồi theo máy chủ trong một quá trình tương tác. Hoạt động truy tìm theo các siêu liên kết thường được gọi là duyệt Web.

Quá trình này cho phép người dùng có thể lướt các trang web để lấy thông tin. Tuy nhiên độ chính xác và chứng thực của thông tin không được đảm bảo

## 1.2. Cơ bản về trình duyệt Web

Trình duyệt web là một phần mềm ứng dụng cho phép người sử dụng xem và tương tác với các văn bản, hình ảnh, đoạn phim, nhạc, trò chơi và các thông tin khác ở trên một trang web của một địa chỉ web trên mạng toàn cầu hoặc mạng nội bộ. Văn bản và hình ảnh trên một trang web có thể chứa siêu liên kết tới các trang web khác của cùng một địa chỉ web hoặc địa chỉ web khác. Trình duyệt web cho phép người sử dụng truy cập các thông tin trên các trang web một cách nhanh chóng và dễ dàng thông qua



Hình 1. Các trình duyệt web

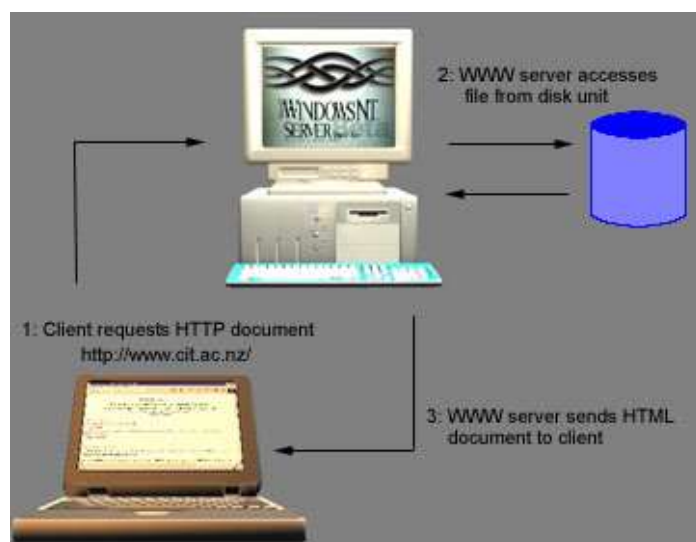
các liên kết đó. Trình duyệt web đọc định dạng HTML để hiển thị, do vậy một trang web có thể hiển thị khác nhau trên các trình duyệt khác nhau.

Một số trình duyệt web hiện nay cho máy tính cá nhân bao gồm Internet Explorer, Mozilla Firefox, Safari, Opera, Avant Browser, Konqueror, Lynx, Google Chrome, Flock, Arachne, Epiphany, K-Meleon, Cốc cốc .....

### 1.3. Web server (máy chủ Web)

Web Server (máy chủ Web): là máy tính mà trên đó cài đặt phần mềm phục vụ Web, đôi khi người ta cũng gọi chính phần mềm đó là Web Server. Tất cả các Web Server đều hiểu và chạy được các file \*.htm và \*.html, tuy nhiên mỗi Web Server lại phục vụ một số kiểu file chuyên biệt chẳng hạn như IIS của Microsoft dành cho \*.asp, \*.aspx...; Apache dành cho \*.php...; Sun Java System Web Server của SUN dành cho \*.p, \*.jsp...

Máy Web Server là máy chủ có dung lượng lớn, tốc độ cao, được dùng để lưu trữ thông tin như một ngân hàng dữ liệu, chứa những website đã được thiết kế cùng với những thông tin liên quan khác. (các mã Script, các chương trình, và các file Multimedia)



Hình 2. Quá trình trao đổi dữ liệu với Web Server

Web Server có khả năng gửi đến máy khách những trang Web thông qua môi trường Internet (hoặc Intranet) qua giao thức HTTP - giao thức được thiết kế để gửi các file đến trình duyệt Web (Web Browser), và các giao thức khác.

Tất cả các Web Server đều có một địa chỉ IP (IP Address) hoặc cũng có thể có một Domain Name. Giả sử khi gõ trên thanh địa chỉ (Address) của trình duyệt một dòng <http://www.abc.com> sau đó gõ phím Enter, trình duyệt sẽ gửi một yêu cầu đến một Server có Domain

Name là [www.abc.com](http://www.abc.com). Server này sẽ tìm trang Web có tên là index.htm (tệp tin mặc định được chỉ định bởi trình duyệt web) rồi sau đó gửi trả dữ liệu đến trình duyệt.

Bất kỳ một máy tính nào cũng có thể trở thành một Web Server bởi việc cài đặt lên nó một chương trình phần mềm Server Software và sau đó kết nối vào Internet. Khi máy tính kết nối đến một Web Server và gửi đến yêu cầu truy cập các thông tin từ một trang Web nào đó, Web Server Software sẽ nhận yêu cầu và gửi lại cho máy tính những thông tin mong muốn.

Giống như những phần mềm khác, Web Server Software cũng chỉ là một ứng dụng phần mềm. Nó được cài đặt, và chạy trên máy tính dùng làm Web Server, nhờ có chương trình này mà người sử dụng có thể truy cập đến các thông tin của trang Web từ một máy tính khác ở trên mạng (Internet, Intranet).

Web Server Software còn có thể được tích hợp với CSDL (Database), hay điều khiển việc kết nối vào CSDL để có thể truy cập và kết xuất thông tin từ CSDL lên các trang Web và truyền tải chúng đến người dùng.

#### 1.4. File Transfer Protocol – FTP

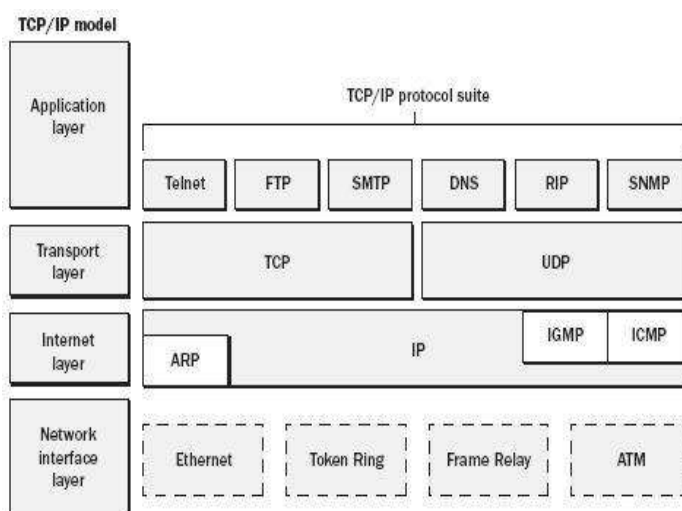
FTP (File Transfer Protocol) là một protocol trong mô hình TCP/IP được dùng để truyền các file giữa các máy. FTP cho phép truyền nhận file và quản lý trực tuyến. FTP không cho phép truy xuất một máy khác để thực thi chương trình, nhưng nó rất tiện lợi cho việc thao tác với file. Để sử dụng FTP thì các máy kết nối phải chạy các chương trình có hỗ trợ các dịch vụ về FTP. Client gọi đến server và thiết lập FTP thông qua một tập các lệnh bắt tay.

Hoạt động của FTP cần có hai máy tính, một máy chủ và một máy khách). Máy chủ FTP, dùng chạy phần mềm cung cấp dịch vụ FTP, gọi là trình chủ, lắng nghe yêu cầu về dịch vụ của các máy tính khác trên mạng lưới. Máy khách chạy phần mềm FTP dành cho người sử dụng dịch vụ, gọi là trình khách, thì khởi đầu một liên kết với máy chủ. Một khi hai máy đã liên kết với nhau, máy khách có thể xử lý một số thao tác về tập tin, như tải tập tin lên máy chủ, tải tập tin từ máy chủ xuống máy của mình, đổi tên của tập tin, hoặc xóa tập tin ở máy chủ v.v. Vì giao thức FTP là một giao thức chuẩn công khai, cho nên bất cứ một công ty phần mềm nào, hay một lập trình viên nào cũng có thể viết trình chủ FTP hoặc trình khách FTP. Hầu như bất cứ một nền tảng hệ điều hành máy tính nào cũng hỗ trợ giao thức FTP. Hiện nay trên thị trường có rất nhiều các trình khách và trình chủ FTP, và hầu hết các chương trình ứng dụng này cho phép người dùng được lấy tự do, không mất tiền có thể kể đến như ftpCute, Filezilla ...

#### 1.5. Giao thức TCP/IP

Bộ giao thức TCP/IP, (tiếng Anh: Internet protocol suite hoặc IP Suite hoặc TCP/IP protocol suite - bộ giao thức liên mạng), là một bộ các giao thức truyền thông cài đặt chồng giao thức mà Internet và hầu hết các mạng máy tính thương mại đang chạy trên đó. Bộ giao thức này được đặt tên theo hai giao thức chính của nó là TCP (Giao thức Điều khiển Giao vận) và IP (Giao thức Liên mạng). Chúng cũng là hai giao thức đầu tiên được định nghĩa.

Như nhiều bộ giao thức khác, bộ giao thức TCP/IP có thể được coi là một tập hợp các tầng, mỗi tầng giải quyết một tập các vấn đề có liên quan đến việc truyền dữ liệu, và cung cấp cho các giao thức tầng cấp trên một dịch vụ được định nghĩa rõ ràng dựa trên việc sử dụng các dịch vụ của các tầng thấp hơn. Về mặt logic, các tầng trên gần với người dùng hơn và làm việc với dữ liệu trừu tượng hơn, chúng dựa vào các giao thức tầng cấp dưới để biến đổi dữ liệu thành các dạng mà cuối cùng có thể được truyền đi một cách vật lý.



Hình 3. Kiến trúc TCP/IP

TCP/IP là bộ các giao thức có vai trò xác định quá trình liên lạc trong mạng và quan trọng hơn cả là định nghĩa “hình dạng” của một đơn vị dữ liệu và những thông tin chứa trong nó để máy tính đích có thể dịch thông tin một cách chính xác. TCP/IP và các giao thức liên quan tạo ra một hệ thống hoàn chỉnh quản lý quá trình dữ liệu được xử lý, chuyển và nhận trên một mạng sử dụng TCP/IP. Một hệ thống các giao thức liên quan, chẳng hạn như TCP/IP, được gọi là bộ giao thức.

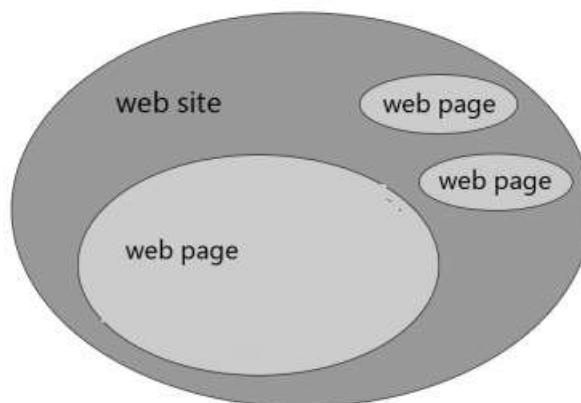
Thực tế của quá trình định dạng và xử lý dữ liệu bằng TCP/IP được thực hiện bằng bộ lọc của các hãng sản xuất. Ví dụ, Microsoft TCP/IP là một phần mềm cho phép Windows NT xử lý các dữ liệu được format theo TCP/IP và vì thế có thể hoà vào mạng TCP/IP.

Một chuẩn TCP/IP là một hệ thống các quy định quản lý việc trao đổi trên các mạng TCP/IP. Bộ lọc TCP/IP là một phần mềm có chức năng cho phép một máy tính hoà vào mạng TCP/IP.

Mục đích của các chuẩn TCP/IP là nhằm đảm bảo tính tương thích của tất cả bộ lọc TCP/IP thuộc bất kỳ phiên bản nào hoặc của bất kỳ hãng sản xuất nào.

### 1.6. Trang Web (Webpage)

Khi họ nói về một trang web (Webpage), là chúng ta đang đề cập đến một phần nhỏ của website được thu hẹp với những nội dung cụ thể, hay nói đơn giản hơn trang web có thể là một bài viết, một trang tin bất kỳ. Và nếu tập hợp nhiều trang web trên cùng một tên miền thì ta có một Website. Khi mọi người tham khảo một website, họ đang nói về một vùng lớn với một lượng lớn thông tin khác nhau. Về nội dung – Một website có thể chứa nhiều thông tin dành cho nhiều người cho nhiều mục đích và nhu cầu khác nhau. Tuy nhiên một trang web chỉ phục vụ cho một mục đích và/hoặc một nhu cầu cụ thể



Hình 4. Website và Webpage

Website là một tập hợp các văn bản, đồ họa, đa phương tiện và nó luôn luôn được chứa đựng trong một tên miền chính hoặc một tên miền phụ. Có thể hiểu website là một văn phòng ảo đối với một công ty bởi vì nó hoạt động liên tục không nghỉ nhằm cung cấp thông tin cho người dùng từ mọi nơi trên thế giới. Nếu như một văn phòng thực sự cần có mặt bằng, thiết bị, nhân viên thì trang web cần có tên miền, nơi lưu trữ web và một đường truyền mạng internet.

### 1.7. Hypertext Transfer Protocol - HTTP

HTTP được coi là một giao thức trong bộ giao thức TCP/IP và chịu trách nhiệm hiển thị nội dung và truyền tải các tệp HTML, HTM, ASP, ASPX, ....

HTTP là chữ viết tắt của HyperText Transfer Protocol (giao thức truyền tải siêu văn bản). HTTP hoạt động dựa trên mô hình Client – Server. Trong mô hình này, các máy tính của người dùng sẽ đóng vai trò làm máy khách (Client). Sau một thao tác nào đó của người dùng, các máy khách sẽ gửi yêu cầu đến máy chủ (Server) và chờ đợi câu trả lời từ những máy chủ này.

Khi gõ một địa chỉ Web URL vào trình duyệt Web, một lệnh HTTP sẽ được gửi tới Web server để ra lệnh và hướng dẫn nó tìm đúng trang Web được yêu cầu. Trang Web này sau đó sẽ được tải về và mở trên trình duyệt Web tại máy Client.

Được sử dụng một cách rộng rãi là vậy nhưng HTTP chứa đựng trong nó không ít những điểm hạn chế. Khi chúng ta tiến hành thực hiện việc truy nhập vào một trang



Hình 5. HTTP và HTTPS

Web thông qua giao thức HTTP, trình duyệt sẽ thực hiện các phiên kết nối đến Server của trang Web đó thông qua địa chỉ IP do hệ thống phân giải tên miền DNS cung cấp.

Trong quá trình kết nối và trao đổi thông tin, trình duyệt của chúng ta sẽ mặc nhiên thừa nhận địa chỉ IP đó đến từ Server của chính Website mà chúng ta muốn truy nhập mà không hề có biện

pháp xác thực nào. Các thông tin được chuyển đi qua giao thức HTTP (bao gồm địa chỉ IP, các thông tin nhập liệu trên Website...) cũng không hề được mã hóa và bảo mật.

Điều này dẫn đến những nguy cơ về việc phiên kết nối tới máy chủ của Website có thể bị “nghe lén”, hoặc việc truy nhập bị chuyển hướng đến một trang Web giả danh với thiết kế giống hệt Website gốc mà người sử dụng không hề hay biết.

Nếu thường xuyên sử dụng các dịch vụ ngân hàng trực tuyến, chúng ta sẽ thấy khi truy nhập vào địa chỉ tên miền của ngân hàng, giao thức mà nó sử dụng sẽ là HTTPS thay vì HTTP như ở những trang web thông thường. Điều này là bởi, phiên bản nâng cấp HTTPS của HTTP được sử dụng nhằm tăng cường khả năng bảo mật thông tin sau mỗi lần truy nhập.

HTTPS là tên viết tắt của "Hypertext Transfer Protocol Secure". Đây là một sự kết hợp giữa giao thức HTTP và giao thức bảo mật SSL hay TLS. HTTPS giúp cho việc trao đổi thông tin một cách bảo mật trên nền Internet.

Khác với HTTP, HTTPS sẽ hỗ trợ việc xác thực tính chính danh của các Website mà người dùng truy nhập thông qua việc kiểm tra xác thực bảo mật (security certificate). Các xác thực bảo mật này được cung cấp và xác minh bởi các CA (Certificate Authority) có uy tín. Với các xác thực từ CA, người sử dụng có thể biết rằng mình đã truy nhập đúng vào Website cần truy nhập chứ không phải một Website giả danh bất kỳ nào khác.

Bên cạnh đó, các phiên kết nối giữa trình duyệt đến Server đều sẽ được mã hóa. Điều này sẽ giúp che giấu địa chỉ IP và những thông tin nhập liệu về tài khoản trên Website khỏi sự nhòm ngó của các hacker. HTTPS không đem đến sự an toàn 100%. Tuy vậy, đây là biện pháp bảo mật hữu hiệu thay vì việc sử dụng giao thức HTTP truyền thông vốn đầy rủi ro sẵn có.

Rõ ràng việc sử dụng giao thức HTTPS giúp tăng cường khả năng bảo mật và phòng vệ đáng kể cho người dùng internet. Cũng chính bởi điều này, các hệ thống ngân hàng, tổ chức tín dụng... đều sử dụng giao thức HTTPS trên các Website của mình.



## 1.8. Hyperlink

Hyperlink là một kết nối từ một trang web này để đến trang web khác trên WWW. Dùng để liên kết nhiều trang Web với nhau thành một Web site. Trong Web browser thẻ liên kết được hiển thị như một định dạng đặc biệt. Các trình duyệt thường gạch chân văn bản có hyperlink và hiển thị chúng theo một màu chỉ định nếu như người lập trình không tương tác trên hyperlink này.

Ngoài ra, còn một số người còn hiểu hyperlink như một **URL** (Uniform Resource Locator) nơi xác định vị trí của một tài nguyên mạng. Một URL bao gồm: *giao thức, tên máy chủ, đường dẫn đến tài nguyên mạng*.

**HTTP** xác định một file sử dụng giao thức HTTP trên máy chủ Web Server.

Ví dụ như địa chỉ: <http://www.fit-hauai.edu.vn/index.html> thì ta có thể thấy **http** là giao thức, **fit-hauai.edu.vn** là tên miền (được trỏ đến một máy chủ chứa trang web) và **index.html** là đường dẫn trên máy chủ.

**FTP** xác định file sử dụng giao thức truyền FTP và file đó nằm trên máy chủ FTP. Ví dụ: <FTP://ftp.microsoft.com/file.doc>

**Mailto** xác định truyền e-mail. Nó xác định một địa chỉ e-mail nào đó. Ví dụ: <mailto:phunt@fit-hauai.edu.vn>

**file** xác định một file lưu trữ ở trên một máy tính hoặc một máy tính nào đó trên mạng. Ví dụ: <File:///dir/file.doc>

**TỔNG KẾT CHƯƠNG 1:** Chương 1 đã giới thiệu đầy đủ các khái niệm cũng như lịch sử hình thành lên nền tảng Internet và đặc biệt là mối quan hệ giữa các thành phần hình thành lên website như trình duyệt web, máy chủ web, các giao thức và các liên kết tồn tại trên các trang web.

### BÀI TẬP:

1. So sánh sự khác nhau giữa giao thức HTTP và HTTPS? Hãy giải thích tại sao các hệ thống lớn cần sử dụng giao thức HTTPS
2. Hãy trình bày mối quan hệ giữa web server và trình duyệt web?

## CHƯƠNG 2: NGÔN NGỮ ĐÁNH DẤU SIÊU VĂN BẢN HTML

Trên thực tế, mỗi ứng dụng Web đều tồn tại hai loại trang Web đó là trang web tĩnh và trang web động. Trang web tĩnh, thông thường là trang web không kết nối cơ sở dữ liệu, điều đó có nghĩa là chúng thiết kế bằng các thẻ HTML và kịch bản tại trình khách (Client Script). Ngược lại, trang web động được thiết kế bằng kịch bản tại trình chủ (Server Script) và có thể được kết nối tới cơ sở dữ liệu. Tuy nhiên để một trang web trở thành một ứng dụng chuyên nghiệp thì ứng dụng đó phải kết nối với cơ sở dữ liệu với mục tiêu làm tươi mới dữ liệu trên trang web.

Trong chương này, trình bày về các kiến thức nền tảng của HTML với mục tiêu trình bày các văn bản, đồ họa đa phương tiện trên trình duyệt web.

### 2.1. Tổng quan về HTML

Tim Berners-Lee là người phát minh ra Web, ý tưởng này xuất hiện vào **năm 1989** khi Tim làm việc tại phòng dịch vụ điện toán ở CERN. Các nghiên cứu Vật lý thường yêu cầu sự hợp tác của các nhà khoa học khắp nơi trên thế giới, và để tiện cho việc tra cứu tài liệu, ý tưởng của Tim là tạo ra một nơi để mọi người có thể kết nối vào và từ đó xem các tài liệu khác thông qua các liên kết. Ý tưởng này là đột phá, thông thường, mọi người phải truy cập rồi tải tất cả tài liệu về đọc, nhưng Tim muốn rằng các tài liệu được liên kết chéo với nhau.

Sau này, với sự phát triển của Web, HTML nhanh chóng được nâng cấp, HTML 3.2 được đưa ra vào năm **1997**, sau đó mùa xuân năm **1998** là HTML 4.0

Cùng với sự phát triển của Internet, HTML 4.0 trở nên già cỗi, W3C tiếp tục đưa ra phiên bản 4.1 và 4.2 được gọi là XHTML (HTML + XML). Tuy nhiên, lúc này HTML trở nên khó hiểu hơn so với ban đầu vì thế một nhóm nghiên cứu khác đã tạo ra HTML5, đơn giản hơn, mạnh mẽ hơn. Sau này W3C đã chọn HTML5 làm tiêu chuẩn cho Web



Hình 6. Tổng quan về HTML

**HTML** viết tắt bởi cụm từ HyperText Markup Language (ngôn ngữ đánh dấu siêu văn bản) sử dụng các thẻ tag để trình bày các văn bản, đồ họa, đa phương tiện trên trình duyệt web.

### 2.2. Cấu tạo thẻ của html và cấu trúc trang HTML

#### 2.2.1. Cấu trúc thẻ HTML

Phần tử HTML hay còn được biết đến với tên Tag hay Entity hay thẻ. HTML là ngôn ngữ đánh dấu (markup-language) do đó có thể hiểu một phần tử HTML chính là một đoạn văn bản được đánh dấu để thể hiện theo một cách nào đó.

Một phần tử HTML luôn có thể được nhận ra bởi nó được bao quanh bởi cặp dấu < và > Ví dụ: <body> hay <b>



Thẻ HTML được chia ra làm 02 loại thẻ: thẻ chứa dữ liệu và thẻ rỗng

### ➤ Thẻ chứa dữ liệu

Mỗi phần tử của HTML luôn bao gồm một cặp thẻ đi song song, một thẻ mở và một thẻ đóng, hai thẻ này còn được gọi là thẻ chứa vì chức năng của thẻ sẽ tác động vào các thành phần nó chứa bên trong thẻ đóng và thẻ mở.

Cấu trúc thẻ mở có dạng: **<ten\_thẻ>**

Còn thẻ đóng cũng tương tự nhưng có thêm dấu gạch chéo: **</ten\_thẻ>**

Ví dụ: `<p>đây là nội dung của thẻ "p"</p>`

Trong thẻ HTML có thể có thêm các thuộc tính, thuộc tính của một phần tử có thể được biểu diễn ngay trong thẻ mở của phần tử.

Cú pháp thuộc tính của thẻ mở:

**<ten\_thẻ thuộc\_tính\_1="giá\_trị\_1" thuộc\_tính2="giá\_trị\_2" .... >**

Trong thẻ mở của ngôn ngữ HTML có thể có nhiều thuộc tính với các thuộc tính sẽ thể hiện thêm các chức năng cụ thể của thẻ như thuộc tính class, id, color, ....

Ví dụ: `<p id="tagline"></p>`.

### ➤ Thẻ rỗng

Thẻ rỗng là thẻ mà trong đó chỉ tồn tại thẻ mở và không có thẻ đóng, đối với thẻ này thì chức năng của thẻ sẽ thực hiện ngay tại vị trí mà được đặt thẻ. Khi đó để đảm bảo tính hợp lệ của HTML ta cần đặt thêm một dấu gạch chéo ở cuối thẻ mở theo cấu trúc: **<ten\_thẻ các\_thuộc\_tính />**

Ví dụ: ``

**Chú ý:** Một số thẻ có nhiều thuộc tính tùy chọn. Với thẻ chứa dữ liệu nếu có thuộc tính thì thuộc tính được liệt kê trong thẻ mở, còn thẻ đóng thì không.

#### 2.2.2. Cấu trúc tài liệu HTML

**<HTML>**

**<HEAD>**

**<TITLE>** Tiêu đề trang **<TITLE>**

//Các thẻ thiết lập cấu trúc

**</HEAD>**

**<BODY>**

.....

Các thẻ HTML xây dựng nội dung trang Web

.....

**</BODY>**

**</HTML>**

Một tài liệu HTML gồm 2 phần riêng biệt

- Phần đầu: **<HEAD > ... </HEAD>**: Chứa các thông tin về tài liệu và không được hiển thị trên màn hình.
- Phần thân **<BODY> ... </BODY>**: Chứa nội dung của trang web được hiển thị trên màn hình.

### 2.2.3. Ý nghĩa của các thẻ trong cấu trúc trang Web

**HTML:** Thẻ này được sử dụng để xác nhận một tài liệu là tài liệu HTML, tức là nó có sử dụng các thẻ HTML để trình bày. Toàn bộ nội dung của tài liệu được đặt gần cặp thẻ này.

Cú pháp:

**<HTML>**

...Toàn bộ nội dung của tài liệu được đặt ở đây.

**</HTML>**

Trình duyệt sẽ xem các tài liệu không sử dụng thẻ **<HTML>** như những tệp văn bản bình thường.

**HEAD:** Thẻ Head được dùng để xác định phần mở đầu cho tài liệu và các thành phần trên trình duyệt như thanh tiêu đề, thanh trạng thái hoặc là bộ phong chữ hỗ trợ trong trang web

Cú pháp:

**<HEAD>**

...Phần mở đầu (**HEADER**) của tài liệu được đặt ở đây

**</HEAD>**

**TITLE:** Thẻ này chỉ có thể sử dụng trong phần mở đầu của tài liệu, tức là nó phải nằm trong thẻ phạm vi giới hạn bởi cặp thẻ **<HEAD>**.

Cú pháp:

**<TITLE>**Tiêu đề của tài liệu**</TITLE>**

**BODY:** Thẻ này được sử dụng để xác định phần nội dung chính của tài liệu-phần thân (body) của tài liệu. Trong phần thân có thể chứa các thông tin định dạng nhất định để đặt ảnh nền cho tài liệu, màu nền, màu văn bản siêu liên kết, đặt lề cho trang tài liệu, những thông tin này được thể hiện bởi các thuộc tính của thẻ.

Cú pháp:

**<BODY>**

....Phần nội dung của tài liệu được đặt ở đây.

**</BODY>**

Trên đây là cú pháp cơ bản của thẻ BODY, tuy nhiên bắt đầu từ HTML thì có nhiều thuộc tính được sử dụng trong thẻ BODY. Sau đây là các thuộc tính chính:

THUỘC TÍNH	Ý NGHĨA
<b>BACKGROUND</b>	Đặt một ảnh nào đó làm ảnh nền (background) cho văn bản. Giá trị của tham số này (phần sau dấu bằng) là URL của tệp tin dạng ảnh. Nếu kích thước ảnh nhỏ hơn cửa sổ trình duyệt thì toàn bộ màn hình cửa sổ trình duyệt sẽ được phủ kín bằng nhiều ảnh.
<b>BGCOLOR</b>	Đặt màu nền cho trang khi hiển thị. Nếu cả hai tham số <b>BACKGROUND</b> và <b>BGCOLOR</b> cùng có giá trị thì trình duyệt sẽ hiển thị màu nền trước, sau đó mới tải

THUỘC TÍNH	Ý NGHĨA
	ảnh lên phía trên.
<b>TEXT</b>	Xác định màu chữ của văn bản, kể cả các đề mục.
<b>ALINK, VLINK, LINK</b>	Xác định màu sắc cho các siêu liên kết trong văn bản. Tương ứng, alink ( <i>active link</i> ) là liên kết đang được kích hoạt - tức là khi đã được kích chuột lên; vlink ( <i>visited link</i> ) chỉ liên kết đã từng được kích hoạt;

**Ví dụ:**

```
<html>
  <body
    background="A1.png"
    text="#FFFFFF">
    Kiểm tra
  </body>
</html>
```



Hình 7. Thuộc tính của thẻ body

## 2.3. Các thẻ trình bày trang web

### 2.3.1. Thẻ ngắt dòng và khai báo đoạn

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<P>	</P>	Khai báo một đoạn văn đồng thời chèn một kí tự xuống dòng và một dòng trống vào sau.
 		Ngắt xuống dòng.

### 2.3.2. Các thẻ đề mục

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<H1>	</H1>	Định dạng dòng văn bản theo các cấp đề mục từ 1 đến 6
<H2>	</H2>	
<H3>	</H3>	
<H4>	</H4>	
<H5>	</H5>	
<H6>	</H6>	

## Căn lề cho đoạn văn bản

Sử dụng thuộc tính Align để căn lề cho đoạn văn bản, cho hình ảnh, cho đường kẻ ngang như sau:

THUỘC TÍNH	Ý NGHĨA
Align="center"	Căn giữa trang.
Align="left"	Căn theo lề trái.
Align="right"	Căn theo lề phải
Align="justify"	Căn đều.

**Ví dụ:**

```
<html>
<body
background="1586868.jpg"
text="#FFFFFF">
<h1>Tiêu đề H1 </h1>
<h2>Tiêu đề H2 </h2>
<h3>Tiêu đề H3 </h3>
<h4>Tiêu đề H4 </h4>
<h5>Tiêu đề H5 </h5>
<h6>Tiêu đề H6 </h6>
<h1 align="center">Tiêu
đề H1 có căn giữa </h1>
<h1 align="right">Tiêu đề
H1 có căn phải </h1>
</body>
</html>
```



Hình 8. Thiết lập tiêu đề và căn chỉnh

## 2.4. Các thẻ định dạng kí tự

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<B>	</B>	Nội dung giữa thẻ được in đậm.
<I>	</I>	Nội dung giữa thẻ được in nghiêng.
<U>	</U>	Nội dung giữa thẻ được gạch chân.
<STRIKE>	</STRIKE>	Nội dung giữa thẻ được gạch giữa.
<CENTER>	</CENTER>	Dùng để căn giữa văn bản,
<BLINK>	</BLINK>	Thẻ tạo chữ nhấp nháy, chỉ có tác dụng trong trình duyệt Netscape.
<SUB>	</SUB>	Tạo chỉ số dưới.
<SUP>	</SUP>	Tạo chỉ số trên.
<Basefont size="1-7" >		Font chuẩn cho tài liệu HTML không có thẻ đóng, nếu không có giá trị cụ thể, trình duyệt IE sẽ có size bằng 3.

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<Font	</Font>	
Size="n"		Định cỡ chữ (n=1-7)
Color="tên màu" hoặc "#mã màu"		Định màu cho chữ đặt giữa thẻ, tên màu viết bằng tiếng anh
Face="tên font">		Xác định font chữ để hiển thị văn bản trên trình duyệt.
<SUB>	</SUB>	Tạo chỉ số dưới.
<SUP>	</SUP>	Tạo chỉ số trên.

## 2.5. Thẻ Body và các thuộc tính

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<Body Bgcolor="tên màu" hoặc "#mã màu" Background="url" Text="tên màu" hoặc "#mã màu" Link="tên màu" hoặc "#mã màu"  Alink="tên màu" hoặc "#mã màu" Vlink="tên màu" hoặc "#mã màu">	</Body>	Đặt màu nền cho trang web.  Đặt ảnh nền cho trang web. Đặt màu cho toàn bộ nội dung văn bản trong trang web. Đặt màu cho liên kết chưa sử dụng ngầm định là màu xanh da trời (Blue). Đặt màu cho các liên kết đang hoạt động ngầm định là màu đỏ (Red). Đặt màu cho các liên kết đã sử dụng ngầm định là màu tím (Purple).

## 2.6. Các thẻ tạo danh sách

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<OL>	</OL>	Khai báo một danh sách theo thứ tự.
<UL>	</UL>	Khai báo một danh sách không theo thứ tự.
<DL>	</DL>	Khai báo một danh sách do người dùng tự định nghĩa.
<LI>		Đây là thẻ định nghĩa thành phần của danh sách.

Thuộc tính của thẻ tạo danh sách theo thứ tự OL

**Cú pháp:** <OL type="kiểu đánh thứ tự">...</OL>

Trong đó:

KIỂU ĐÁNH THỨ TỰ	Ý NGHĨA
I	Đánh thứ tự theo số La mã I, II, ...
1	Đánh thứ tự theo số thập phân 1, 2, 3...
a	Đánh thứ tự theo chữ thường a, b..
A	Đánh thứ tự theo chữ hoa A, B...

Thuộc tính của thẻ tạo danh sách không theo thứ tự UL

**Cú pháp:** <UL type="Kiểu hiển thị">...</UL>

Trong đó:

KIỂU HIỂN THỊ	Ý NGHĨA
Square	Hiển thị Bullet hình vuông đầy.
Circle	Hiển thị Bullet dạng hình tròn.
Disc	Sử dụng Bullet mặc định.

Thẻ tạo danh sách tự định nghĩa

Danh sách này được sử dụng để tạo một danh sách các thuật ngữ và định nghĩa các thuật ngữ đó.

<DL>
<DT>Thuật ngữ
<DD>Diễn giải</DD>
</DT>
</DL>

Trong đó:

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<DT>	</DT>	Nêu các thuật ngữ.
<DD>	</DT>	Định nghĩa các thuật ngữ.

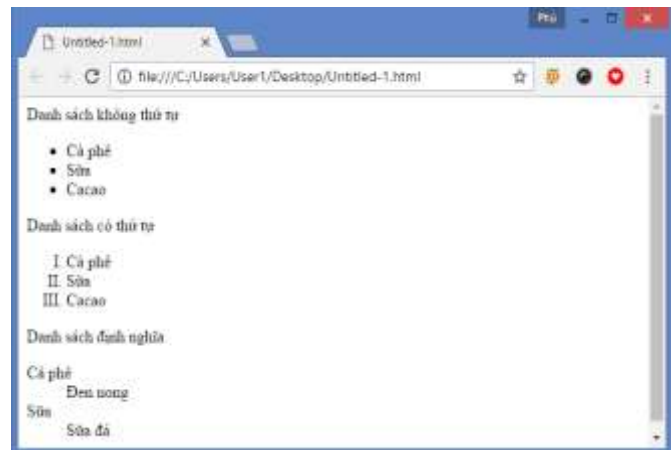
**Ví dụ:**

```
<html>
<body>
    Danh sách không thứ tự
    <ul type=" square">
        <li>Cà phê</li>
        <li>Sữa</li>
        <li>Cacao</li>
```

```

</ul>
Danh sách có thứ
tự
<ol type="I">
    <li>Cà phê</li>
    <li>Sữa</li>
    <li>Cacao</li>
</ol>
Danh sách định
nghĩa
<dl>
    <dt>Cà phê</dt>
    <dd>Đen nong
</dd>
    <dt>Sữa</dt>
    <dd>Sữa đá</dd>
</dl>
</body>
</html>

```



Hình 9. Sử dụng danh sách

## 2.7. Chèn ảnh, âm thanh và video

### 2.7.1. Cú pháp thẻ IMG chèn hình ảnh vào trang web

Ảnh ở đây có thể là các biểu tượng, bullet, ảnh, logo công ty...Ngày nay có nhiều định dạng ảnh đang được sử dụng. Ba định dạng đồ họa thông thường được hiện thị trên hầu hết các trình duyệt là:

- GIF (Graphics Interchange Format-ảnh \*.gif)
- JPEG (Joint Photographic Expert Group- ảnh \*.jpg)
- PNG (Portable Network Graphics- ảnh \*.png)

**Cú pháp: <IMG src="Đường dẫn đến file ảnh">**

Ảnh cũng có thể đóng vai trò là một siêu liên kết bằng cách dùng cặp thẻ <a> chứa thẻ IMG theo cấu trúc như sau:

**<A href=""><IMG src=""></A>**

Ý nghĩa các thuộc tính trong thẻ IMG

THUỘC TÍNH	Ý NGHĨA
Align="alignment"	Căn lề cho ảnh hay cho đoạn văn bản bao quanh ảnh (giá trị cụ thể là: TOP, BOTTOM, MIDDLE, LEFT, RIGHT)
Alt="Text"	Đoạn văn bản hiển thị thay thế cho hình ảnh khi chức năng hiển thị bị lỗi trong trình duyệt.
Src="url"	Chỉ định đường dẫn tới hình ảnh.
Border="n"	Xác định đường viền ảnh. Nếu ảnh đóng vai trò là một siêu

THUỘC TÍNH	Ý NGHĨA
	liên kết thì đường viền ảnh có màu trùng với màu của siêu liên kết còn nếu ảnh không đóng vai trò là siêu liên kết thì đường viền ảnh không hiển thị.
Height="x"	Xác định chiều cao của ảnh.
Width="y"	Xác định chiều rộng của ảnh.
Hspace="n" (đơn vị là pixel)	Xác định khoảng cách từ ảnh đến văn bản xung quanh nó theo chiều ngang (trái, phải).
Vspace="n" (đơn vị là pixel)	Xác định khoảng cách từ ảnh đến văn bản xung quanh nó theo chiều dọc (trên, dưới).

### 2.7.2. Cú pháp thẻ EMBED chèn hình âm thanh và video vào trang web

Âm thanh và video ở đây có thể là tệp tin thuộc nhóm đa phương tiện được sử dụng phổ biến hiện nay. Ngày nay có nhiều định dạng âm thanh và video đang được sử dụng và hầu hết các tệp tin định dạng khi thể hiện trên trình duyệt web luôn được chạy mặc định bằng những phần mềm, ứng dụng hỗ trợ cho nó. Có thể kể đến đó là Windows Media, VLC, Flash Player .... tuy nhiên đối với việc đưa ứng dụng đa phương tiện lên trình duyệt có thể sử dụng nhiều phương thức khác nhau.

Với công nghệ HTML 4.0 thì các trình duyệt thường sử dụng ứng dụng mặc định của hệ điều hành như khi chạy windows thì ứng dụng là Window Media.

**Cú pháp: <Embed src="Đường dẫn đến file âm thanh/video">**

**Ví dụ:**

```
<html><body><center>
    
    
    <embed
src="Beethoven's
Symphony No. 9
(Scherzo).wma"><br>
    <embed src="Picture
130.avi">
</center></body></html>
```



Hình 10. Chèn ảnh, âm thanh và video



### 2.7.3. Span & div

2.7.4. **Thẻ <span>:** trong HTML là một thẻ trung hòa, nó không thêm hay bớt bất cứ một thứ gì vào một tài liệu HTML cả.

2.7.5. **Nhóm phần tử với <div>:** Cũng như <span>, <div> cũng là một thẻ trung hòa và được thêm vào tài liệu HTML với mục đích nhóm các phần tử lại cho mục đích định dạng bằng CSS. Tuy nhiên, điểm khác biệt là <span> dùng để nhóm một khối phần tử trong khi đó <div> có thể nhóm một hoặc nhiều khối phần tử.

## 2.8. Siêu liên kết

Liên kết (Link) là một đặc trưng của World Wide Web. Chúng cho phép chuyển từ trang này đến trang khác, và tải tập tin về bằng FTP (File transfer Protocol)

Các loại liên kết bao gồm liên kết trong và liên kết ngoài:

**Liên kết trong:** Là liên kết với các phần trong cùng một tài liệu hoặc cùng một Website. Có thể sử dụng điểm neo cho phép người dùng di chuyển đến các phần khác nhau của tài liệu

+ Đặt tên cho điểm neo bằng cú pháp sau:

`<A name="tên neo">Vị trí đặt neo</A>`

+ Để xây dựng liên kết trong ta thực hiện cú pháp sau:

`<A HREF="#tên neo">Tên siêu liên kết</A>`

+ Để Tạo liên kết đến một điểm neo cụ thể trong một tài liệu khác ta có thể dùng cú pháp như sau:

`<A HREF="URL#tên neo">Tên siêu liên kết</A>`

URL (Uniform Resource Locator) là đường dẫn tới trang hay tệp tin có chứa điểm neo cần liên kết đến.

**Liên kết ngoài:** Là các liên kết với các trang tài liệu khác.

`<A HREF="URL">Tên siêu liên kết</A>`

Thuộc tính HREF được sử dụng để chỉ đường dẫn tới tài liệu hoặc tệp tin được liên kết đến.

**Liên kết đến địa chỉ Mail:** Cho phép người dùng gửi mail từ trang web của bạn. Để liên kết đến địa chỉ mail bạn dùng cú pháp như sau:

`<A HREF="mailto:addmail@mymail.com">Địa chỉ Email</A>`

Tạo siêu liên kết là việc tạo một địa chỉ đầy đủ, hoặc URL của file được nối đến. Tên siêu liên kết có thể là một dòng văn bản hoặc thậm chí một ảnh. Khi người dùng nhấp vào điểm có hiệu ứng, trình duyệt đọc địa chỉ được xác định trong URL và chuyển đến vị trí mới

Các dạng URL bao gồm URL tuyệt đối và URL tương đối. Trong đó:

**URL tuyệt đối:** Là địa chỉ đầy đủ của một trang hoặc một file, bao gồm giao thức, vị trí mạng, đường dẫn tùy chọn và tên file. Ví dụ: `http://edu.net.vn/index.html` là một URL tuyệt đối.

**URL tương đối:** Là một URL với một hoặc nhiều phần bị thiếu, thường được thể hiện một thành phần trên cùng một địa chỉ máy chủ. Ví dụ: `/img/anh1.jpg`

Để gán phím tắt cho siêu liên kết ta có thể thực hiện theo cú pháp như sau:

`<A HREF="URL" ACCESKEY="Phím gõ tắt">Tên liên kết</A>`

Để chỉ định thứ tự các Tab cho liên kết ta sử dụng cú pháp như sau :

`<A HREF="URL" TABINDEX="n">Tên liên kết</A>`

Với n là số tự nhiên chỉ định thứ tự TAB.

Ví dụ :

```
<html><body><center>
    <a href="#">Liên kết 1</a><br>
    <a href="trang1.htm">Liên kết
2</a><br>
    <a href="#">Liên kết 3</a><br>
    <a href="#">Liên kết 4</a><br>
    <a href="#">Liên kết 5</a><br>
    <a
href="http://www.hau1.edu.vn/index.html"> Liên kết 6
</a>
</center></body></html>
```

[Liên kết 1](#)  
[Liên kết 2](#)  
[Liên kết 3](#)  
[Liên kết 4](#)  
[Liên kết 5](#)  
[Liên kết 6](#)

Hình 11. Siêu liên kết

## 2.9. Bảng

### 2.9.1. Các thẻ dùng tạo bảng

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<code>&lt;TABLE&gt;</code>	<code>&lt;/TABLE&gt;</code>	Thẻ khai báo bảng.
<code>&lt;TR&gt;</code>	<code>&lt;/TR&gt;</code>	Thẻ khai báo một hàng của bảng.
<code>&lt;TD&gt;</code>	<code>&lt;/TD&gt;</code>	Thẻ khai báo một cột của bảng.
<code>&lt;TH&gt;</code>	<code>&lt;/TH&gt;</code>	Khai báo tiêu đề hàng và tiêu đề cột của bảng, các tiêu đề sẽ được định dạng chữ đậm và căn giữa.

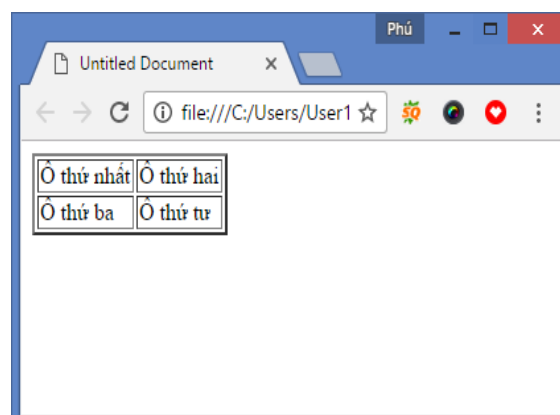
Ví dụ: Để tạo một bảng gồm 2 dòng 2 cột ta sử dụng cấu trúc như sau:

`<HTML>`

`<Head><Title>Ví dụ chương 4 phần 1</Title></Head>`

`<Body>`

```
<Table border=2>
<tr>
    <td>Ô thứ nhất</td>
    <td>Ô thứ hai</td>
</tr>
<tr>
    <td>Ô thứ ba</td>
    <td>Ô thứ tư</td>
```



Hình 12. Ví dụ về bảng

</tr></Table></Body></HTML>

### 2.9.2. Thẻ Table và thuộc tính

Trong thẻ Table có các thuộc tính bao gồm:

THUỘC TÍNH	Ý NGHĨA
Align="alignment"	Sử dụng để căn chỉnh bảng trên trang web.
Width="n"	Xác định chiều rộng của toàn bảng
Background="url"	Chỉ định ảnh nền cho bảng.
Bgcolor="Color"	Chỉ định màu nền cho bảng.
Border="n"	Xác định độ rộng đường viền bảng.
Bordercolor="Color"	Xác định màu được chọn cho khung viền của bảng
Cellpadding="n"	Xác định độ rộng đường bao xung quanh bảng.
Cellspacing="n"	Xác định độ rộng các đường kẻ dọc ngang trong bảng.

### 2.9.3. Thẻ TR và thuộc tính

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<TR		
Align="alignment"	</TR>	Canh lề theo chiều ngang nội dung của tất cả các ô trong một hàng: các giá trị có thể nhận left, right, center.
Valign="Valignment"		Canh lề theo chiều dọc nội dung của tất cả các ô trong một hàng: các giá trị có thể nhận top, bottom, middle.
Background="url"		Chỉ định ảnh nền cho hàng.
Bgcolor="Color"		Chỉ định màu nền cho hàng.
>		

### 2.9.4. Thẻ TD và thuộc tính

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<TD	</TD>	
Align="alignment"		Căn lề theo chiều ngang nội dung một ô theo đường viền trái phải: các giá trị có thể nhận left, right, center.
Valign="Valignment"		Căn lề theo chiều dọc nội dung một ô

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
		theo đường viền trên dưới: các giá trị có thể nhận top, bottom, middle.
Background="url"		Chỉ định ảnh nền cho ô đó.
Bgcolor="Color"		Chỉ định màu nền cho ô đó.
Width="n"		Xác định chiều rộng của ô đó.
Height="m"		Xác định chiều cao của ô đó.
Colspan="n"		Tạo một ô có độ rộng bằng n số cột khai báo.
Rowspan="n"		Tạo một ô có chiều cao bằng n số dòng khai báo.
Nowrap		Buộc trình duyệt đặt toàn bộ văn bản trong ô trên một dòng đơn lẻ.
>		

#### 2.9.5. Thẻ TH và thuộc tính

THẺ MỞ	THẺ ĐÓNG	Ý NGHĨA
<TH	</TH>	
Valign="Valignment"		Căn lề theo chiều dọc nội dung một ô tiêu đề theo đường viền trên dưới: các giá trị có thể nhận top, bottom, middle.
Background="url"		Chỉ định ảnh nền cho ô tiêu đề đó.
Bgcolor="Color"		Chỉ định màu nền cho ô tiêu đề đó.
Width="n"		Xác định chiều rộng của ô tiêu đề đó.
Height="m"		Xác định chiều cao của ô tiêu đề đó.
Colspan="n"		Tạo một ô tiêu đề có độ rộng bằng n số cột khai báo.
Rowspan="n"		Tạo một ô tiêu đề có chiều cao bằng n số dòng khai báo.
Nowrap		Buộc trình duyệt đặt toàn bộ văn bản trong ô trên một dòng đơn lẻ.
>		

Ví dụ 4.2: Để tạo một bảng gồm 2 dòng 2 cột ta sử dụng cấu trúc như sau:

```
<HTML>
```

```
  <Head><Title>Ví dụ chương 4 phần 1</Title></Head>
```

```
  <Body>
```

```

<Table border=2>
  <tr>
    <td>Ô thứ nhất</td>
    <td>Ô thứ hai</td>
  </tr>
  <tr>
    <td>Ô thứ ba</td>
    <td>Ô thứ tư</td>
  </tr>
</Table>
</Body>
</HTML>

```

## 2.10. Phân khung cho trang web

**Khung trong dòng - PHẦN TỬ (IFRAME):** Giúp người thiết kế có thể tạo, chèn một khung vào khối văn bản

**Cú pháp:**

```

<IFRAME SRC="url" HEIGHT="n" WIDTH="m"
  SCROLLING="Auto/Yes/No"><IFRAME>

```

**Ý nghĩa các thuộc tính:**

Thuộc tính	Tác dụng
SRC	Xác định đường dẫn tới trang tài liệu ban đầu được chọn làm nội dung của khung.
SCROLLING= $\begin{cases} \text{"Auto"} \\ \text{"Yes"} \\ \text{"No"} \end{cases}$	Xác định kiểu hiển thị thanh cuộn. +Auto: Thanh cuộn xuất hiện khi cần thiết (khi kích cỡ của khung < kích cỡ của trang web được chọn làm nội dung của khung) +Yes: Thanh cuộn luôn xuất hiện. +No: Thanh cuộn không xuất hiện.
WIDTH="m"	Xác định chiều rộng của khung (WIDTH >= 1 pixel)
HEIGHT="n"	Xác định chiều cao của khung (HEIGHT >= 1 pixel)

*Ví dụ:*

```

<HTML><HEAD>
  <TITLE>Ví dụ khung trong dòng</TITLE>
</HEAD>
<BODY>

```

```

<FONT face="Arial"
color="Green"> Chèn
Frame vào trong trang
web mà không dùng phần
tử Frameset

<IFRAME
src="Vidu/a.html" height=400
width=450 scrolling="yes">

</IFRAME >

</BODY>
</HTML>

```



Hình 13. Sử dụng IFRAME

## 2.11. Mẫu biểu (FORM)

### 2.11.1. Giới thiệu mẫu biểu

Mẫu biểu được sử dụng để nhận các thông tin vào từ người dùng và cung cấp một tương tác nào đó thông qua các thành phần của mẫu biểu. Form thường được dùng cho các mục đích:

- **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 (e-mail), 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** (search box) 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.

Các thành phần của mẫu biểu: Mẫu biểu chứa các phần tử đặc biệt gọi là các điều khiển (control). Các điều khiển bao gồm: Hộp nhập liệu (Textbox), Vùng nhập liệu (Textarea), Nút lựa chọn (Radio), Nút kiểm tra (Checkbox), Hộp lựa chọn (Combobox), Nút bấm (Submit), Nút bấm (Reset), Nút bấm (Button), hộp tệp tin (File)

### 2.11.2. Các phần tử nhập 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 phần tử nhập HTML bao gồm: phần tử INPUT, phần tử TEXTAREA, phần tử LABEL, phần tử SELECT.

**Phần tử FORM:** được sử dụng để tạo một vùng trên trang web mà sẽ được xem như một biểu mẫu.

#### Cú pháp

```
<FORM
  ACCEPT="Internet media type"
  ACTION="url"
  METHOD="Get/Post">
```

### Ý nghĩa các thuộc tính

Thuộc tính	Tác dụng
ACCEPT	Xác định danh sách các kiểu MIME được máy chủ nhận ra, trong đó có chứa script để xử lý biểu mẫu.
ACTION	Xác định địa chỉ của script sẽ xử lý biểu mẫu hoàn chỉnh và được gửi đi.
METHOD	Xác định phương thức dữ liệu được gửi đến máy chủ. Nó cũng xác định giao thức được sử dụng khi khách hàng gửi dữ liệu đến máy chủ. -Nếu giá trị là GET : dữ liệu được gửi đi thể hiện để người dùng biết trên thanh địa chỉ -Nếu giá trị là POST : dữ liệu được gửi đi không thể hiện để người dùng biết

**Phần tử hộp nhập liệu (TEXTBOX):** Cho phép người dùng nhập dữ liệu trên một dòng đơn

### Cú pháp

```
<INPUT TYPE="Text" NAME="Tên" MAXLENGTH="n" SIZE="m" >
```

### Ý nghĩa các thuộc tính

Thuộc tính	Tác dụng
TYPE	Chỉ loại phần tử của form như textbox, checkbox, radio ....
NAME	Xác định tên của hộp textbox. Tên của hộp textbox đóng vai trò là biến chứa giá trị mà người dùng nhập vào hộp textbox.
MAXLENGTH	Xác định số ký tự tối đa mà người dùng có thể nhập vào hộp textbox.
SIZE	Chỉ độ rộng ban đầu của hộp textbox.

**Phần tử vùng nhập liệu <TEXTAREA>:** Cho phép người dùng nhập liệu trên một vùng văn bản

### Cú pháp

```
<TEXTAREA NAME="Tên" ROWS="n" COLS="m"></TEXTAREA>
```

### Ý nghĩa các thuộc tính

Thuộc tính	Tác dụng
NAME	Xác định tên của hộp văn bản textarea. Tên của hộp văn bản textarea đóng vai trò là biến chứa giá trị mà người dùng nhập vào vùng văn bản đó.
ROWS	Xác định số dòng của vùng văn bản.
COLS	Xác định số cột của vùng văn bản.

**Phần tử nút lựa chọn (RADIO):** Điều khiển này được sử dụng đối với các tập giá trị loại trừ lẫn nhau. Người dùng chỉ có thể chọn một lựa chọn vào một thời điểm nào đó.

### Cú pháp

```
<INPUT TYPE="Radio" NAME="Tên" VALUE="Giá trị" CHECKED >
```

### Ý nghĩa các thuộc tính

Thuộc tính	Tác dụng
TYPE	Chỉ loại phần tử của form.
NAME	Xác định tên của nút lựa chọn Radio. Tên của nút lựa chọn Radio đóng vai trò là biến chứa giá trị của nút mà người dùng lựa chọn.
VALUE	Xác định giá trị của nút sẽ được gửi tới server.
CHECKED	Xác lập trạng thái lựa chọn ngầm định cho nút..

### Chú ý

Để các nút lựa chọn Radio loại trừ được nhau, các điều khiển nút tùy chọn trong nhóm phải cùng tên.

**Phần tử nút kiểm tra (CHECKBOX):** Hộp kiểm tra thường được dùng khi thiết kế cho một yêu cầu với nhiều lựa chọn

### Cú pháp

```
<INPUT TYPE="Checkbox" NAME="Tên" VALUE="Giá trị" CHECKED >
```

### Ý nghĩa các thuộc tính

Thuộc tính	Tác dụng
TYPE	Chỉ loại phần tử của form.
NAME	Xác định tên của hộp kiểm tra. Tên của hộp kiểm tra đóng vai trò là biến chứa giá trị của hộp kiểm tra mà người dùng lựa chọn.
VALUE	Xác định giá trị của hộp kiểm tra sẽ được gửi tới server.
CHECKED	Xác lập trạng thái lựa chọn ngầm định cho hộp kiểm tra.



**Phần tử hộp lựa chọn (COMBOBOX hoặc LISTBOX):** Phần tử này được sử dụng để hiển thị một danh sách các tùy chọn cho người dùng.

### Cú pháp:

```
<SELECT name="Tên"Multiple >
    <OPTION value="gt1" selected>Nhấn hiển thị</OPTION>
    <OPTION value="gt2">Nhấn hiển thị</OPTION>
    .....
</SELECT>
```

Trong đó:

- Thẻ Select sử dụng để khai báo hộp lựa chọn
- Thẻ Option sử dụng để khai báo các thành phần trong danh sách thả xuống

### Ý nghĩa các thuộc tính

Thuộc tính	Tác dụng
Name	Gán tên cho phần tử, có vai trò là một biến chứa giá trị được chọn.
Multiple	Là thuộc tính logic cho phép người dùng chọn nhiều hơn một tùy chọn
value	Xác định giá trị cho các thành phần trong danh sách thả xuống. Mỗi khi biểu mẫu được gửi đi giá trị này được gán cho name của phần tử select.
selected	Đây là thuộc tính logic được sử dụng để chọn trước một tùy chọn.
size	Thuộc tính xác định đây là combo box hay list box

**Phần tử nút SUBMIT:** Khi người dùng nhấp vào nút submit, biểu mẫu được gửi đi đến địa chỉ được xác định trong thuộc tính action.

### Cú pháp

```
<INPUT TYPE="Submit" NAME="Tên" VALUE="hiệu ứng" CHECKED >
```

**Phần tử 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 thuộc tính của chúng.

### Cú pháp

```
<INPUT TYPE="Reset" NAME="Tên" VALUE="hiệu ứng" CHECKED >
```

**Phần tử nút bấm BUTTON:** Phần tử này tạo ra điều khiển **Button**. Khi người dùng nhấp vào nút button, biểu mẫu được nhận để xử lý. Cặp tên/giá trị của nút button được nhận cùng với biểu mẫu.

```
INPUT TYPE="Button" NAME="Tên" VALUE="hiệu ứng" CHECKED >
```

### 2.11.3. Ví dụ về biểu mẫu:

```
<!DOCTYPE html>
<html>
<body>
  <h2>Gửi thư tới địa chỉ
  phunt@fit-hau1.edu.vn </h2>
  <form
    action="mailto:phunt@fit-
    hau1.edu.vn " method="post"
    enctype="text/plain">
    Tên:<br>
    <input type="text"
    name="name"><br>
    Địa chỉ E-mail:<br>
    <input type="text"
    name="mail"><br>
    Đánh giá:<br>
    <input type="text" name="comment" size="50"><br><br>
    <input type="submit" value="Send">
    <input type="reset" value="Reset">
  </form>
</body>
</html>
```

### Gửi thư tới địa chỉ phunt@fit-hau1.edu.vn

Tên:

Địa chỉ E-mail:

Đánh giá:

Hình 14 Ví dụ về form

**TỔNG KẾT CHƯƠNG 2:** Chương này đã hướng dẫn bạn làm thế nào để sử dụng các thẻ HTML tạo ra trang web cho mình. HTML là một ngôn ngữ nền tảng của web với việc cho phép đưa vào các thành phần gồm văn bản, đồ họa, tạo liên kết, khung, bảng và đặc biệt là biểu mẫu (nơi giúp người sử dụng giao tiếp với mạng Internet).

### BÀI TẬP:

1. Hãy sử dụng các thẻ trình bày văn bản để tạo đơn xin việc (CV) cho cá nhân.
2. Hãy sử dụng các thẻ đã học để tạo ra một trang web có giao diện như hình dưới?



Hình 15. Mẫu giao diện

## CHƯƠNG 3: NGÔN NGỮ CSS

### 3.1. Giới thiệu

CSS là viết tắt của cụm từ "Cascading Style Sheet", nó là một ngôn ngữ quy định cách trình bày của các thẻ html trên trang web. Là ngôn ngữ đang được sử dụng rất nhiều trong lập trình web, có thể nói CSS ra đời đã tạo nên một cuộc cách mạng ở các website trên toàn thế giới.

Có thể thấy CSS có thể quản lý toàn bộ các thành phần (giao diện) của trang web bạn thông qua 1 file CSS, điều khiển chính xác hơn cách trình bày Layout. Quan trọng nhất là tính tái sử dụng để áp dụng cho nhiều thành phần giống nhau. Sử dụng được các kỹ thuật tiên tiến, phức tạp mà HTML không làm được.

Để học CSS cần có kiến thức về HTML, nó không thật sự cần thiết nếu bạn chỉ dùng CSS để trình bày cho một trang HTML có sẵn, nhưng bạn vẫn cần biết ý nghĩa một số thẻ HTML, nó sẽ có ích khi bạn viết CSS. Tuy nhiên, nếu bạn muốn tự thiết kế, trình bày một trang web của riêng mình thì tùy theo quy mô trang web, bạn cần phải học thêm cả HTML, XHTML, Javascript và một số ngôn ngữ lập trình web khác.

Để viết mã CSS có thể sử dụng một trình soạn thảo đơn giản như Notepad, Wordpad trong Windows hay Pico trong Linux, Simple Text trong Mac, ... tuy nhiên được sử dụng phổ biến nhất hiện nay vẫn là công cụ Macro Dreamweaver sẽ được trình bày tại phần phụ lục của cuốn sách. Ngoài ra để có thể hiển thị tốt nhất cách thể hiện của một trang HTML kết hợp với CSS nên sử dụng trình duyệt phiên bản mới nhất.

### 3.2. Một số quy ước về cách viết CSS

#### 3.2.1. Cú pháp CSS

Để định màu nền cho một trang web là xanh nhạt (light cyan) chúng ta dùng đoạn mã lệnh sau:

+ Trong HTML: `<body bgcolor="#00BFF3">`

+ Trong CSS: `body {background-color:#00BFF3; }`

Nhìn qua ví dụ trên ít nhiều chúng ta cũng thấy được mối tương đồng giữa các thuộc tính trong HTML và CSS cho nên nếu bạn đã học qua HTML thì cũng sẽ rất dễ dàng tiếp thu CSS.

**Cú pháp CSS cơ bản:**

**Tên\_thẻ{Thuộc\_tính:giá\_trị;}**

Hoặc

**.Tên\_class{Thuộc\_tính:giá\_trị;}**

Hoặc

**#Tên\_ID{Thuộc\_tính:giá\_trị;}**

*Trong đó:*

- + Tên thẻ: là tên một thẻ HTML mà người lập trình muốn CSS tương tác
- + Tên class: tên lớp bất kỳ do người lập trình định nghĩa được sử dụng nhiều lần trên trang HTML
- + Tên ID: Tên của một phần tử ID bất kỳ chỉ được sử dụng 1 lần duy nhất trên trang HTML

Ví dụ trong HTML ta có đoạn mã như:

```
<input name="Search" type="Text" value="Key Word">
```

Để áp dụng thuộc tính CSS cho riêng ô tìm kiếm này chúng ta sẽ dùng thẻ `input[name="Search"]`.

Ngoài việc viết tên thẻ cụ thể, cũng có thể dùng một thẻ đại diện như `*{ color:red }` sẽ tác động đến tất cả các thành phần có trên trang web làm cho chúng có text màu đỏ.

+ Thuộc tính: Chính là các thuộc tính quy định cách trình bày. Ví dụ: `background-color`, `font-family`, `color`, `padding`, `margin`,...

Mỗi thuộc tính CSS phải được gán một giá trị. Nếu có nhiều hơn một thuộc tính cho một thẻ thì chúng ta phải dùng một dấu `;` (chấm phẩy) để phân cách các thuộc tính. Tất cả các thuộc tính trong một selector sẽ được đặt trong một cặp ngoặc nhọn sau thẻ.

*Ví dụ:*

```
body { background:#FFF; color:#FF0000; font-size:14pt }
```

Để dễ đọc hơn, bạn nên viết mỗi thuộc tính CSS ở một dòng. Tuy nhiên, nó sẽ làm tăng dung lượng lưu trữ CSS của bạn.

*Ví dụ:*

```
body {  
    background:#FFF;  
    bolor:#FF0000;  
    font-size:14pt  
}
```

Đối với một trang web có nhiều thành phần có cùng một số thuộc tính, chúng ta có thể thực hiện gom gọn lại như sau:

```
h1 { color:#0000FF;  
    text-transform:uppercase }  
h2 {  
    color:#0000FF;  
    text-transform:uppercase;  
}  
h3 {  
    color:#0000FF;  
    text-transform:uppercase;  
}  
=> h1, h2, h3 {  
    color:#0000FF;  
    text-transform:uppercase; }
```

Đối với một giá trị có khoảng trắng, bạn nên đặt tất cả trong một dấu ngoặc kép.

*Ví dụ:* `font-family:"Times New Roman"`

Đối với các giá trị là đơn vị đo, không nên đặt một khoảng cách giữa số đo với đơn vị của nó. Nó sẽ làm CSS của bạn bị vô hiệu trên Mozilla Firefox hay Netscape.

*Chú thích trong CSS:* Cũng như nhiều ngôn ngữ web khác. Trong CSS, chúng ta cũng có thể viết chú thích cho các đoạn code để dễ dàng tìm, sửa chữa trong những lần cập nhật sau. Chú thích trong CSS được viết như sau `/* Nội dung chú thích */`

*Ví dụ:*

```
/* Màu chữ cho trang web */
body {
    color:red
}
```

Nhóm phần tử với class cho phép định dạng style của các đối tượng (table, td, div, span...) Có thể sử dụng lặp đi lặp lại nhiều lần trong cùng một file HTML. Để thiết lập phần tử là class ta sử dụng dấu `.` để thực hiện khai báo

Ví dụ chúng ta có một đoạn mã HTML sau đây :

```
<p>Danh Sách Các Tỉnh, Thành Phố Của Việt Nam</p>
<ul>
    <li>Hà Nội</li>
    <li>TP. Hồ Chí Minh</li>
    <li>Đà Nẵng</li>
    <li>Thừa Thiên Huế</li>
    <li>Khánh Hòa</li>
    <li>Quảng Ninh</li>
    <li>Tiền Giang</li>
</ul>
```

Yêu cầu đặt ra là làm thế nào để tên các thành phố là màu đỏ và tên các tỉnh là màu xanh da trời. Để giải quyết vấn đề này chúng ta sẽ dùng một thuộc tính HTML gọi là class để tạo thành 2 nhóm là thành phố và tỉnh. Ta sẽ viết lại đoạn HTML sau thành như thế này:

```
<p>Danh Sách Các Tỉnh, Thành Phố Của Việt Nam</p>
<ul>
    <li class="tp">Hà Nội</li>
    <li class="tp">TP. Hồ Chí Minh</li>
    <li class="tp">Đà Nẵng</li>
    <li class="tinh">Thừa Thiên Huế</li>
    <li class="tinh">Khánh Hòa</li>
    <li class="tinh">Quảng Ninh</li>
    <li class="tinh">Tiền Giang</li>
</ul>
```

Với việc dùng class để nhóm các đối tượng như trên thì công việc của chúng ta sẽ trở nên đơn giản hơn nhiều:

```
li .tp {color:FF0000}
li .tinh {color:0000FF}
```

Cũng giống như **class** nhưng **ID** chỉ được sử dụng một lần nếu sử dụng nhiều lần một id thì vẫn được tuy nhiên nó không đúng chuẩn của w3c và sẽ gặp rắc rối khi dùng

ID trong javascript. Việc sử dụng phần tử ID ta sử dụng dấu # để thực hiện khai báo cho IDs

Ví dụ: Cũng với đoạn HTML như ví dụ về class. Nhưng yêu cầu đặt ra là Hà Nội sẽ có màu đỏ sậm, TP. Hồ Chí Minh màu đỏ, Đà Nẵng màu đỏ tươi còn các tỉnh màu xanh da trời. Để giải quyết vấn đề này chúng ta sẽ sử dụng thuộc tính HTML là id để nhận dạng mỗi thành phố và dùng class để nhóm các tỉnh. Đoạn HTML của chúng ta bây giờ sẽ là :

```
<p>Danh Sách Các Tỉnh, Thành Phố Của Việt Nam</p>
<ul>
  <li id="hanoi">Hà Nội</li>
  <li id="hcmc">TP. Hồ Chí Minh</li>
  <li id="danang">Đà Nẵng</li>
  <li class="tinh">Thừa Thiên Huế</li>
  <li class="tinh">Khánh Hòa</li>
  <li class="tinh">Quảng Ninh</li>
  <li class="tinh">Tiền Giang</li>
</ul>
```

Và đoạn CSS cần dùng sẽ là :

```
#hanoi { color:# 790000 }
#hcmc { color:#FF0000 }
#danang { color:#FF00FF }
.tinh { color:#0000FF }
```

**Lưu ý:** Không nên đặt tên id và class với ký tự đầu là chữ số, nó sẽ không làm việc cho Firefox.

### 3.2.2. Đơn vị CSS

Trong CSS hỗ trợ các loại đơn vị là đơn vị đo chiều dài và đơn vị đo góc, thời gian, cường độ âm thanh và màu sắc. Tuy nhiên, sử dụng phổ biến nhất vẫn là đơn vị đo chiều dài và màu sắc. Sau đây là bảng liệt kê các đơn vị chiều dài dùng trong CSS.

Đơn vị chiều dài

Đơn vị	Mô tả	Đơn vị	Mô tả
%	Phần trăm	Pt	Point (1pt = 1/72 inch)
in	Inch (1 inch=2,54cm)	Pc	Pica (1 pc = 12 pt)
cm	Centimeter	px	Pixels (điểm ảnh trên màn hình máy tính)
mm	Milimeter	ex	1 ex bằng chiều cao của chữ x in thường của font hiện hành. Do đó, đơn vị này không những phụ thuộc trên kích cỡ font chữ mà còn phụ thuộc vào loại font chữ vì cùng 1 cỡ 14px nhưng chiều cao của font Time New Roman và font Tahoma khác nhau
em	1 em tương đương kích thước font chữ hiện hành, nếu font chữ hiện hành là 14px thì 1em = 14px. Đây là đơn vị đo rất hữu ích cho trang web		

### Đơn vị màu sắc

Đơn vị	Mô tả
Color-name	Tên màu bằng tiếng Anh. Ví dụ: green, blue, red, black ....
RGB (r,g,b)	Màu RGB tương ứng với 03 giá trị R, G, B (red, green, blue) có giá trị từ 0 đến 255 kết hợp với nhau tạo ra màu.
RGB (%r,%g,%b)	Màu RGB với 03 giá trị R,G,B có giá trị từ 0 đến 100% kết hợp với nhau để tạo ra màu
Hexadecimal RGB	Mã màu RGB với cách thể hiện bằng chữ số thập lục phân (hexa) với 2 ký tự đầu dành cho màu đỏ (Red), 02 ký tự kế tiếp dành cho màu xanh lá cây (Green) và 02 ký tự cuối dành cho màu xanh da trời (Blue). Ví dụ: #FFFFFF là màu trắng, #000000 là màu đen, #FF0000 là màu đỏ

#### 3.2.3. Vị trí đặt CSS

**Cách 1:** Nội tuyến (kiểu thuộc tính) đây là 1 phương pháp nguyên thủy nhất để nhúng CSS vào 1 trang html. Bằng cách nhúng vào từng thẻ HTML muốn áp dụng và dĩ nhiên trong trường hợp này chúng ta sẽ không cần thẻ trong cú pháp.

Lưu ý: Nếu bạn muốn áp dụng nhiều thuộc tính cho nhiều thẻ HTML khác nhau thì không nên dùng cách này.

Ở ví dụ sau chúng ta sẽ tiến hành định nền màu trắng cho trang và màu chữ xanh lá cho đoạn văn bản như sau:

VD:

```
<html>
  <head><title>Vi du</title></head>
  <body style="background-color=#FFF;">
    <p style="color:green"> Học lập trình thật là
    thú vị </p>
  </body>
</html>
```

**Cách 2:** Bên trong (thẻ style)

Thật ra nếu nhìn kỹ chúng ta cũng nhận ra đây chỉ là một phương cách thay thế cách thứ nhất bằng cách rút tất cả các thuộc tính CSS vào trong thẻ style (để tiện cho công tác bảo trì, sửa chữa ấy mà).

Cũng ví dụ làm trang web có màu nền trắng, đoạn văn bản chữ xanh lá, chúng ta sẽ thể hiện như sau:

```
<html>
  <head><title>vi du </title></head>
  <style type="text/css">
```

```

        body {background-color:#FFF}\
        p {color:#00FF00}
    </style>
</body>
    <p> Hoc lap trinh that la thu vi </p>
</body>
</html>

```

**Lưu ý:** Thẻ style nên đặt trong thẻ head.

Đối với những trình duyệt cũ, không thể nhận ra thẻ <style>. Theo mặc định, thì khi một trình duyệt không nhận ra một thẻ thì nó sẽ hiện ra phần nội dung chứa trong thẻ. Như ở ví dụ trên, nếu trình duyệt không hỗ trợ thẻ style thì 2 dòng CSS: body {background-color:#FFF } p { color:#00FF00 } sẽ hiện ra trên trình duyệt.

Để tránh tình trạng này, bạn nên đưa vào thêm dấu <!-- ở trước và --> ở sau khối code CSS. Như ví dụ trên sẽ viết lại là:

```

<style type="text/css">
    <!--  body { background-color:#FFF }
        p { color:#00FF00 }  -->
</style>

```

**Cách 3 :** Bên ngoài (liên kết với một file CSS bên ngoài)

Tương tự như cách 2, thay vì đặt tất cả mã CSS trong thẻ style chúng ta sẽ đưa chúng vào trong 1 file CSS (có đuôi là .css) bên ngoài là liên kết nó vào trong web bằng thuộc tính href trong thẻ link. Đây là cách làm tối ưu nhất, được khuyến cáo, nó đặc biệt hữu ích cho việc đồng bộ hay bảo trì một website lớn sử dụng cùng một kiểu.

Đầu tiên chúng ta sẽ tạo ra một file vidu.html có nội dung như sau:

```

<html>
    <head><title>Vi du</title>
        <link      rel="stylesheet"      type="text/css"
        href="style.css">
    </head>
</body>
    <p>Hoc lap trinh that thu vi</p>
</body>
</html>

```

Sau đó ta tạo 1 file style.css với nội dung:

```

body{background-color: #FFF }
p{color: #00FF00}

```

### 3.2.4. Sự ưu tiên

Trước khi thực thi CSS cho 1 trang web, trình duyệt sẽ đọc toàn bộ CSS mà trang web có thể được áp dụng, bao gồm CSS mặc định của trình duyệt, file CSS bên ngoài, CSS nhúng vào thẻ <style> và các CSS nội tuyến. sau đó trình duyệt sẽ tổng hợp lại vào 1 CSS ảo, và nếu thuộc tính CSS giống nhau thì thuộc tính nào nằm sau sẽ được



ưu tiên sử dụng vậy thứ tự sẽ là : CSS nội tuyến (cách 1), CSS bên trong (cách 2), CSS bên ngoài (cách 3), CSS mặc định của trình duyệt

Ví dụ: Trong một trang web có liên kết tới **file style.css** có nội dung như sau:

```
p {  
    color:#333;  
    text-align:left;  
    width:500px  
}
```

Trong thẻ **<style>** giữa thẻ **<head>** cũng có một đoạn CSS liên quan:

```
p {  
    background-color:#FF00FF;  
    text-align:right;  
    width:100%;  
    height:150px  
}
```

Trong phần nội dung trang web đó cũng có sử dụng **CSS nội tuyến**:  
`<p style="height:200px; text-align:center; border:1px solid #FF0000; color:#000" >`

Khi thực thi CSS trình duyệt sẽ đọc hết tất cả các nguồn chứa style rồi sẽ tổng hợp lại vào một CSS ảo và nếu có sự trùng lặp các thuộc tính CSS thì nó sẽ lấy thuộc tính CSS có mức ưu tiên cao hơn. Như ví dụ trên chúng ta sẽ thấy CSS cuối cùng mà phần tử p nhận được là:

```
p {  
    background-color:#FF00FF;  
    width:100%;  
    height:200px;  
    text-align:center;  
    border:1px solid #FF0000;  
    color:#000  
}
```

Vậy có cách nào để thay đổi độ ưu tiên cho một thuộc tính nào đó? Thật ra thì trong CSS đã có sẵn một thuộc tính giúp chúng ta thực hiện điều này, đó chính là thuộc tính **!important**. Chỉ cần bạn đặt thuộc tính này sau một thuộc tính nào đó theo cú pháp selector {property:value !important} thì trình duyệt sẽ hiểu đây là một thuộc tính được ưu tiên. Bây giờ, chúng ta cùng xét lại ví dụ trên nhưng có đặt thêm một số thuộc tính **!important** vào xem kết quả như thế nào nhé.

```
p {  
    width:500px;  
    text-align:left !important;  
    color:#333 !important  
}
```

```

p {
    background-color:#FF00FF;
    width:100%;
    height:150px !important;
    text-align:right;
}
<p style="text-align:center; height:200px; border:1px
solid #FF0000; color:#000" >

```

Phần CSS sẽ tác động lên thuộc tính p là:

```

p {
    background-color:#FF0000;
    width:100%;
    height:150px !important;
    text-align:left !important;
    border:1px solid #FF0000;
    color:#333 !important }

```

**Lưu ý:** Cùng một thuộc tính cho một selector thì nếu cả hai thuộc tính đều đặt !important thì cái sau được lấy.

### 3.3. Màu chữ và màu nền

**Thuộc tính background-color:** giúp định màu nền cho một thành phần trên trang web. Các giá trị mã màu của background-color cũng giống như color nhưng có thêm giá trị transparent để tạo nền trong suốt.

Ví dụ sau đây sẽ chỉ cho chúng ta biết cách sử dụng thuộc tính background-color để định màu nền cho cả trang web, các thành phần h1, h2 lần lượt là xanh lơ, đỏ và cam.

```

body {background-color:cyan }
h1 {background-color:red}
h2 {background-color:orange}

```

#### Thuộc tính background-image

Việc sử dụng ảnh nền giúp trang web trông sinh động và bắt mắt hơn. Để chèn ảnh nền vào một thành phần trên trang web chúng ta sử dụng thuộc tính background-image.

Bây giờ chúng ta sẽ cùng làm một ví dụ minh họa để xem thuộc tính background-image sẽ hoạt động ra sao. Đầu tiên hãy tìm một tấm ảnh mà bạn thích, ở đây tôi sẽ lấy tấm ảnh logo của Khoa Công nghệ thông tin trường Đại học Công nghiệp Hà nội

Sau đó, chúng ta sẽ viết CSS để đặt logo này làm ảnh nền trang web như sau:

```

h1 {background-color:red}
h2 {background-color:orange}
p { background-color: FDC689}

```

Như các bạn đã thấy chúng ta sẽ phải chỉ định đường dẫn của ảnh trong cặp ngoặc đơn sau url. Do ảnh đặt trong cùng thư mục với file style3.css nên chúng ta chỉ cần ghi cntt-hau1.jpg. Nhưng nếu chúng ta tạo thêm một thư mục img trong thư mục thì chúng ta sẽ phải ghi là

```
background-image:url(img/cntt-hau1.jpg)
```

Đôi khi nếu không chắc lắm bạn có thể dùng đường dẫn tuyệt đối cho ảnh

### **Thuộc tính background-repeat**

Nếu sử dụng một ảnh có kích thước quá nhỏ để làm nền cho một đối tượng lớn hơn thì theo mặc định trình duyệt sẽ lặp lại ảnh nền để phủ kín không gian còn thừa. Thuộc tính background-repeat cung cấp cho chúng ta các điều khiển giúp kiểm soát trình trạng lặp lại của ảnh nền. Thuộc tính này có 4 giá trị:

- + repeat-x: Chỉ lặp lại ảnh theo phương ngang.
- + repeat-y: Chỉ lặp lại ảnh theo phương dọc.
- + repeat: Lặp lại ảnh theo cả 2 phương, đây là giá trị mặc định.
- + no-repeat: Không lặp lại ảnh.

Bây giờ, chúng ta hãy thêm thuộc tính background-repeat này vào ví dụ trên thử xem sao.

```
body {  
    background-image:url(cntt-hau1.png);  
    background-repeat:no-repeat; }
```

Các bạn xem, có phải ảnh nền đã không bị lặp lại như trong ví dụ trước, hãy thử thay đổi qua lại giữa các giá trị và xem kết quả tạo ra.

### **Thuộc tính background-attachment**

Background-attachment là một thuộc tính cho phép bạn xác định tính cố định của ảnh nền so với nội dung trang web. Thuộc tính này có 2 giá trị:

- scroll: Ảnh nền sẽ cuộn cùng nội dung trang web, đây là giá trị mặc định.
- fixed: Cố định ảnh nền so với nội dung trang web. Khi áp dụng giá trị này, ảnh nền sẽ đứng yên khi bạn đang cuộn trang web.

### **Thuộc tính background-position**

Theo mặc định ảnh nền khi được chèn sẽ nằm ở góc trên, bên trái màn hình. Tuy nhiên với thuộc tính background-position bạn sẽ có thể đặt ảnh nền ở bất cứ vị trí nào (trong không gian của thành phần mà nó làm nền).

Background-position sẽ dùng một cặp 2 giá trị để biểu diễn tọa độ đặt ảnh nền. Có khá nhiều kiểu giá trị cho thuộc tính position. Như đơn vị chính xác như centimeters, pixels, inches,... hay các đơn vị qui đổi như %, hoặc các vị trí đặt biệt như top, bottom, left, right.

Giá trị một số thuộc tính của Background

GIÁ TRỊ	Ý NGHĨA
Background-position:5cm 2cm	Ảnh được định vị 5cm từ trái qua và 2cm từ trên xuống.

Background-position:20% 30%	Ảnh được định vị 20% từ trái qua và 30% từ trên xuống.
Background-position:bottom left	Ảnh được định vị ở góc trái phía dưới

**Thuộc tính background rút gọn:** Khi sử dụng quá nhiều thuộc tính CSS sẽ gây khó khăn cho người đọc, công tác chỉnh sửa cũng như tốn nhiều dung lượng ổ cứng cho nên CSS đưa ra một cấu trúc rút gọn cho các thuộc tính cùng nhóm.

Chúng ta có thể nhóm lại đoạn CSS sau:

```
background-color:transparent;
background-image: url(cntt-hau1.png);
background-repeat: no-repeat;
background-attachment: fixed;
background-position: right bottom;
```

Thành một dòng ngắn gọn:

```
background:transparent url(cntt-hau1.png) no-repeat fixed right bottom;
```

Từ ví dụ trên chúng ta có thể khái quát cấu trúc rút gọn cho nhóm background:

```
background:<background-color> | <background-image> | <background-repeat> | <background-attachment> | <background-position>
```

Theo mặc định thì các thuộc tính không được đề cập sẽ nhận các giá trị mặc định. Chúng ta sẽ bỏ qua hai thuộc tính background-attachment và background-position ở dòng mã trên đi:

```
background:transparent url(cntt-hau1.png) no-repeat;
```

Hai thuộc tính không được chỉ định sẽ đơn thuần được thiết lập tới giá trị mặc định mà chúng ta điều biết là scroll và top left.

### 3.4. Font chữ

**Thuộc tính font-family:** có công dụng định nghĩa một danh sách ưu tiên các font sẽ được dùng để hiển thị một thành phần trang web. Theo đó, thì font đầu tiên được liệt kê trong danh sách sẽ được dùng để hiển thị trang web. Nếu như trên máy tính truy cập chưa cài đặt font này thì font thứ hai trong danh sách sẽ được ưu tiên...cho đến khi có một font phù hợp.

Có hai loại tên font được dùng để chỉ định trong font-family: family-names và generic families.

+ Family-names: Tên cụ thể của một font. Ví dụ: Arial, Verdana, Tohama,...

+ Generic families: Tên của một họ gồm nhiều font. Ví dụ: sans-serif, serif,...

Khi lên danh sách font dùng để hiển thị một trang web ta sẽ chọn những font mong muốn trang web sẽ được hiển thị để đặt ở các vị trí ưu tiên. Tuy nhiên, có thể những font này sẽ không thông dụng lắm nên cũng cần chỉ định thêm một số font thông dụng như Arial, Tohama hay Times New Roman. Thực hiện theo cách này thì sẽ đảm bảo trang web của bạn có thể hiển thị tốt trên bất kỳ hệ thống nào.

Ví dụ sau sẽ viết CSS để quy định font chữ dùng cho cả trang web là Times New Roman, Tohama, sans-serif, và font chữ dùng để hiển thị các tiêu đề h1, h2, h3 sẽ là Arial, Verdana và các font họ serif.

```
body { font-family:"Times New Roman",Tohama,sans-serif }
h1, h2, h3 { font-family:arial,verdana,serif }
```

Mở trang web trong trình duyệt và kiểm tra kết quả. Chúng ta thấy phần tiêu đề sẽ được ưu tiên hiển thị bằng font Arial, nếu trên máy không có font này thì font Verdana sẽ được ưu tiên và kế đó sẽ là các font thuộc họ serif.

**Chú ý:** Đối với các font có khoảng trắng trong tên như Times New Roman cần được đặt trong dấu ngoặc kép.

**Thuộc tính font-style:** định nghĩa việc áp dụng các kiểu in thường (normal), in nghiêng (italic) hay xiên (oblique) lên các thành phần trang web. Trong ví dụ bên dưới chúng ta sẽ thử thực hiện áp dụng kiểu in nghiêng cho thành phần h1 và kiểu xiên cho h2.

Ví dụ:

```
h1 {font-style:italic; }
h2 {font-style:oblique; }
```

**Thuộc tính font-variant:** được dùng để chọn giữa chế độ bình thường và small-caps của một font chữ. Một font small-caps là một font sử dụng chữ in hoa có kích cỡ nhỏ hơn in hoa chuẩn để thay thế những chữ in thường. Nếu như font chữ dùng để hiển thị không có sẵn font small-caps thì trình duyệt sẽ hiện chữ in hoa để thay thế.

Trong ví dụ sau chúng ta sẽ sử dụng kiểu small-caps cho phần h1

```
h1 {font-variant:small-caps}
```

**Thuộc tính font-weight:** mô tả cách thức thể hiện của font chữ là ở dạng bình thường (normal) hay in đậm (bold). Ngoài ra, một số trình duyệt cũng hỗ trợ mô tả độ in đậm bằng các con số từ 100 – 900.

*Ví dụ:* p{font-weight:bold}

**Thuộc tính font-size:** Kích thước của một font được định bởi thuộc tính font-size. Thuộc tính này nhận các giá trị đơn vị đo hỗ trợ bởi CSS bên cạnh các giá trị xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger. Tùy theo mục đích sử dụng của website có thể lựa chọn những đơn vị phù hợp. Ví dụ trang web phục vụ chủ yếu là những người già, thị lực kém hay những người dùng sử dụng các màn hình máy tính kém chất lượng có thể cân nhắc sử dụng các đơn vị qui đổi như em hay %. Như vậy sẽ đảm bảo font chữ trên trang web luôn ở kích thước phù hợp.

Ở ví dụ sau trang web sẽ có kích cỡ font là 20px, h1 là 3em = 3 x 20 = 60px, h2 là 2em = 40px.

```
body {font-size:20px}
h1{font-size:3em }
h2 { font-size:2em }
```

Thuộc tính font rút gọn Tương tự như các thuộc tính background, chúng ta cũng có thể rút gọn các thuộc tính font lại thành một thuộc tính đơn như ví dụ sau:

```
h1 {
```

```

font-style: italic;
font-variant: small-caps;
font-weight: bold;
font-size: 35px;
font-family: arial, verdana, sans-serif;
}
Thành
h1 {
    font: italic bold 35px arial, verdana, sans-serif;
}

```

Cấu trúc rút gọn cho các thuộc tính nhóm font:

Font :<font-style> | <font-variant> | <font-weight> | <font-size> |<font-family>

### 3.5. Text

**Thuộc tính color:** Để định màu chữ cho một thành phần nào đó trên trang web chúng ta sử dụng thuộc tính color. Giá trị của thuộc tính này là các giá trị màu CSS hỗ trợ. Ví dụ sau sẽ viết CSS để định màu chữ chung cho một trang web là đen, cho tiêu đề h1 màu xanh da trời, cho tiêu đề h2 màu xanh lá chúng ta sẽ làm như sau:

```

body {color:#000}
h1 {color:#0000FF }
h2 {color:#00FF00 }

```

**Thuộc tính text-indent:** cung cấp khả năng tạo ra khoảng thụt đầu dòng cho dòng đầu tiên trong đoạn văn bản. Giá trị thuộc tính này là các đơn vị đo cơ bản dùng trong CSS.

Trong ví dụ sau sẽ định dạng thụt đầu dòng một khoảng 30px cho dòng văn bản đầu tiên trong mỗi đoạn văn bản đối với các thành phần <p>. Ví dụ: p {text-indent:30px}

**Thuộc tính text-align:** thêm các canh chỉnh văn bản cho các thành phần trong trang web. Cũng tương tự như các lựa chọn canh chỉnh văn bản trong các trình soạn thảo văn bản thông dụng như MS Word, thuộc tính này có tất cả 4 giá trị : left (canh trái – mặc định), right (canh phải), center (canh giữa) và justify (canh đều).

Trong ví dụ sau sẽ thực hiện canh phải các thành phần h1, h2 và canh đều đối với thành phần <p>

```

h1, h2 {text-align:right}
p {text-align:justify}

```

**Thuộc tính letter-spacing:** được dùng để định khoảng cách giữa các ký tự trong một đoạn văn bản. Muốn định khoảng cách giữa các ký tự trong thành phần h1, h2 là 7px và thành phần <p> là 5px chúng ta sẽ viết CSS sau:

```

h1, h2 {letter-spacing:7px}
p { letter-spacing:5px }

```

**Thuộc tính text-decoration:** thêm các hiệu ứng gạch chân (underline), gạch xiên (line-through), gạch đầu (overline), và một hiệu ứng đặc biệt là văn bản nhấp

nháy (blink). Ví dụ sau sẽ định dạng gạch chân cho thành phần h1, gạch đầu thành phần h2

```
h1 {text-decoration:underline}
```

```
h2 {text-decoration:overline}
```

**Thuộc tính text-transform:** là thuộc tính qui định chế độ in hoa hay in thường của văn bản mà không phụ thuộc vào văn bản gốc trên HTML. Thuộc tính này có tất cả 4 giá trị: uppercase (in hoa), lowercase (in thường), capitalize (in hoa ở ký tự đầu tiên trong mỗi từ) và none (không áp dụng hiệu ứng – mặc định).

Trong ví dụ dưới đây sẽ định dạng cho thành phần h1 là in hoa, h2 là in hoa đầu mỗi ký tự.

```
h1 {text-transform:uppercase}
```

```
h2 {text-transform:capitalize}
```

### 3.6. Pseudo-classes for Links

Một thành phần rất quan trọng trong mọi website chính là liên kết. Cũng như một đối tượng văn bản thông thường, hoàn toàn có thể áp dụng các thuộc tính định dạng đã học ở 2 bài trước như định font chữ, gạch chân, màu chữ,... cho một liên kết. Hơn nữa, CSS còn cung cấp một điều khiển đặc biệt được gọi là pseudo-classes. Pseudo-classes cho phép bạn xác định các hiệu ứng định dạng cho một đối tượng liên kết ở một trạng thái xác định như khi liên kết chưa được thăm (a:link), khi rê chuột lên liên kết (a:hover), khi liên kết được thăm (a:visited) hay khi liên kết đang được kích hoạt – đang giữ nhấn chuột (a:active). Với điều khiển pseudo-classes cùng với các thuộc tính CSS đã học chắc chắn sẽ mang lại rất nhiều ý tưởng về trang trí liên kết cho trang web. Sau đây, chúng ta sẽ tiến hành một số ví dụ để tìm hiểu thêm về các khả năng trang trí cho một liên kết dựa trên pseudo-classes.

*Ví dụ 1:* Ví dụ này sẽ áp dụng 4 màu sắc khác nhau cho từng trạng thái liên kết: các liên kết chưa thăm có màu xanh lá; các liên kết mouse over sẽ có màu đỏ tươi; các liên kết đã thăm sẽ có màu đỏ và các liên kết đang kích hoạt có màu tím.

```
a:link {color:#00FF00}
a:hover {color:#FF00FF}
a:visited {color:#FF0000}
a:active {color:# 662D91}
```

*Ví dụ 2:* Tạo các hiệu ứng tương ứng với trình trạng liên kết: các liên chưa thăm có màu xanh lá, kích cỡ font 14px; liên kết mouse over có màu đỏ tươi, kích cỡ font 1.2em, hiệu ứng nhấp nháy; liên kết đã thăm sẽ có màu xanh da trời, không có đường gạch chân; các liên kết đang kích hoạt có màu tím và font dạng small-caps.

```
a:link {
    color:#00FF00;
    font-size:14px
}
a:hover {
    color:#FF00FF;
    font-size:1.2em;
```

```

        text-decoration:blink
    }
    a:visited {
        color:#FF0000;
        text-decoration:none
    }
    a:active {
        color:# 662D91;
        font-variant:small-caps
    }

```

*Ví dụ 3:* Ví dụ này cũng tạo cho liên kết hiệu ứng màu sắc giống ví dụ 2 nhưng sẽ có thêm 1 số hiệu ứng: các liên kết sẽ có khung viền màu đen, kích cỡ font 14px; liên kết mouse over có nền light cyan; các liên kết đã thăm có nền light yellow.

```

a {
    border:1px solid #000;
    font-size:14px
}
a:link {
    color:#00FF00;
}
a:hover {
    background-color:#00BFF3;
    color:#FF00FF;
    font-size:1.2em;
    text-decoration:blink
}
a:visited {
    background-color:#FFF568;
    color:#FF0000;
    text-decoration:none
}
a:active { color:#662D91; font-variant:small-caps }

```

### 3.7. Span & div

**Nhóm phần tử với <span>:** Như đã trình bày ở chương trước thẻ <span> trong HTML thật ra là một thẻ trung hòa, nó không thêm hay bớt bất cứ một thứ gì vào một tài liệu HTML cả. Nhưng chính nhờ tính chất trung hòa này mà nó lại là một công cụ đánh dấu tuyệt vời để qua đó có thể viết CSS định dạng cho các phần tử mong muốn.

Ví dụ: Chúng ta có đoạn HTML sau trích dẫn câu nói của chủ tịch Hồ Chí Minh

```
<p>Không có gì quý hơn độc lập, tự do.</p>
```

Yêu cầu ở đây là hãy dùng CSS tô đậm 2 từ độc lập, tự do. Để giải quyết vấn đề này, chúng ta sẽ thêm thẻ <span> vào đoạn HTML như sau:



```
<p>Không có gì quý hơn <span
class="nhanmanh">độc lập</span>, <span
class="nhanmanh">tự do</span>.
```

Và bây giờ chúng ta có thể viết CSS cho yêu cầu trên:

```
.nhanmanh {font-weight:bold}
```

**Nhóm phần tử với <div>:** Điểm khác biệt giữa <div> là <span> dùng để nhóm một khối phần tử trong khi đó <div> có thể nhóm một hoặc nhiều khối phần tử. Trở lại ví dụ về danh sách tỉnh, thành trong phần class bài trước chúng ta sẽ giải quyết vấn đề bằng cách nhóm các phần tử với <div> như sau:

```
<p>Danh Sách Các Tỉnh, Thành Phố Việt Nam:</p>
<ul>
<div id="tp">
<li>Hà Nội</li>
<li>TP. Hồ Chí Minh</li>
<li>Đà Nẵng</li>
</div>
<div id="tinh">
<li>Thừa Thiên Huế</li>
<li>Khánh Hòa</li>
<li>Quảng Ninh</li>
<li>Tiền Giang</li>
</div>
</ul>
```

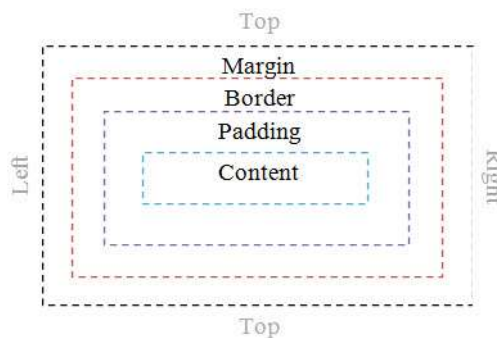
Và đoạn CSS cho mục đích này sẽ là:

```
#tp {color:#FF0000}
#tinh {color:0000FF}
```

### 3.8. Box Model

Trong CSS, box model (mô hình hộp) mô tả cách mà CSS định dạng khối không gian bao quanh một thành phần. Nó bao gồm padding (vùng đệm), border (viền) và margin (canh lề) và các tùy chọn. Hình bên dưới mô tả cấu trúc minh họa mô hình hộp cho một thành phần web.

Mô hình hộp trên chỉ là một mô hình lý thuyết lý tưởng. Bên dưới đây, chúng ta sẽ xét mô hình hộp của một đối tượng web cụ thể:



Hình 16. Model Box

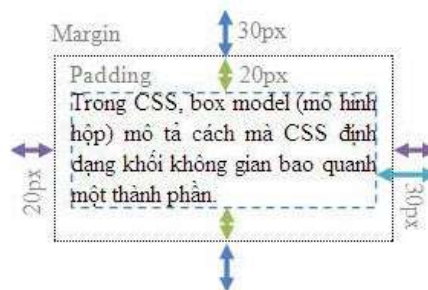
Chúng ta có một đoạn HTML sau:

<p>Trong CSS, box model (mô hình hộp) mô tả cách mà CSS định dạng khối không gian bao quanh một thành phần.  
</p>

Phần CSS cho đoạn HTML trên:

```
p {  
    width:200px;  
    margin:30px 20px;  
    padding:20px 10px;  
    border:1px solid #000;  
    text-align:justify  
}
```

Với ví dụ này chúng ta sẽ khái quát được mô hình hộp như sau:



Hình 17. Kết quả thực hiện

### 3.9. Margin & padding

Như tất cả những ai đã học qua MS Word đều biết là trong phần thiết lập Page Setup của Word cũng có một thiết lập margin để định lề cho trang in. Tương tự, thuộc tính margin trong CSS cũng được dùng để canh lề cho cả trang web hay một thành phần web này với các thành phần web khác hay với viền trang.

#### 3.9.1. Thuộc tính margin

Margin dùng để xác định khoảng cách giữa nó và đối tượng bao quanh nó. Có thể sử dụng 4 thuộc tính của margin bao gồm:

```
margin-left: length/percent/auto;  
margin-right: length/percent/auto;  
margin-top: length/percent/auto;  
margin-bottom: length/percent/auto;
```

Ví dụ sau cho biết cách canh lề cho một trang web.

```
body {  
    margin-top:80px;  
    margin-bottom:40px;  
    margin-left:50px;  
    margin-right:30px;  
    border:1px dotted #FF0000}
```

Hoặc gọn hơn sẽ viết như sau:

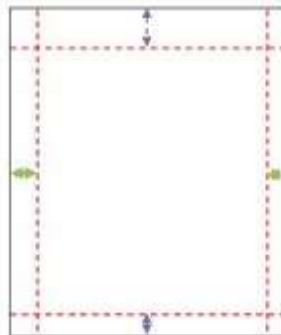
```
body {  
    margin:80px 30px 40px 50px;  
    border:1px dotted #FF0000}
```

**Cú pháp:**margin:<margin-top> | <margin-right> | <margin-bottom> | <margin-left>

Hoặc:

```
margin:<value1>|< value2>
```

Với value 1 là giá trị margin-top và margin-bottom và value2 là giá trị margin-left và margin-right. Kết quả của ví dụ trên sẽ được mô hình hóa như sau:



*Hình 18. Mô hình hóa bằng thuộc tính margin*

Ví dụ kế tiếp sẽ thể hiện rõ hơn về việc dùng margin để canh lề cho các đối tượng trong trang web. Hãy quan sát các đường viền và nhận xét.

```
body {  
    margin:80px 30px 40px 50px;  
    border:1px solid #FF0000  
}  
#box1 {  
    margin:50px 30px 20px 40px;  
    border:1px solid #00FF00  
}  
#box2 {  
    margin:50px 30px 20px 40px;  
    border:1px solid #0000FF  
}
```

### 3.9.2. Thuộc tính padding

Padding có thể hiểu như là một thuộc tính đệm. Padding không ảnh hưởng tới khoảng cách giữa các đối tượng như margin mà nó chỉ quy định khoảng cách giữa phần nội dung và viền của một đối tượng (xem lại ảnh minh họa về boxmodel – Hình 4).

**Cú pháp:** <padding-top> | <padding-right> | <padding-bottom> | <padding-left>

### 3.10. Border

Thuộc tính Border (đường viền) trong css cho phép người dùng định dạng đường viền cho các phần tử dạng khối (block) trong tài liệu (X)HTML.

**Thuộc tính border-width:** được sử dụng để chỉ định độ dày của đường viền

**Cú pháp:**

border-width: value;

Trong đó giá trị value có thể là: Length, Thin, Medium, Thick

**Thuộc tính border-color:** được sử dụng để thiết lập màu sắc cho đường viền

**Cú pháp:**

border-color: value;

Trong đó giá trị value có thể là: tên màu, tên màu ở dạng Hexa, RGB color code, transparent – trong suốt, không màu.

**Thuộc tính border-style:** được sử dụng để thiết lập kiểu đường viền(đường thẳng, đường gạch nổi,...)

**Cú pháp:**

border-style: value;

Trong đó value có thể là: dashed, dotted, double, groove, hidden, inset, none, outset, ridge, solid

### 3.11. Height & width

#### 3.11.1. Thuộc tính width

Width là một thuộc tính CSS dùng để quy định chiều rộng cho một thành phần web. Ví dụ sau sẽ định chiều rộng cho thành phần p của một trang web.

Ví dụ: `p{ width:700px; }`

**Thuộc tính max-width:** là thuộc tính CSS dùng để quy định chiều rộng tối đa cho một thành phần web.

**Cú pháp:**

Thẻ\_tag{ max-width: giá trị; }

Với giá trị như sau:

Thuộc tính	Giá trị	Ví dụ	Mô tả
max-width	Đơn vị	max-width: 50px;	Chiều cao tính từ mép bên trái ngoài cùng của thành phần, đơn vị có thể là px, em, %, ...
	none	max-width: none;	Không sử dụng chiều rộng lớn nhất, đây là dạng mặc định.
	inherit	max-width: inherit;	Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài).

Ví dụ

<html>

<body>

```

        <div>
            <p>Đoạn text này có chiều rộng lớn nhất
            là 120px.</p>
        </div>
    </body>
</html>

```

Giả sử bạn đầu CSS viết:

```
div { border: 1px solid red; }
```

Hiện thị trình duyệt khi chưa có thuộc tính max-width:

Đoạn text này có chiều rộng lớn nhất là 120px.

Thêm thuộc tính max-width vào CSS:

```
div {
    border: 1px solid red;
    max-width: 120px;
}
```

Hiện thị trình duyệt khi đã thêm max-width vào CSS:

Đoạn text này có  
chiều rộng lớn  
nhất là 120px.

Khi đã sử dụng thuộc tính max-width thì chiều rộng của box sẽ không vượt quá giá trị của max-width

**Thuộc tính min-width:** thiết lập chiều rộng tối thiểu (nhỏ nhất) cho thành phần.

**Cú pháp:**

```
Thẻ_tag{ min-width: giá trị; }
```

Với giá trị như sau:

Thuộc tính	Giá trị	Ví dụ	Mô tả
min-width	đơn vị	min-width: 200px;	Chiều rộng tối thiểu tính từ mép trên cùng của thành phần, đơn vị có thể là px, em, %, ...
	inherit	min-width: inherit;	Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài).

Ví dụ

```

<html>
    <body>
        <div>
            <ul>
                <li><a href="html4-xhtml.php">Tham khảo
                html4</a></li>
            </ul>
        </div>
    </body>
</html>

```

```

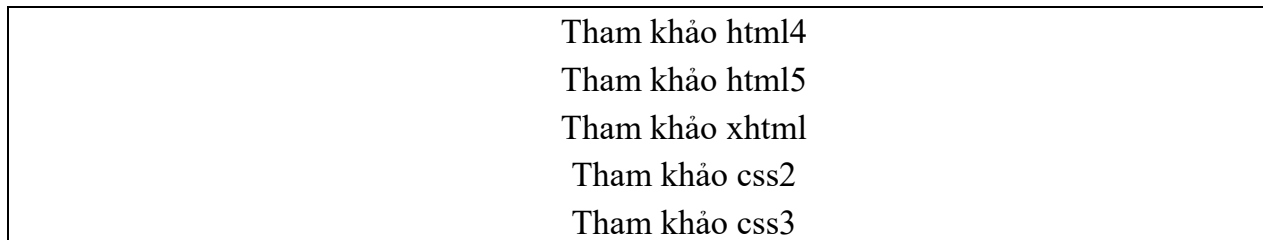
        <li><a href="html5/">Tham khảo html5</a></li>
        <li><a href="cssSection/">Tham khảo
css2</a></li>
        <li><a href="cssSection/">Tham khảo
css3</a></li>
        <li><a href="tag/">Tham khảo jquery</a></li>
    </ul>
</div>
</body>
</html>

```

Giả sử ban đầu CSS viết:

```
div { border: 1px solid red; }
```

Hiện thị trình duyệt khi chưa có thuộc tính min-width:

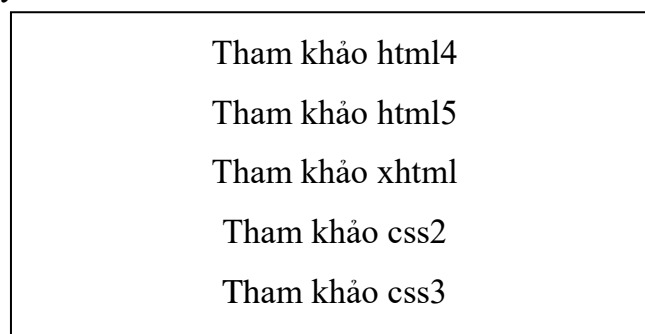


Hình 7. Kết quả hiển thị không sử dụng thuộc tính min-width

Thêm thuộc tính min-width vào CSS:

```
div {
    border: 1px solid red;
    min-width: 180px;
}
```

Hiện thị trình duyệt khi đã thêm min-width vào CSS:



Khi nội dung dài thì chiều rộng của box được tăng thêm, nhưng khi nội dung ngắn thì giá trị chiều rộng nhỏ nhất sẽ bằng với giá trị min-width, chiều rộng lúc này không thể nhỏ hơn giá trị min-width.

### 3.11.2. Thuộc tính height

Thuộc tính height thiết lập chiều cao cho thành phần. Chiều cao này không bao gồm: *border, padding, margin*.

**Cú pháp:**

Thẻ\_tag { height: giá trị; }

Với giá trị như sau:

Thuộc tính	Giá trị	Ví dụ	Mô tả
height	đơn vị	height: 100px;	Định rõ đơn vị cho chiều cao.
	auto	height: auto;	Định chiều cao tự động, đây là dạng mặc định.
	inherit	height: inherit;	Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài).

*Ví dụ*

```
<html>
  <body>
    <p>HỌC WEB CHUẨN</p>
  </body>
</html>
```

Hiển thị trình duyệt khi chưa có CSS:

HỌC WEB CHUẨN
---------------

CSS viết:

```
p {
  background: #cccccc;
  height: 100px;
}
```

Hiển thị trình duyệt khi có CSS:

HỌC WEB CHUẨN
---------------

**Thuộc tính max-height:** thiết lập chiều cao tối đa cho thành phần, về cách sử dụng tương tự như max-width tuy nhiên khi áp dụng sẽ tương tác với chiều cao

**Cú pháp:**

Thẻ\_tag { max-height: giá trị; }

Với giá trị như sau:

Thuộc tính	Giá trị	Ví dụ	Mô tả
max-height	đơn vị	max-height: 50px;	Chiều cao tối đa tính từ mép trên cùng của thành phần, đơn vị có thể là px, em, %, ...
	none	max-height: none;	Không sử dụng chiều cao lớn nhất, đây là dạng mặc định.
	inherit	max-height: inherit;	Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài).

**Thuộc tính min-height:** thiết lập chiều cao tối thiểu (nhỏ nhất) cho thành phần

**Cú pháp:**

Thẻ\_tag { min-height: giá trị; }

Với giá trị như sau:

Thuộc tính	Giá trị	Ví dụ	Mô tả
min-height	đơn vị	min-height: 200px;	Chiều cao tối thiểu tính từ mép trên cùng của thành phần, đơn vị có thể là px, em, %, ...
	inherit	min-height: inherit;	Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài).

**3.12. Float & clear****3.12.1. Thuộc tính float**

Thuộc tính float xác định có hay không một thành phần được float (trôi nổi).

**Cú pháp:**

Thẻ\_tag { float: giá trị; }

Với giá trị như sau:

Thuộc tính	Giá trị	Ví dụ	Mô tả
Float	left	float: left;	Thành phần được trôi nổi (float) qua bên trái....
	right	float: right;	Thành phần được trôi nổi (float) qua bên phải.
	none	float: none;	Thành phần không được trôi nổi (float) qua bên phải hay trái, đây là dạng mặc định.
	inherit	float: inherit;	Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài).

Ví dụ

```
<html><body>
    <p>Float left</p>
    <p>Float left</p>
</body>
</html>
```

Hiển thị trình duyệt khi chưa có CSS:

Float left Float left
--------------------------



CSS viết:

```
p {  
    float: left;  
}
```

Hiển thị trình duyệt khi có CSS:

Float left Float left

### 3.12.2. Thuộc tính clear

Thuộc tính clear xác định 2 bên của phần tử (left, right), nơi mà phần tử float không được cho phép (ngăn cản thành phần không được float trái, phải hay cả hai).

**Cú pháp:**

Thẻ\_tag {clear: giá trị; }

Với giá trị như sau:

Thuộc tính	Giá trị	Ví dụ	Mô tả
clear	left	clear: left;	Bên trái của thành phần không được float.
	right	clear: right;	Bên phải của thành phần không được float
	both	clear:both	Bên trái và phải của thành phần không được float
	none	clear: none;	Đây là mặc định của thành phần clear, bên trái và phải của thành phần được float
	inherit	clear: inherit;	Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài).

Ví dụ

```
<html>  
  <body>  
    <p class="exFloat">  
        
    </p>  
    <p>Dòng text này không sử dụng thuộc tính clear, nên bị ảnh hưởng float của thành phần p có class = exFloat</p>  
    <p class="exClear">Dòng text này có sử dụng thuộc tính clear, nên không bị ảnh hưởng float của thành phần p class=exFloat</p>  
  </body>  
</html>
```

CSS viết:

Giả sử ta có một thành phần dùng float: left; như sau

```
p.exFloat {float: left;}
```



Dòng text này không sử dụng thuộc tính clear, nên bị ảnh hưởng float của thành phần p có class = exFloat

Dòng text này có sử dụng thuộc tính clear, nên không bị ảnh hưởng float của thành phần p class=exFloat

Hình 19 Ví dụ về float trong CSS

Thêm thuộc tính clear vào CSS:

```
p.exFloat {float: left;}
```

```
p.exClear {clear: left;}
```

Hiện thị trình duyệt khi dùng thuộc tính clear left:



Dòng text này không sử dụng thuộc tính clear, nên bị ảnh hưởng float của thành phần p có class = exFloat

Dòng text này có sử dụng thuộc tính clear, nên không bị ảnh hưởng float của thành phần p class=exFloat

Hình 20 Ví dụ về thuộc tính clear trong CSS

### 3.13. Position

Thuộc tính position trong CSS dùng để xác định vị trí hiển thị cho thẻ HTML và thường được dùng để xây dựng CSS cho menu đa cấp, tooltip hoặc một số chức năng khác. Position có tổng cộng 5 giá trị như bảng dưới đây:

Tên giá trị	Ý nghĩa
static	Dạng mặc định - sẽ hiển thị theo đúng thứ tự của nó (thường dùng để hủy các thuộc tính bên dưới)
relative	Định vị trí tương đối theo thẻ cha (thẻ khai báo relative) hoặc thẻ body nếu ko có khai báo
absolute	Định vị trí tuyệt đối (vị trí bao ngoài), lúc này các thẻ HTML bên trong sẽ coi nó là thẻ cha
fixed	Định vị trí tương đối cho cửa sổ Browser của trình duyệt (khi kéo scroll nó sẽ không bị ẩn đi)
inherit	Thừa hưởng các thuộc tính từ thành phần cha (thành phần bao ngoài nó)

### 3.13.1. Absolute position

Định vị tuyệt đối là sự định vị mà trong đó các thành phần được định vị không để lại bất cứ một khoảng trống nào trong tài liệu. Một thành phần được định vị tuyệt đối sẽ nhận giá trị position là absolute. Các đối tượng đã định vị tuyệt đối sẽ dùng kết hợp với các thuộc tính top, left, right, bottom để xác định tọa độ

```
<div id="container">
  <div id="relative">
    
  </div>
</div>
```

Viết đoạn mã CSS:

```
div {
  box-sizing: border-box;
}
#container {
  width: 960px;
  margin: 0 auto;
  background-color: #e6e6e6;
  border: 1px solid #333;
  padding: 15px;
}
#relative{
  width: 100%;
  border: 1px solid #333;
  background-color: red;
  height: 300px;
  position: relative;
}
#absolute {
  position: absolute;
  right: 0;
  bottom: 0;
```



Hình 21. Ví dụ về Absolute position

### 3.13.2. *Relative position*

Sự định vị tương đối cho một thành phần là sự định vị được tính từ vị trí gốc trong tài liệu. Các thành phần đã được định vị tương đối sẽ để lại khoảng không trong tài liệu. Các thành phần được định vị tương đối sẽ nhận giá trị position là relative. Hãy làm lại ví dụ trên nhưng thay absolute thành relative. Hãy ghi nhận lại vị trí 4 ảnh logo lúc áp dụng thuộc tính position là none, absolute và relative rồi rút ra nhận xét.

Ví dụ:

```

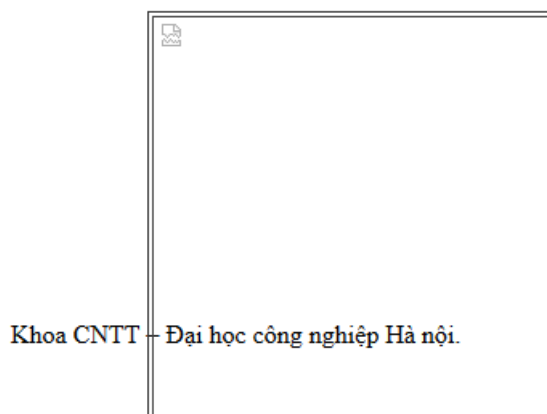
```

```
<p>Khoa CNTT - Đại học công nghiệp Hà nội.</p>
```

Viết đoạn mã CSS:

```
img {
    position: relative;
    top: 85px;
    left: 85px;
    border: 1px solid #333;
    padding: 2px;
}
```

Kết quả thể hiện:



Hình 22. Ví dụ về relative position

**TỔNG KẾT CHƯƠNG 3:** Chương này thể hiện đầy đủ cách thức trình bày một trang HTML có khoa học và cấu trúc chặt chẽ hơn thông qua việc sử dụng CSS.

#### **BÀI TẬP:**

1. Hãy tạo ra một trang web có sử dụng 4 loại CSS bao gồm CSS nội tuyến, CSS bên trong, CSS bên ngoài và CSS mặc định của trình duyệt nhằm làm rõ thứ tự ưu tiên của chúng.
2. Hãy sử dụng các thẻ HTML và CSS đã học để tạo ra một trang web có giao diện như hình dưới?

## CHƯƠNG 4: NGÔN NGỮ KỊCH BẢN JAVASCRIPT

### 4.1. Nhập môn JavaScript

**Nhúng JavaScript vào file HTML:** Bạn có thể nhúng JavaScript vào một file HTML theo một trong các cách sau đây:

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

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

**Sử dụng thẻ SCRIPT:** Script được đưa vào file HTML bằng cách sử dụng cặp thẻ **<SCRIPT>** và **</SCRIPT>**. Các thẻ **<SCRIPT>** có thể xuất hiện trong phần **<HEAD>** hay **<BODY>** của file HTML. Nếu đặt trong phần **<HEAD>**, nó sẽ được tải và sẵn sàng trước khi phần còn lại của văn bản được tải. Với công nghệ hiện nay thẻ **<SCRIPT>** luôn được thiết lập thuộc tính **LANGUAGE** mặc định là **JAVASCRIPT**.

Điểm khác nhau giữa cú pháp viết các ghi chú giữa HTML và JavaScript là cho phép ẩn các mã JavaScript trong các ghi chú của file HTML, để các trình duyệt cũ không hỗ trợ cho JavaScript có thể đọc được nó như trong ví dụ sau đây:

```
<SCRIPT>
    <!--Mã lệnh Javascript sẽ bị ẩn từ đây
    // Các lệnh Javascript
    // Đây là đoạn kết thúc ẩn mã lệnh -->
</SCRIPT>
```

Dòng cuối cùng của script cần có dấu **//** để trình duyệt không diễn dịch dòng này dưới dạng mã JavaScript.

#### 4.1.1. Sử dụng một file nguồn JavaScript

Thuộc tính **SRC** của thẻ **<SCRIPT>** cho phép chỉ rõ file nguồn JavaScript được sử dụng.

#### Cú pháp:

```
<SCRIPT SRC="file_name.js">
```

....

```
</SCRIPT>
```

Thuộc tính này rấy hữu dụng cho việc chia sẻ các hàm dùng chung cho nhiều trang khác nhau. Các câu lệnh JavaScript nằm trong cặp thẻ **<SCRIPT>** và **</SCRIPT>** có chứa thuộc tính **SRC** trừ khi nó có lỗi. Ví dụ nếu muốn đưa dòng lệnh sau vào giữa cặp thẻ **<SCRIPT SRC="...">** và **</SCRIPT>** có thể viết:

```
document.write("Không tìm thấy file JS đưa vào!");
```

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

```
<SCRIPT SRC=" http://cse.com.vn ">
```

Các file JavaScript bên ngoài không được chứa bất kỳ thẻ HTML nào. Chúng chỉ được chứa các câu lệnh JavaScript và định nghĩa hàm. Tên file của các hàm JavaScript bên ngoài cần có đuôi **.js**, và server sẽ phải ánh xạ đuôi **.js** đó tới kiểu MIME **application/x-javascript**. Đó là những gì mà server gửi trở lại phần Header của file HTML. Để ánh xạ đuôi này vào kiểu MIME, ta thêm dòng sau vào file **mime.types** trong đường dẫn cấu hình của server, sau đó khởi động lại server:

**type=application/x-javascript**

Nếu server không ánh xạ được đuôi **.js** tới kiểu MIME **application/x-javascript**, Navigator sẽ file JavaScript được chỉ ra trong thuộc tính **SRC** về không đúng cách.

#### 4.1.2. *Hiển thị một dòng text*

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

Hơn nữa, JavaScript còn cho phép người lập trình sinh ra một hộp thông báo hoặc xác nhận gồm một hoặc hai nút. Ngoài ra, dòng text và các con số còn có thể hiển thị trong trường **TEXT** và **TEXTAREA** của một form.

Trong phần này, ta sẽ học cách thức **write()** và **writeln()** của đối tượng **document**. Đối tượng **document** trong JavaScript được thiết kế sẵn hai cách thức để xuất một dòng text ra màn hình client: **write()** và **writeln()**. Cách gọi một phương thức của một đối tượng như sau:

**object\_name.method\_name**

Dữ liệu mà phương thức dùng để thực hiện công việc của nó được đưa vào dòng tham số, ví dụ:

**document.write("Test");**

**document.writeln("Test");**

Ví dụ: Cách thức **write()** và **writeln()** xuất ra thẻ HTML

<pre> &lt;HTML&gt; &lt;HEAD&gt; &lt;TITLE&gt;Outputting Text&lt;/TITLE&gt; &lt;/HEAD&gt; &lt;BODY&gt; This text is plain&lt;B&gt; &lt;PRE&gt; &lt;SCRIPT LANGUAGE="JavaScript"&gt; &lt;!-- HIDE FROM OTHER BROWSERS document.writeln("This text is bold."); document.write("This text is bold.&lt;/B&gt;"); // STOP HIDING FROM OTHER BROWSERS --&gt; &lt;/SCRIPT&gt; &lt;/PRE&gt; &lt;/BODY&gt;&lt;/HTML&gt; </pre>	<pre> This text is plain This text is bold. This text is bold. </pre>
--	---

Hình 22. Phương thức write

#### 4.1.3. Giao tiếp với người sử dụng

JavaScript hỗ trợ khả năng cho phép người lập trình tạo ra một hộp hội thoại. Nội dung của hộp hội thoại phụ thuộc vào trang HTML có chứa đoạn script mà không làm ảnh hưởng đến việc xuất nội dung trang.

Cách đơn giản để làm việc đó là sử dụng cách thức **alert()**. Để sử dụng được cách thức này, bạn phải đưa vào một dòng text như khi sử dụng `document.write()` và `document.writeln()` trong phần trước. Ví dụ: `alert("Nhấn vào OK để tiếp tục");`



Hình 23. Hộp alert

Khi đó file sẽ chờ cho đến khi người sử dụng nhấn vào nút OK rồi mới tiếp tục thực hiện. Thông thường, cách thức **alert()** được sử dụng trong các trường hợp:

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

Tuy nhiên cách thức **alert()** mới chỉ cho phép thông báo với người sử dụng chứ chưa thực sự giao tiếp với người sử dụng. JavaScript cung cấp một cách thức khác để giao tiếp với người sử dụng là **prompt()**. Tương tự như **alert()**, **prompt()** tạo ra một hộp hội thoại với một dòng thông báo do bạn đưa vào, nhưng ngoài ra nó còn cung cấp một trường để nhập dữ liệu vào. Người sử dụng có thể nhập vào trường đó rồi kích vào OK. Khi đó có thể xử lý dữ liệu do người sử dụng vừa đưa vào.

Ví dụ: Hộp hội thoại gồm một dòng thông báo, một trường nhập dữ liệu, một nút OK và một nút Cancel. Chương trình này sẽ hỏi tên người dùng và sau đó sẽ hiển thị một thông báo ngắn sử dụng tên mới đưa vào. Ví dụ được lưu vào file Hello.html

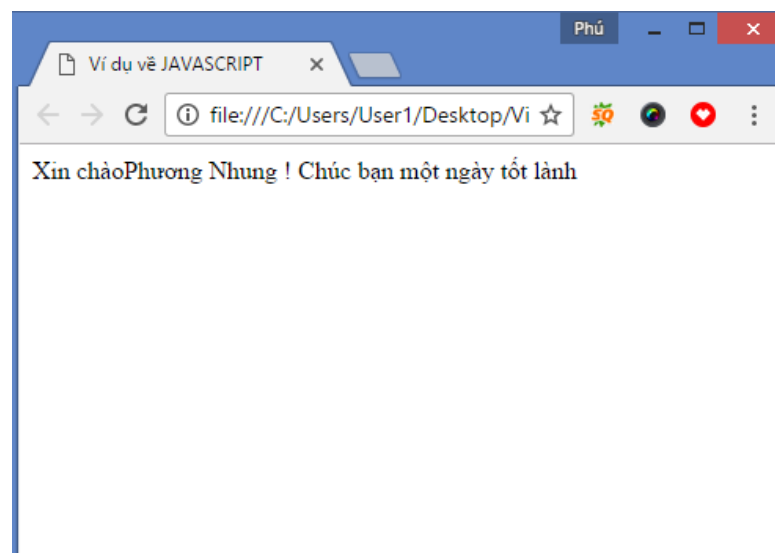
```
<HTML>
<HEAD>
  <TITLE>Ví dụ về JAVASCRIPT</TITLE>
  <SCRIPT LANGUAGE="JavaScript">
    var name=window.prompt("Xin chào tên bạn là gì?", "");
    document.write("Xin chào " + name + " ! Chúc bạn một ngày tốt lành ");
  </SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```

Khi duyệt có kết quả:



Hình 24: Hiển thị cửa sổ nhập tên

Ví dụ này hiển thị dấu nhắc nhập vào tên với phương thức **window.prompt**. Giá trị đạt được sẽ được ghi trong biến có tên là *name*. Biến *name* được kết hợp với các chuỗi khác và được hiển thị trong cửa sổ của trình duyệt nhờ phương thức **document.write**.



Hình 25. Kết hợp phương thức write và prompt

Hộp thoại **confirm** hỏi người dùng câu hỏi Yes – No và cung cấp tùy chọn cho người dùng trả lời qua 02 nút ấn OK và CANCEL. Hộp thoại **confirm** trả về một trong hai giá trị giá trị đúng (True) hoặc sai (False), kết quả trả về này do người dùng xác định cho đến khi hộp thoại được đóng lại.

Ví dụ:

```
var truthBeTold = window.confirm("Ấn OK để tiếp  
tục. Cancel để thoát");  
if (truthBeTold) {  
    window.alert("Chào mừng bạn đến trang web của  
chúng tôi!");  
} else window.alert("Chào tạm biệt!");
```



## 4.2. Biến trong Javascript

### 4.2.1. Biến và phân loại biến

Tên biến trong JavaScript phải bắt đầu bằng chữ hay dấu gạch dưới. Các chữ số không được sử dụng để mở đầu tên một biến nhưng có thể sử dụng sau ký tự đầu tiên. Phạm vi của biến có thể là một trong hai kiểu sau:

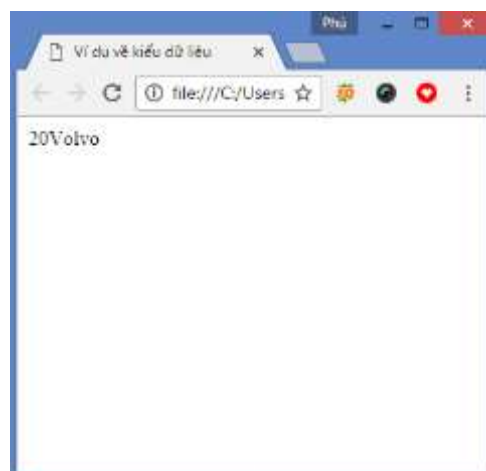
- *Biến toàn cục*: Có thể được truy cập từ bất kỳ đâu trong ứng dụng. được khai báo như sau: **x = 0;**
- *Biến cục bộ*: Chỉ được truy cập trong phạm vi chương trình mà nó khai báo. Biến cục bộ được khai báo trong một hàm với từ khoá **var** như sau:  
**var x = 0;**

### 4.2.2. Kiểu dữ liệu

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

Ví dụ file *javascript2.html*:

```
<HTML><HEAD>
<TITLE> Ví dụ về kiểu dữ liệu
</TITLE>
<SCRIPT LANGUAGE= "JavaScript">
var x = 16 + 4 + "Volvo";
document.write(x);
</SCRIPT>
</HEAD><BODY></BODY></HTML>
```



Hình 26. Kiểu dữ liệu

Kiểu dữ liệu nguyên thủy là một giá trị mà trong đó không chứa phương thức và thuộc tính. Kiểu dữ liệu nguyên thủy sẽ mang giá trị nguyên thủy. Trong javascript định nghĩa gồm 5 kiểu dữ liệu nguyên thủy bao gồm : xâu ký tự (string), kiểu số (number), kiểu logic (boolean), kiểu dữ liệu trống (null) và kiểu dữ liệu không định nghĩa (undefined).

**Kiểu số (Số nguyên và số thực):** Số nguyên (Integer) có thể được biểu diễn theo ba cách: *Hệ cơ số 10* (hệ thập phân), *hệ cơ số 8* (hệ bát phân), *hệ cơ số 16* (hệ thập lục phân). Số thực (Floating Point): **phần nguyên thập phân**, dấu chấm thập phân (.), phần dư, phần mũ. Để phân biệt kiểu dấu phẩy động với kiểu số nguyên, phải có ít nhất một chữ số theo sau dấu chấm hay **E**. Ví dụ: -0.85E4.

**Kiểu logic (Boolean):** được sử dụng để chỉ hai điều kiện : đúng hoặc sai. Miền giá trị của kiểu này chỉ có hai giá trị: true, false.

**Kiểu chuỗi (String):** Một literal kiểu chuỗi được biểu diễn bởi không hay nhiều ký tự được đặt trong cặp dấu " ... " hay '... '. Ví dụ: "The dog ran up the tree"

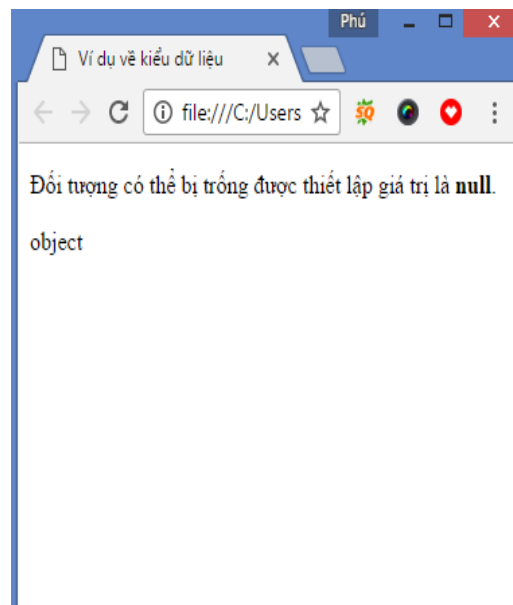
Để biểu diễn dấu nháy kép ( " ), trong chuỗi sử dụng ( \ " ), ví dụ: document.write(“ \”This text inside quotes.\””);

**Kiểu không định nghĩa (Undefined):** là khi một biến không có giá trị, chứa giá trị không xác định.

**Kiểu dữ liệu trống (NULL):** trong javascript kiểu dữ liệu trống là không có gì (nothing) chứa bên trong. Kiểu dữ liệu của biến có giá trị là null thường là object.

*Ví dụ:*

```
<TITLE> Ví dụ về kiểu dữ liệu
</TITLE>
</HEAD><BODY>
<p>Đối tượng có thể bị trống được
thiết lập giá trị là
<b>null</b>.</p>
<p id="demo"></p>
<script>
var person = {firstName:"John",
lastName:"Doe", age:50,
eyeColor:"blue"};
var person = null;
document.getElementById("demo").inn
erHTML = typeof person;
</script>
</BODY></HTML>
```



Hình 27. Kiểu dữ liệu null

### 4.3. Xây dựng các biểu thức trong Javascript

#### 4.3.1. Định nghĩa và phân loại biểu thức

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

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

#### 4.3.2. Các toán tử

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

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

#### *Kiểu gán thông thường*

$x = x + y$

$x = x - y$

$x = x * y$

#### *Kiểu gán rút gọn*

$x += y$

$x -= y$

$x *= y$

$x = x / y$

$x /= y$

$x = x \% y$

$x \% = y$

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

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

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

- var1% var2** Toán tử phần dư, trả lại phần dư khi chia var1 cho var2
- Toán tử phủ định, có giá trị phủ định toán hạng
- var ++** Toán tử này tăng var lên 1
- var --** Toán tử này giảm var đi 1

**Toán tử chuỗi:** Khi được sử dụng với chuỗi, toán tử + được coi là kết hợp hai chuỗi, ví dụ: "abc" + "xyz" được "abcxyz"

**Toán tử Logic:** JavaScript hỗ trợ các toán tử logic sau đây:

- expr1 && expr2** Là toán tử logic AND, trả lại giá trị đúng nếu cả expr1 và expr2 cùng đúng.
- Expr1 || expr2** Là toán tử logic OR, trả lại giá trị đúng nếu ít nhất một trong hai expr1 và expr2 đúng.
- ! expr** Là toán tử logic NOT phủ định giá trị của expr.

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

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

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

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

Có một số toán tử dịch chuyển bitwise rút gọn:

**Kiểu bitwise thông thường**

**Kiểu bitwise rút gọn**

$x = x \ll y$

$x \ll = y$

$x = x \gg y$

$x \gg = y$

$x = x \ggg y$

$x \ggg = y$

$x = x \& y$

$x \& = y$

$x = x \wedge y$

$x \wedge = y$

$x = x | y$

$x | = y$

- **Toán tử ?:** cú pháp sau:

**(condition) ? valTrue : valFalse**

Trong đó :

- Nếu điều kiện *condition* được đánh giá là đúng, biểu thức nhận giá trị *valTrue*, ngược lại nhận giá trị *valFalse*. Ví dụ:  
state = (temp>32) ? "liquid" : "solid"
- Trong ví dụ này biến state được gán giá trị "liquid" nếu giá trị của biến temp lớn hơn 32; trong trường hợp ngược lại nó nhận giá trị "solid".

## 4.4. Các lệnh

### 4.4.1. Câu lệnh điều kiện

Câu lệnh điều kiện cho phép chương trình ra quyết định và thực hiện công việc nào đây dựa trên kết quả của quyết định. Trong JavaScript, câu lệnh điều kiện là *if...else* và *switch ... case*

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

**Cú pháp**

*if ( <biểu thức điều kiện> )*

{

*//Các câu lệnh với điều kiện đúng*

```

    }
    else
    {
        //Các câu lệnh với điều kiện sai
    }

```

*Ví dụ:*

```

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

```

**Chú ý:** Ký tự { và } được sử dụng để tách các khối mã.

**Câu lệnh switch ... case:** Switch so sánh một biểu thức nguyên với một danh sách giá trị các số nguyên, các hằng kí tự hoặc biểu thức hằng. Mỗi giá trị trong danh sách chính là một case label (nhãn trường hợp) trong khối mã lệnh của switch. Ngoài ra, trong khối mã lệnh của switch còn có thể có một default label (nhãn mặc định) - có thể có hoặc không. Mặt khác, trong mỗi label còn chứa các khối mã lệnh chờ được thực thi

**Cú pháp:**

```

switch (<biến điều khiển>) {
    case "nhãn 1":
        //Các câu lệnh với nhãn 1;
        break;
    case nhãn 2:
        //Các câu lệnh với nhãn 2;
        break;
    .....
    default:
        //Các câu lệnh trong trường hợp mặc định;
}

```

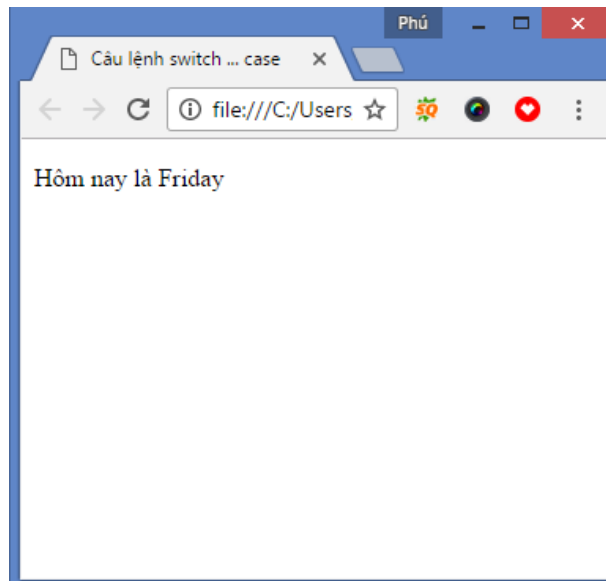
*Ví dụ:*

```

<HTML><TITLE> Câu lệnh switch ... case </TITLE>
</HEAD>
<BODY>
<p id="demo"></p>
<script>
var day;
switch (new Date().getDay()) {
    case 0:day = "Sunday";break;
    case 1:day = "Monday";break;
    case 2:day = "Tuesday";break;
    case 3:day = "Wednesday";break;
    case 4:day = "Thursday";break;
    case 5:day = "Friday";break;
    case 6:day = "Saturday";
}

```

```
document.getElementById("demo").innerHTML = "Hôm nay là "
+ day;
</script>
</BODY></HTML>
```



Hình 28. Câu lệnh switch ... case

#### 4.4.2. Câu lệnh lặp

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

- for loop
- while loop
- do while

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

**Cú pháp:**

***for (initExpr; <biểu thức điều kiện> ; incrExpr){***

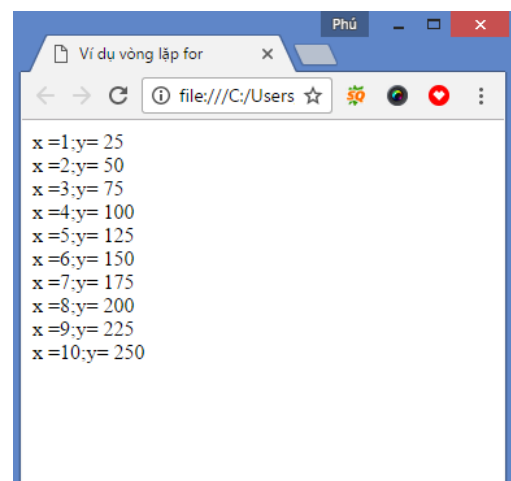
***//Các lệnh được thực hiện trong***

***khi lặp***

***}***

***Ví dụ:***

```
<HTML> <HEAD>
<TITLE>For loop Example </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
for (x=1; x<=10 ; x++) {
    y=x*25;
    document.write("x =" + x
+"; y= " + y + "<BR>");
}
</SCRIPT>
</HEAD><BODY></BODY></HTML>
```



Hình 29: Kết quả của lệnh for...loop

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

**Cú pháp:**

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

*Ví dụ:*

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

Kết quả của ví dụ này giống như ví dụ trước.

**Câu lệnh do ... while:** thực hiện giống như vòng lặp while nhưng khác là được thực hiện ít nhất 1 lần.

**Cú pháp:**

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

*Ví dụ:*

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

Kết quả của ví dụ này giống như vòng lặp while

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

**Cú pháp:**

*break;*

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

*Ví dụ:*

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





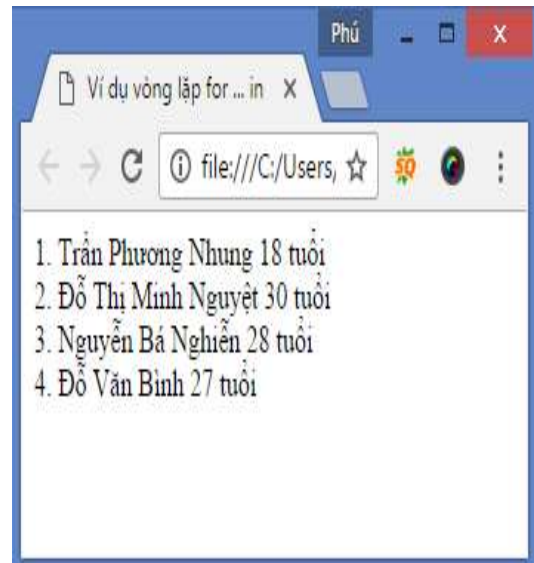
**Câu lệnh new:** được thực hiện để tạo ra một thể hiện mới của một đối tượng

**Cú pháp:**

***objectvar = new object\_type ( param1 [,param2]... [,paramN])***

Ví dụ sau tạo đối tượng **person** có các thuộc tính *firstname*, *lastname*, *age*, *sex*. Chú ý rằng từ khoá **this** được sử dụng để chỉ đối tượng trong hàm **person**. Sau đó ba thể hiện của đối tượng **person** được tạo ra bằng lệnh **new**

```
<HTML>
<HEAD>
<TITLE>New Example </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
function person(first_name, last_name, age, sex){
this.first_name=first_name;
this.last_name=last_name;
this.age=age;
this.sex=sex;}
person1= new person("Nhưng",
"Trần Phương", "18",
"Female");
person2= new person("Nguyệt",
"Đỗ Thị Minh", "30",
"Female");
person3= new person("Nghien",
"Nguyễn Bá", "28", "Male");
person4= new person("Bình",
"Đỗ Văn", "27", "Male");
document.write ("1.
"+person1.last_name+" " +
person1.first_name + "<BR>" );
document.write("2. "+person2.last_name + " "+
person2.first_name + "<BR>");
document.write("3. "+ person3.last_name + " "+
person3.first_name + "<BR>");
document.write("4. "+ person4.last_name + " "+
person4.first_name+"<BR>");
</SCRIPT>
</HEAD>
<BODY>
</BODY></HTML>
```



Hình 32. Kết quả của ví dụ lệnh New

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

**Cú pháp:**

***this [.property]***

Có thể xem ví dụ của lệnh new.

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

### Cú pháp

*with (đối\_tượng)*

```
{  
    //Các câu lệnh thực hiện  
}
```

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

```
<HTML>  
<HEAD>  
<TITLE>Ví dụ lệnh with</TITLE>  
<SCRIPT LANGUAGE= "JavaScript">  
with (document){  
    write("Đây là ví dụ có thể thực hiện tốt <BR>");  
    write("với lệnh <B>with<B> <P>");  
    write("Chúc các bạn một ngày tốt lành");  
}  
</SCRIPT>  
</HEAD>  
<BODY>  
</BODY>  
</HTML>
```



Hình 33: Kết quả của ví dụ lệnh with

### 4.5. Các hàm(Functions)

JavaScript cũng cho phép sử dụng các hàm. 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.

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

#### Cú pháp

```
function fnName([param1],[param2],...,[paramN])  
{  
    //Các câu lệnh của hàm  
}
```

Có thể xem lại ví dụ trong lệnh new

### 4.6. Các hàm có sẵn

JavaScript có một số hàm có sẵn, gắn trực tiếp vào chính ngôn ngữ và không nằm trong một đối tượng nào. Các hàm này bao gồm: **eval**, **parseInt**, **parseFloat**.

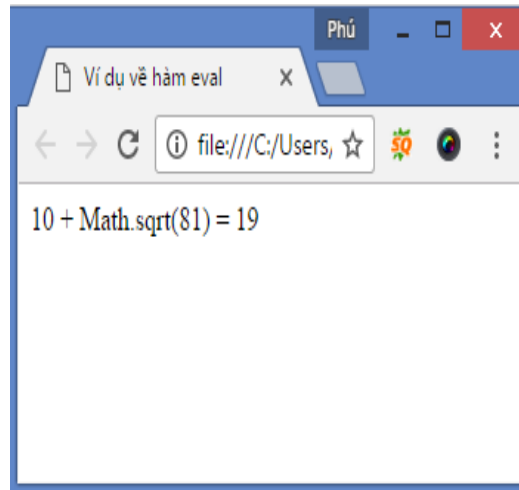
**Hàm eval:** Hàm này được sử dụng để đánh giá các biểu thức hay lệnh. Biểu thức, lệnh hay các đối tượng của thuộc tính đều có thể được đánh giá.

#### Cú pháp:

*returnval=eval (bất kỳ biểu thức hay lệnh hợp lệ trong Java)*

*Ví dụ:*

```
<HTML>
<HEAD>
<TITLE>Ví dụ về hàm Eval
</TITLE>
<SCRIPT LANGUAGE=
"JavaScript">
    var string="10+
Math.sqrt(81)";
    document.write(string+
"="+ eval(string));
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```



Hình 34 Ví dụ hàm Eval

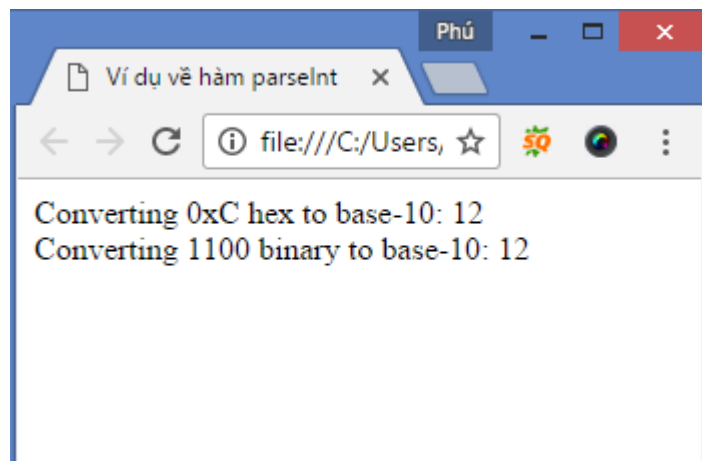
**Hàm parseInt:** Hàm này chuyển một chuỗi số thành số nguyên với cơ số là tham số thứ hai (tham số này không bắt buộc). Hàm này thường được sử dụng để chuyển các số nguyên sang cơ số 10 và đảm bảo rằng các dữ liệu được nhập dưới dạng ký tự được chuyển thành số trước khi tính toán. Trong trường hợp dữ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

**Cú pháp**

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

*Ví dụ:*

```
<HTML>
<HEAD>
<TITLE> parseInt Exemple </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
document.write("Converting 0xC hex to base-10: " +
parseInt(0xC,10) + "<BR>");
document.write("Converting 1100 binary to base-10: " +
parseInt(1100,2) + "<BR>");
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```



Hình 35: Ví dụ parseInt

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

**Cú pháp**

**parseFloat**

**(string)**

*Ví dụ:* Mô tả cách thức xử lý

của parseFloat với các kiểu chuỗi khác nhau.

```
<HTML> <HEAD>
<TITLE> perseFload Exemple </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
    document.write("This script will show how diffrent
        strings are ");
    document.write("Converted using parseFloat<BR>");
    document.write("137= " + parseFloat("137") + "<BR>");
    document.write("137abc= " + parseFloat("137abc") +
        "<BR>");
    document.write("abc137= " + parseFloat("abc137") +
        "<BR>");
    document.write("1abc37= " + parseFloat("1abc37") +
        "<BR>");
</SCRIPT>
</HEAD>
<BODY> </BODY>
</HTML>
```

#### 4.7. Mảng(Array)

Mảng là một biến đặc biệt để cung cấp nhiều giá trị trong cùng một thời điểm. Nếu muốn liệt kê ra các thành phần của mảng sẽ thấy việc lưu trữ giống như các biến độc lập. Ví dụ: var comp1="dell"; comp2="Toshiba"

Tuy nhiên, nếu muốn duyệt qua tất cả các biến comps thì ta cần làm gì? Và nếu như không phải chỉ có 2 biến mà là 200 biến thì sẽ thực hiện ra sao?

Việc giải quyết này sẽ được giải quyết ở mảng, mảng sẽ lưu trữ rất nhiều giá trị dưới cùng một tên và để lấy giá trị của mảng chỉ cần lấy thông qua chỉ số của mảng. Việc khởi tạo mảng thực hiện rất đơn giản theo cú pháp:

**var biến\_mảng = [thành\_phần1, thành\_phần2, ...];**

Khi thực hiện khởi tạo có các khoảng trắng và giữa các dòng cũng không quá quan trọng vì việc khai báo có thể thực hiện trên nhiều dòng. Ví dụ:

```
var comps=[
    "dell",
    "toshiba",
    "lenovo"
];
```

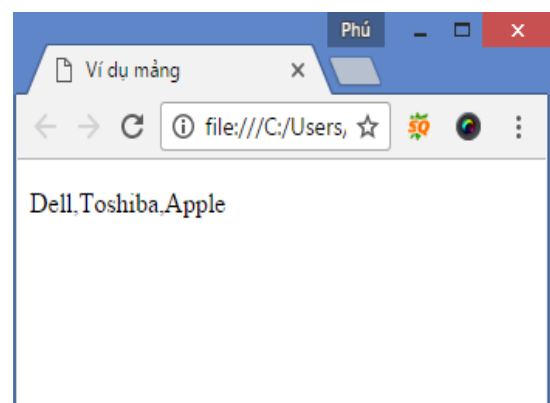
Ngoài ra, việc khai báo cũng có thể thực hiện thông qua câu lệnh **new**, việc thực hiện khai báo theo cú pháp sau:

**var biến\_mảng = new Array(thành\_phần1, thành\_phần2 ....);**

Để truy xuất vào các thành phần thuộc mảng được thực hiện thông qua các chỉ số của mảng. Chỉ số đầu tiên được xác định trong mảng là 0, nếu muốn lấy giá trị của tất cả các thành phần trong mảng chỉ cần sử dụng tên biến mảng.

*Ví dụ:*

```
<HTML>
<TITLE> Ví dụ mảng </TITLE>
```



Hình 36: Ví dụ mảng

```

<body>
<p id="demo"></p>
<script>
var comps = ["Dell", "Toshiba", "Apple"];
document.getElementById("demo").innerHTML = comps;
</script>
</BODY></HTML>

```

Mảng cũng có thể là một kiểu đối tượng đặc biệt, toán tử typeof trong javascript sẽ trả về đối tượng kiểu mảng, tuy nhiên để tốt nhất nên thực hiện mô tả từ thành phần trong mảng thông qua tên.

*Ví dụ:*

```

<HTML>
<TITLE> Ví dụ truy xuất thông qua tên thành phần của mảng
</TITLE>
<body>
<p id="demo"></p>

<script>
var person = {firstName:"Nhưng", lastName:"Trần Phương",
age:18};
document.getElementById("demo").innerHTML =
person["firstName"];
</script>

```

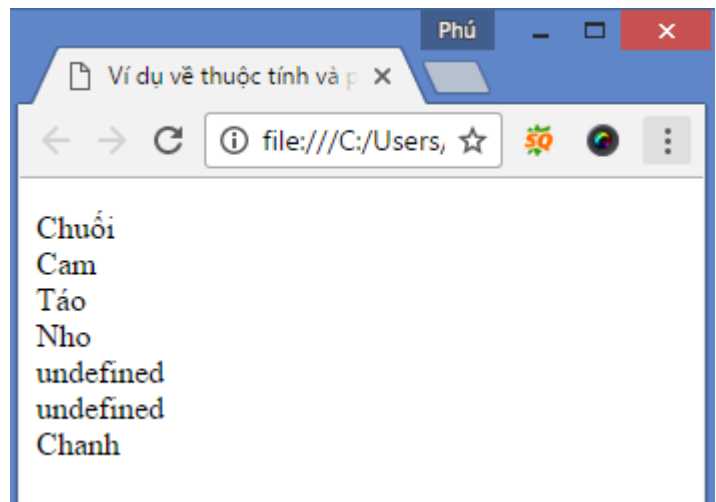
Điểm đặc biệt của mảng đó chính là các phương thức và thuộc tính, chúng được xây dựng trong nội tại của mảng. Các phương thức và thuộc tính bao gồm: toString(), join(), pop(), push(), shift(), unshift(), delete, splice(), concat(), slice(), sort(), reverse(), length.

*Ví dụ:*

```

<HTML>
<TITLE> Ví dụ về thuộc tính và phương thức của mảng
</TITLE>
<body>
<p id="demo"></p>
<script>
var fruits, text, fLen,
i;
fruits      = ["Chuối",
"Cam", "Táo", "Nho"];
fruits[6] = "Chanh";
fLen = fruits.length;
text = "";
for (i = 0; i < fLen;
i++) {
    text += fruits[i] + "<br>";
}
document.getElementById("demo").innerHTML = text;

```



Hình 37. Ví dụ về thực hiện thuộc tính và phương thức của mảng

</script></BODY></HTML>

**Chú ý:** Việc thêm thành phần vào mảng trong đó có các thành phần trống ở giữa sẽ được nhận giá trị là *undefined*.

#### 4.8. Các đối tượng trong JavaScript

Như đã nói JavaScript là ngôn ngữ lập trình *dựa trên đối tượng*, nhưng không *hướng đối tượng* bởi vì nó không hỗ trợ các lớp cũng như tính thừa kế. Có thể thấy mọi thứ trong javascript đều có thể là đối tượng (trừ kiểu dữ liệu nguyên thủy) như :

- Kiểu boolean có thể là đối tượng nếu được khai báo với từ khóa **new**
- Kiểu số có thể là đối tượng nếu được khai báo với từ khóa **new**
- Kiểu xâu ký tự có thể là đối tượng nếu được khai báo với từ khóa **new**
- Ngày tháng (Dates)
- Toán học (Maths)
- Biểu thức thông thường (Regular expressions)
- Mảng
- Hàm

Các đối tượng liên quan đến trình duyệt (BOM - Browser Object Model). Mỗi browser sẽ có những đối tượng khác nhau nên không có một chuẩn chung nào cả, tuy nhiên để có tính thống nhất giữa các trình duyệt thì người ta quy ước ra các loại BOM sau: window, screen, location, history, navigator, popup, timing, cookies.

##### 4.8.1. Đối tượng window

Đối tượng window là đối tượng được hỗ trợ trên tất cả các trình duyệt, có thể hiểu nó đại diện cho cửa sổ trình duyệt. Tất cả các đối tượng có chứa trong javascript, các hàm, các biến, đều tự động trở thành thuộc tính (thành viên) của đối tượng window.

Biến là thuộc tính, hàm là phương thức của đối tượng window, ngay cả đối tượng tài liệu (DOM HTML) là một thuộc tính của đối tượng window.

Ví dụ:

```
window.document.getElementById("header");
```

được giống như:

```
document.getElementById("header");
```

**Các thuộc tính:** có 2 thuộc tính để xác định kích thước của cửa sổ và trả về giá trị pixel đó là *window.innerHeight* – chiều cao bên trong cửa sổ và *window.innerWidth* – chiều rộng bên trong cửa sổ. Kích thước này không tính thanh công cụ (toolbars) và thanh cuộn (scrollbars)

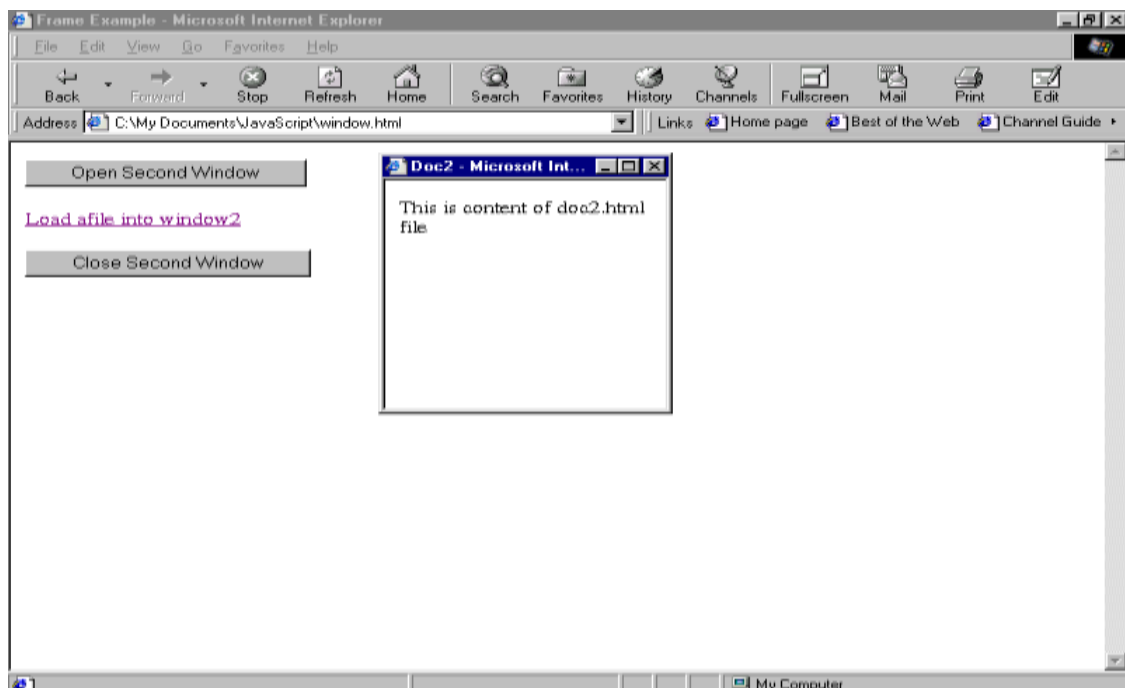
**Các phương thức:**

- `window.alert("message")` -Hiển thị hộp hội thoại với chuỗi "message" và nút OK.
- `window.close()` -Đóng cửa sổ windowReference.
- `window.confirm("message")` -Hiển thị hộp hội thoại với chuỗi "message", nút OK và nút Cancel. Trả lại giá trị True cho OK và False cho Cancel.
- `window.open()` - Mở cửa sổ mới.
- `window.prompt("message" [, "defaultInput"])` - Mở một hộp hội thoại để nhận dữ liệu vào trường text.
- `window.moveTo()` - di chuyển cửa sổ hiện tại
- `window.resizeTo()` – thay đổi kích thước cửa sổ hiện tại

**Ví dụ:** Sử dụng tên cửa sổ khi gọi tới nó như là đích của một form submit hoặc trong một Hypertext link (thuộc tính TARGET của thẻ FORM và A).

Trong ví dụ tạo ra một tới cửa sổ thứ hai, như nút thứ nhất để mở một cửa sổ rỗng, sau đó một liên kết sẽ tải file doc2.html xuống cửa sổ mới đó rồi một nút khác dùng để đóng cửa sổ thứ hai lại, ví dụ này lưu vào file *window.html*:

```
<HTML>
<HEAD>
<TITLE>Frame Example </TITLE>
</HEAD>
<BODY>
<FORM>
    <INPUT TYPE="button" VALUE="Open Second Window"
           onClick="msgWindow=window.open('','window2','resizable=no,width=200,height=200') ">
</FORM>
<P>
<A HREF="doc2.html" TARGET="window2">
Load a file into window2 </A>
</P>
<INPUT TYPE="button" VALUE="Close Second Window"
       onClick="msgWindow.close() ">
</FORM>
</BODY>
</HTML>
```



Hình 38: Minh họa cho đối tượng cửa sổ

### Các chương trình xử lý sự kiện

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



#### 4.8.2. Đối tượng navigator

Đối tượng window.navigator chứa thông tin về trình duyệt của người truy cập, khi viết ta có thể bỏ qua tiền tố window. Ví dụ như navigator.platform

##### Các thuộc tính

<i>appName</i>	Xác định tên mã nội tại của trình duyệt (Atlas).
<i>AppVersion</i>	Xác định tên trình duyệt.
<i>AppVersion</i>	Xác định thông tin về phiên bản của đối tượng navigator.
<i>userAgent</i>	Xác định header của user – agent được gửi tới máy chủ
<i>cookieEnabled</i>	Trả về giá trị đúng nếu tệp tinh cookie được kích hoạt
<i>product</i>	Trả về tên các công cụ của trình duyệt
<i>platform</i>	Trả về nền tảng của trình duyệt (hệ điều hành)
<i>language</i>	Trả về ngôn ngữ của trình duyệt
<i>online</i>	Trả về giá trị đúng nếu trình duyệt đang sử dụng

**Phương thức** `javaEnable()` trả về giá trị đúng nếu java đang được bật

Ví dụ sau sẽ hiển thị các thuộc tính của đối tượng navigator

```
<HTML>
<HEAD>
<TITLE> Ví dụ về đối tượng Navigator </TITLE>
<SCRIPT LANGUAGE= "JavaScript">
document.write("appName    =    "+navigator.appCodeName    +
"<BR>");
document.write("appName = "+navigator.appName + "<BR>");
document.write("appVersion    =    "+navigator.appVersion    +
"<BR>");
document.write("userAgent      =      "+navigator.userAgent      +
"<BR>");
document.write("cookieEnable = "+navigator.cookieEnable +
"<BR>");
document.write("product = "+navigator.product + "<BR>");
document.write("platform = "+navigator.platform + "<BR>");
document.write("language = "+navigator.language + "<BR>");
document.write("online
= "+navigator.online +
"<BR>");
document.write("javaEn
able =
"+navigator.javaEnable
());
</SCRIPT>
</HEAD>
<BODY>
</BODY>
</HTML>
```



Hình 39: Ví dụ đối tượng Navigator



#### 4.8.3. Đối tượng location

Các thuộc tính của đối tượng location duy trì các thông tin về URL của document hiện thời. Ví dụ: `http:// www.abc.com/ chap1/page2.html#topic3`

##### Các thuộc tính

- hostname - Tên của host và domain (ví dụ [www.abc.com](http://www.abc.com/) ).
- href - Toàn bộ URL cho document hiện tại.
- pathname - Phần đường dẫn của URL (ví dụ `/chap1/page2.html`).
- protocol - Giao thức được sử dụng (cùng với dấu hai chấm) (ví dụ `http:`).

**Phương thức:** `location.assign()` được sử dụng để tải lại một địa chỉ URL được xác định trong assign. Ví dụ: `location.assign("http://www.fit-hau.edu.vn")`

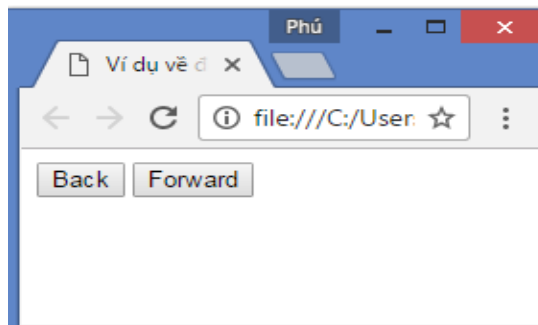
#### 4.8.4. Đối tượng history

Đối tượng này chứa đựng lịch sử của trình duyệt, để đảm bảo tính riêng tư của người dùng javascript cũng hạn chế việc truy cập vào đối tượng này. Có hai phương thức được hỗ trợ chính bao gồm:

- `history.back()`: được thực hiện tương tự như nút quay lại (back) trên trình duyệt, khi gọi phương thức này nó sẽ tải lại địa chỉ URL trước đó trong danh sách lịch sử.
- `history.forward()`: được thực hiện tương tự như nút chuyển tiếp (forward) trên trình duyệt, khi gọi phương thức này nó sẽ tải lại địa chỉ URL kết tiếp trong danh sách lịch sử.

Ví dụ:

```
<html><head>
<script>
function goForward() {
    window.history.forward()
}
function goBack() {
    window.history.back()
}
</script></head>
<body>
<input type="button" value="Back" onclick="goBack()">
    <input type="button" value="Forward" onclick="goForward()">
</body>
</html>
```



Hình 40 Ví dụ về đối tượng history

#### 4.8.5. Đối tượng document

Đối tượng document đại diện cho trang tài liệu HTML, nếu muốn truy cập vào bất kỳ một thành phần nào của HTML phải luôn thực hiện bắt đầu với đối tượng này.

Các đối tượng anchor, forms, history, links là thuộc tính của đối tượng document. Không có các chương trình xử lý sự kiện cho các frame. Sự kiện `onLoad` và `onUnload` là cho đối tượng window.

##### Các thuộc tính

- `alinkColor` - Giống như thuộc tính `ALINK`.

- anchor - Mảng tất cả các anchor trong document.
- bgColor - Giống thuộc tính BGCOLOR.
- cookie - Sử dụng để xác định cookie.
- fgColor - Giống thuộc tính TEXT.
- forms - Mảng tất cả các form trong document.
- lastModified - Ngày cuối cùng văn bản được sửa.
- linkColor - Giống thuộc tính LINK.
- links - Mảng tất cả các link trong document.
- location - URL đầy đủ của văn bản.
- referrer - URL của văn bản gọi nó.
- title - Nội dung của thẻ <TITLE>.
- vlinkColor - Giống thuộc tính VLINK.

### **Các phương thức**

- Tìm kiếm các thành phần HTML:
  - o document.getElementById(*id*): Thực hiện tìm kiếm các thành phần HTML theo id được khai báo trong thẻ HTML
  - o document.getElementsByTagName(*name*): tìm các thành phần bởi tên thẻ HTML
  - o document.getElementsByClassName(*name*): tìm các thành phần HTML bởi tên lớp (class name) được khai báo trong HTML
- Thay đổi các thành phần HTML
  - o element.innerHTML = new html content: thay đổi nội dung bên trong một thành phần HTML
  - o element.attribute = new value: thay đổi giá trị thuộc tính của thẻ HTML
  - o element.setAttribute(attribute, value): thay đổi giá trị thuộc tính của thẻ HTML
  - o element.style.property = new style: thay đổi thuộc tính style của một thẻ HTML
- Thêm mới hoặc xóa bỏ một thành phần HTML
  - o document.createElement(*element*): tạo một thẻ HTML
  - o document.removeChild(*element*): bỏ một thẻ HTML
  - o document.appendChild(*element*): thêm một thẻ HTML
  - o document.replaceChild(*element*): thay thế một thẻ HTML
  - o document.write(*text*): viết các thẻ HTML theo dạng chuỗi
- Thêm trình xử lý sự kiện (adding event handlers):
  - o document.getElementById(*id*).onclick = function(){*code*}: cho phép thêm một đoạn mã lệnh xử lý sự kiện cho một thành phần HTML thông qua sự kiện nhấp chuột (click event)

#### 4.8.6. Đối tượng forms

Đối tượng form được tạo ra bởi thẻ `<form>` trong HTML và để truy xuất vào các thành phần của một form có thể thực hiện phương thức **getElementById**. Để khởi tạo đối tượng form bằng javascript có thể sử dụng phương thức **createElement** của đối tượng document. Ví dụ: `document.createElement("FORM")`

**Tập hợp các đối tượng của form (form object collections):** đó là các phần tử (elements) tham gia vào form, chúng xuất hiện theo thứ tự trình bày trong trang HTML. Để truy xuất vào từng thành phần sẽ phải thực hiện thông qua chỉ số (index), việc truy xuất được thực hiện theo cú pháp: **formObject.elements**. Để lấy số lượng thành phần thuộc form có thể sử dụng thuộc tính *length*.

**Các phương thức:** gồm 2 phương thức chính là reset và submit, các phương thức này thực hiện tương tự như nút ấn reset và submit của thẻ `<form>`

**Các thuộc tính:** Ngoài các phương thức kể trên đối tượng form còn các thuộc tính là thuộc tính của thẻ `<form>` bao gồm:

- **acceptCharset:** Thiết lập hoặc trả về giá trị của thuộc tính accept – charset của thẻ `<form>`
- **action:** Thiết lập hoặc trả về giá trị action của thẻ `<form>`
- **autocomplete:** Thiết lập hoặc trả về giá trị autocomplete của thẻ `<form>`
- **encoding:** thiết lập hoặc trả về giá trị encoding của thẻ `<form>`
- **enctype:** thiết lập hoặc trả về giá trị enctype của thẻ `<form>`
- **method:** thiết lập hoặc trả về giá trị method của thẻ `<form>`
- **name:** thiết lập hoặc trả về giá trị name của thẻ `<form>`
- **noValidate:** Thiết lập hoặc trả về dữ liệu mà không cần kiểm soát trong quá trình gửi đi.
- **target:** Thiết lập hoặc trả về giá trị target của thẻ `<form>`

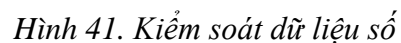
Ngoài các thuộc tính trên thì đối tượng form còn rất nhiều thuộc tính khác được sử dụng từ các thuộc tính của thẻ `<form>` và các thành phần trên form. Các thành phần này bao gồm thẻ `<input>`, `<select>` và `<textarea>`.

Trong javascript, đối tượng form khi được sử dụng thường để kiểm soát dữ liệu trước khi gửi đi. Nếu một trường trên form bị trống, hàm kiểm soát sẽ thông báo và trả về kết quả sai nhằm ngăn chặn việc gửi các dữ liệu sai đi. Javascript có thể kiểm soát việc nhập số vào hộp text.

*Ví dụ:*

```
<HTML><Head>
<TITLE> Ví dụ về kiểm soát dữ liệu trên form </TITLE>
</head>
<body>
<h3>JavaScript có thể kiểm soát form</h3>
<p>Hãy nhập số vào hộp text (từ 1 đến 10):</p>
<input id="numb">
<button                                type="button"
onclick="myFunction()">Submit</button>
<p id="demo"></p>
<script>
function myFunction() {
```

```
document.getElementById("demo")
).innerHTML = text;
}
</script>
</body>
</HTML>
```



Kiểm soát dữ liệu là quá trình đảm bảo rằng dữ liệu do người dùng nhập vào chính xác, hợp lý và hữu ích. Nhiệm vụ kiểm soát dữ liệu chính bao gồm điền đầy đủ thông tin vào tất cả các trường, điền đúng định dạng ngày tháng, điền đúng kiểu dữ liệu số. Kiểm soát việc nhập liệu có thể được thực hiện bằng nhiều cách khác nhau. Nếu kiểm soát dữ liệu được thực hiện phía máy chủ web thì dữ liệu sẽ được kiểm soát sau khi dữ liệu đã gửi tới máy chủ. Ngược lại, nếu việc kiểm soát được thực hiện phía máy trạm dữ liệu sẽ được kiểm soát trước khi dữ liệu được gửi tới máy chủ.

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

## Các thuộc tính

- 76

- LOG2E                      - logarit cơ số 2 của e, khoảng 1,442.
- PI                            - Giá trị của  $\pi$ , khoảng 3,14159.
- SQRT1\_2                   - Căn bậc 2 của 0,5, khoảng 0,707.
- SQRT2                      - Căn bậc 2 của 2, khoảng 1,414.

### Các phương thức

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

#### 4.8.8. Đối tượng Date

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

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

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

### Các phương thức

- *dateVar*.getDate() - Trả lại ngày trong tháng (1-31) cho *dateVar*.
- *dateVar*.getDay() - Trả lại ngày trong tuần (0=chủ nhật,...6=thứ bảy) cho *dateVar*.
- *dateVar*.getHours() - Trả lại giờ (0-23) cho *dateVar*.
- *dateVar*.getMinutes() - Trả lại phút (0-59) cho *dateVar*.
- *dateVar*.getSeconds() - Trả lại giây (0-59) cho *dateVar*.

- *dateVar.getTime()* - Trả lại số lượng các mili giây từ 00:00:00 ngày 1/1/1970.
- *dateVar.getTimeZoneOffset()* - Trả lại độ dịch chuyển bằng phút của giờ địa phương hiện tại so với giờ quốc tế GMT.
- *dateVar.getYear()* - Trả lại năm cho *dateVar*.
- *Date.parse(dateStr)* - Phân tích chuỗi *dateStr* và trả lại số lượng các mili giây tính từ 00:00:00 ngày 01/01/1970.
- *dateVar.setDay(day)* - Đặt ngày trong tháng là *day* cho *dateVar*.
- *dateVar.setHours(hours)* - Đặt giờ là *hours* cho *dateVar*.
- *dateVar.setMinutes(minutes)* - Đặt phút là *minutes* cho *dateVar*.
- *dateVar.setMonths(months)* - Đặt tháng là *months* cho *dateVar*.
- *dateVar.setSeconds(seconds)* - Đặt giây là *seconds* cho *dateVar*.
- *dateVar.setTime(value)* - Đặt thời gian là *value*, trong đó *value* biểu diễn số lượng mili giây từ 00:00:00 ngày 01/01/1970.
- *dateVar.setYear(years)* - Đặt năm là *years* cho *dateVar*.
- *dateVar.toGMTString()* - Trả lại chuỗi biểu diễn *dateVar* dưới dạng GMT.
- *dateVar.toLocaleString()* - Trả lại chuỗi biểu diễn *dateVar* theo khu vực thời gian hiện thời.
- *Date.UTC(year, month, day [,hours] [,minutes] [,seconds])* - Trả lại số lượng mili giây từ 00:00:00 01/01/1970 GMT.

#### 4.8.9. Đối tượng String

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

##### Các phương thức

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

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

**TỔNG KẾT CHƯƠNG 4:** Chương này thể hiện đầy đủ về ngôn ngữ JavaScript để tạo ra các kịch bản cho trang Web.

#### **BÀI TẬP:**

1. Hãy tạo ra một biểu mẫu bằng ngôn ngữ HTML để đăng ký thành viên gồm các trường: Họ tên, ngày tháng năm sinh, giới tính, địa chỉ, số điện thoại, e-mail, tên đăng nhập, mật khẩu, khẳng định mật khẩu. Sau đó sử dụng ngôn ngữ JavaScript để kiểm soát dữ liệu của người dùng nhập vào biểu mẫu trên.
2. Sử dụng ngôn ngữ Javascript để thực hiện tính toán giá trị của biểu thức  $ax^2+bx+c=0$  với a,b,c được nhập từ bàn phím trên giao diện của biểu mẫu (form) gồm 3 hộp text a,b,c. Kết quả của biểu thức  $x_1, x_2$  được trả về khi người dùng nhấn vào nút Tính.

## CHƯƠNG 5: NGÔN NGỮ HTML5 VÀ CSS3

### 5.1. Những khái niệm đầu tiên về HTML5

#### 5.1.1. Định nghĩa về HTML5

**HTML5** là một ngôn ngữ cấu trúc và trình bày nội dung cho **World Wide Web** và sẽ là công nghệ cốt lõi của Internet trong tương lai không xa, được đề xuất đầu tiên bởi Opera Software. Đây là phiên bản thứ 5 của ngôn ngữ HTML và hiện tại vẫn đang được phát triển bởi World Wide Web Consortium và WHATWG. Mục tiêu cốt lõi khi thiết kế ngôn ngữ là cải thiện khả năng hỗ trợ cho đa phương tiện mới nhất trong khi vẫn giữ được việc con người và các thiết bị, các chương trình máy tính như trình duyệt web, trình đọc màn hình, v.v.. có thể đọc, hiểu, hay xử lý một cách dễ dàng. HTML5 vẫn sẽ giữ lại những đặc điểm cơ bản của HTML4 và bổ sung thêm các đặc tả nổi trội của XHTML, DOM, đặc biệt là JavaScript.

Là phiên bản tiếp sau của HTML 4.01 và XHTML 1.1, HTML5 là một phản ứng để đáp lại lời phê bình rằng HTML và XHTML được sử dụng phổ biến trên World Wide Web là một hỗn hợp các tính năng với các thông số kỹ thuật khác nhau, được giới thiệu bởi nhiều nhà sản xuất phần mềm ví dụ Adobe, Sun Microsystems, Mozilla, Apple, Google,... và có nhiều lỗi cú pháp trong các văn bản web. Đây là một nỗ lực để tạo nên một ngôn ngữ đánh dấu có thể được viết bằng cú pháp HTML hoặc XHTML. Nó bao gồm các mô hình xử lý chi tiết để tăng tính tương thích, mở rộng, cải thiện và hợp lý hóa các đánh dấu có sẵn cho tài liệu, đưa ra các đánh dấu mới và giới thiệu giao diện lập trình ứng dụng (application programming interfaces API) để tạo ra các ứng dụng Web phức tạp. Cùng một lý do như vậy, HTML5 là một ứng cử viên tiềm năng cho nền tảng ứng dụng di động. Nhiều tính năng của HTML5 được xây dựng với việc xem xét chúng có thể sử dụng được trên các thiết bị di động như điện thoại thông minh và máy tính bảng hay không.

#### 5.1.2. Cú pháp của HTML5

HTML5 được phát triển dựa trên HTML4 và (X) HTML nên về cơ bản, nó có cấu trúc và cách khai báo không khác nhiều so với 2 phiên bản đó.

Cú pháp khai báo HTML5 rất mở:

- Không phân biệt kiểu chữ in hoa, in thường:  
`<H1> Đoạn tiêu đề 1</h1>`
- Các phần tử không bắt buộc phải có thẻ đóng  
`<p> Đoạn văn bản cho phần nội dung`
- Không bắt buộc phải có dấu nháy kép cho thuộc tính tuy nhiên đối với thuộc tính có dấu cách thì cần phải để trong dấu nháy kép  
`<img src=imageone.jpg alt=landscape>`

#### 5.1.3. Các thành phần mới của HTML5

- Các phần tử dạng khối (block elements)
  - **<header>**, **<footer>**: Bạn sẽ không còn cần thiết phải tạo các ID cho phần header và footer vì đã có các thẻ được định nghĩa trước này.
  - **<article>**: Đánh dấu một bài viết, một comment hoặc một thông báo.



- **<aside>**: Đánh dấu nội dung ngoài lề của một trang web, ví dụ như một sidebar.
- **<nav>**: Thanh điều hướng hoặc menu giờ có thể được đặt trong thẻ này, nó sẽ tự động tạo ra cho bạn một danh sách trông giống như một thanh điều hướng thực sự.
- **<section>**: Với cặp thẻ này, bạn có thể định nghĩa bất kỳ phân vùng nào trên trang web của mình.
- Các phần tử nội tuyến (inline elements): **<mark>**, **<time>**, **<meter>**, **<progress>**
- Các kiểu input trong form: **datetime**, **datetime-local**, **date**, **month**, **week**, **time**, **number**, **range**, **email**, **url**
- **<canvas>**: Kết hợp với Javascript, tạo một vùng có thể vẽ đồ họa.
- **<video>**, **<audio>**: Đính kèm video hoặc âm thanh.

#### 5.1.4. HTML5 API (giao diện lập trình ứng dụng) và công nghệ hỗ trợ

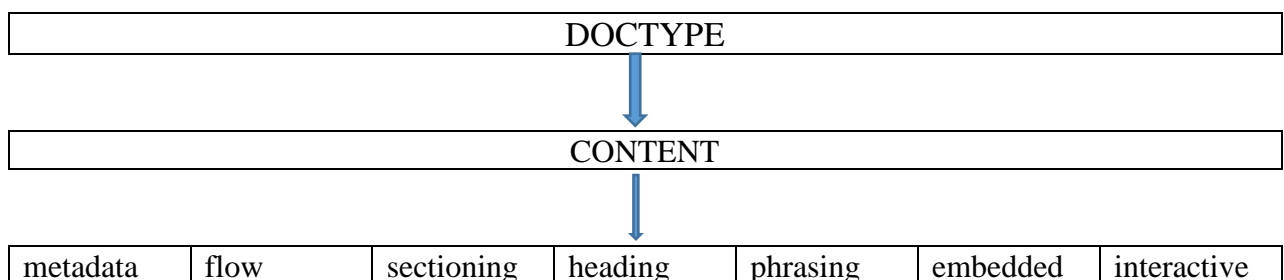
**API**: (Application programming interface – Giao diện lập trình ứng dụng): là tạo ra các ứng dụng sử dụng các thành phần được dựng sẵn không chỉ được phát triển trong ứng dụng web mà còn cả với các ngôn ngữ kịch bản. Mục đích chính của API là để chuẩn hóa cơ chế làm việc và đơn giản hóa các nhiệm vụ lập trình phức tạp. Một số HTML5 API như Drag and Drop, Web storage, Microdata và Geolocation.

- API Geolocation: Giúp xác định vị trí địa lý của trình duyệt web. Thông tin này được sử dụng để gửi dưới dạng dữ liệu liên quan dựa trên vị trí. Geolocation hiện tại đang được kích hoạt trong một số trình duyệt hiện đại.
- Web workers: là một framework giải quyết vấn đề hiệu suất của trình duyệt, là mã kịch bản chạy trên một luồng riêng biệt.
- Web storage: Nhằm cải tiến cookie của trình duyệt. Cookie là một công nghệ bị giới hạn và khó khăn cho các nhà thiết kế để có thể sử dụng. Web storage nâng cấp mô hình này để cung cấp không gian lưu trữ lớn hơn cho các ứng dụng web hiện đại.

## 5.2. Khởi tạo, làm việc với mã HTML5 và thành phần Form

### 5.2.1. Sử dụng ngôn ngữ đánh dấu HTML5

Cấu trúc file mã HTML5



- **Khai báo DOCTYPE:**

Được sử dụng để kiểm tra hợp lệ các tài liệu. Đối với các phiên bản HTML cũ hơn, phần DOCTYPE này yêu cầu được khai báo khá dài. Bởi lẽ HTML được dựa trên SGML (Standard Generalized Markup Language) và vì thế nên chúng cần một tham chiếu đến DTD.

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
```

```
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
```

Còn đối với HTML5, bạn chỉ cần khai báo DOCTYPE đơn giản như sau.

```
<!DOCTYPE html>
```

- Vùng mã nội dung (content) của HTML5 bao gồm:

Vùng nội dung metadata: là phần nội dung thiết lập cách trình bày hoặc hành vi của các nội dung còn lại trên trang. Có thể sử dụng nội dung metadata để thiết lập quan hệ giữa các tài liệu HTML, thường chứa các từ khóa hoặc mô tả cho trang web, và được các bộ máy tìm kiếm sử dụng để phân loại trang web, được đặt trong phần <head>.

Để sử dụng các ngôn ngữ khác ngoài tiếng Anh bên trong trang web của mình, chúng ta cần dùng đến việc mã hóa ký tự. Đối với việc sử dụng tiếng Việt, chúng ta sẽ dùng hệ mã hóa utf-8.

```
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
```

Còn đối với HTML5, bạn chỉ cần khai báo đơn giản như sau.

```
<meta charset="UTF-8">
```

- Khai báo thẻ script

Chèn các đoạn mã javascript hoặc vbscript bên ngoài hoặc ngay trong tài liệu.

```
<script type="text/javascript" src="file.js"></script>
```

Còn đối với HTML5, bạn chỉ cần khai báo đơn giản như sau.

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

- Khai báo thẻ link

Chèn các liên kết ngoài(chủ yếu là các tệp tin CSS).

```
<link rel="stylesheet" type="text/css" href="file.css">
```

Còn đối với HTML5, bạn chỉ cần khai báo đơn giản như sau.

```
<link rel="stylesheet" href="file.css">
```

- Vùng Flow: Đại diện cho các phần tử được coi là nội dung của trang web như các thẻ đánh dấu nội dung những phần tử được thêm mới trong HTML5 đều thuộc vùng này.
- Vùng Sectioning: Là vùng nội dung mới của HTML5. Bao gồm bốn phần tử <article>, <aside>, <nav> và <section>.
- Vùng Heading: Chứa tất cả các phần tử tiêu đề tiêu chuẩn hiện đang được sử dụng trong HTML 4.0 như <h1>,<h2>.....Với HTML5 bổ sung thêm phần tử <hgroup>.
- Vùng phrasing: Là văn bản của tài liệu bao gồm các phần tử đánh dấu văn bản bên trong một đoạn, là tập con của vùng flow.
- Vùng embedded: Là nội dung nhập một tài nguyên khác như hình ảnh hay video vào trong tài liệu.
- Vùng interactive: Là những phần tử được sử dụng để tương tác với người dùng.

### 5.2.2. Làm việc với các phần tử nội dung(content) của HTML5

HTML5 đã tạo ra những thẻ có ý nghĩa mới để quy định từng phần khác nhau trong trang web bao gồm: <aside>, <article>, <footer>, <header>, <nav>, <section>.

**Phần tử <header>**: Thẻ <header> được sử dụng cho phần đầu trang hoặc phần đầu của một thẻ. Thẻ <header> nên được sử dụng để bao ngoài nội dung giới thiệu trang. Bạn có thể sử dụng một vài thẻ <header> trong một trang HTML.

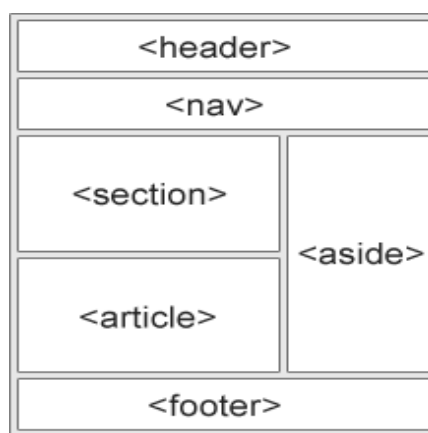
**Phần tử <section>**: Biểu diễn một vùng chung của tài liệu hoặc ứng dụng. nên sử dụng section khi muốn phân chia nội dung quan trọng như văn bản hoặc hình ảnh thành các vùng.

**Phần tử <nav>**: Thường sử dụng để chứa các thành phần điều hướng cho trang web.

**Phần tử <article>**: được sử dụng cho các nội dung khép kín, độc lập. Nội dung trong thẻ <article> có ý nghĩa riêng biệt và có thể độc lập với những phần còn lại của website. Người dùng có thể đọc nội dung bên trong thẻ <article> mà không cần quan tâm tới những phần khác. Bạn có thể sử dụng thẻ <article> trong: bài viết của diễn đàn, bài viết của blog, bBài viết của một trang báo.

**Phần tử <aside>**: được sử dụng để chứa những thông tin bên cạnh nội dung chính. Nội dung bên trong thẻ <aside> nên liên quan tới nội dung chính.

**Phần tử <footer>**: được sử dụng cho phần cuối trang hoặc hoặc phần cuối của một thẻ. Thông thường thẻ <footer> dùng để chứa thông tin về tác giả, thông tin bản quyền, liên kết tới điều khoản sử dụng, thông tin liên hệ, v.vv. Bạn có thể sử dụng một vài thẻ <footer> trong một trang HTML.

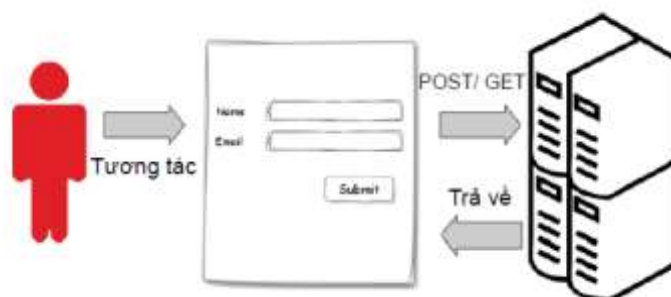


Hình 39. Thành phần HTML5

### 5.2.3. Tổng quan về các thành phần form mới của HTML5

Các thành phần mới của form trong HTML5 bổ sung thêm chức năng mà các nhà thiết kế cũng như các nhà phát triển web thường phải kết hợp thông qua các phương tiện khác như javascript và flash...

Cách làm việc của form:



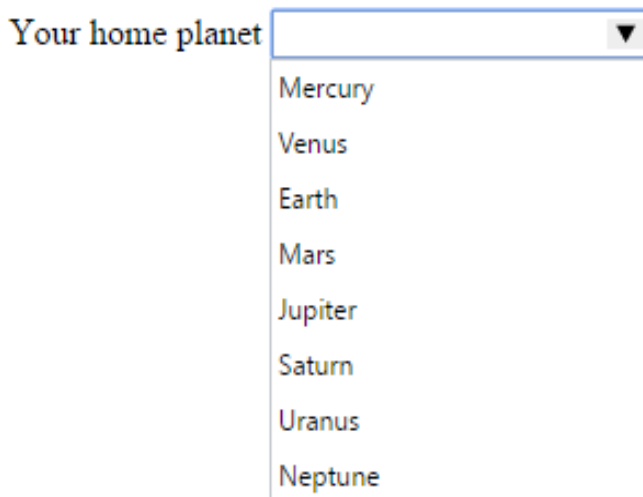
Hình 42. Cách làm việc của form

#### 5.2.4. Làm việc với thành phần form mới trong HTML5

- **Thành phần <datalist> trong HTML5**

Thẻ <datalist> được sử dụng để tạo các lựa chọn gợi ý cho thẻ <input>. Người dùng sẽ nhìn thấy một danh sách thả xuống trước khi bắt đầu nhập dữ liệu. Thuộc tính list trong thẻ <input> phải trùng với thuộc tính id trong thẻ <datalist>.

```
<label for="homeworld"> Your home planet</label>
<input type="text" name="homeworld" id="homeworld"
list="planets">
<datalist id="planets">
  <option value="Mercury">
  <option value="Venus">
  <option value="Earth">
  <option value="Mars">
  <option value="Jupiter">
  <option value="Saturn">
  <option value="Uranus">
  <option value="Neptune">
</datalist>
```



Hình 43. Thành phần datalist

- **Thành phần <input> trong HTML5**

Thành phần input trong HTML4 sử dụng thuộc tính *type* để chỉ định kiểu dữ liệu. Tương tự như vậy, trong HTML5, ngoài các kiểu dữ liệu đã được cung cấp sẵn từ HTML4, chúng ta còn được hỗ trợ thêm một số kiểu dữ liệu mới. Dưới đây là các kiểu dữ liệu mà HTML5 cung cấp:

Kiểu dữ liệu	Mô tả
datetime	ngày, tháng, năm, thời gian được mã hóa bằng ISO 8601 với time zone khởi tạo theo UTC.
datetime-local	tương tự như datetime nhưng không có timezone.
date	một ngày(bao gồm năm,tháng,ngày)
month	một tháng(bao gồm năm,tháng)

Kiểu dữ liệu	Mô tả
week	một tuần(bao gồm năm,số tuần)
time	khoảng thời gian(bao gồm giờ,phút,giây,tíc tắc)
number	chỉ chấp nhận giá trị nhập vào là số.
email	chỉ chấp nhận email.
url	chỉ chấp nhận đường dẫn url.

HTML5 thêm một số thuộc tính mới cho thẻ <input> trong HTML5:

- *Thuộc tính autocomplete:*

Thuộc tính autocomplete quy định một form hoặc một thẻ input trong form có được sử dụng chức năng tự động điền, gợi ý thông tin hay không. Khi thuộc tính autocomplete có giá trị “on”, trình duyệt sẽ tự động gợi ý những giá trị dựa trên thông tin người dùng nhập vào.

```
<form action="/selfdestruct" autocomplete="off">
<input type="text" name="onetime token" autocomplete="on">
```

- *Thuộc tính autofocus*

```
<input type="text" name="myvalue" autofocus/>
```

Thay vì như trước đây chúng ta phải sử dụng các script như js hoặc vbscript để trở tới một phần tử trong form thì nay đã có thuộc tính autofocus.

- *Thuộc tính min và max*

Thuộc tính min và max quy định giá trị tối thiểu và tối đa mà người dùng có thể nhập vào thẻ <input>. Thuộc tính min, max được sử dụng trong thẻ input với các type có giá trị: number, range, date, datetime, datetime-local, month, time và week.

```
<input type="number" name="quantity" min="1" max="5">
```

- *Thuộc tính placeholder*

```
<input type="text" name="myvalue" placeholder="đoạn
chữ này sẽ hiển thị bên trong thẻ input"/>
```

Thuộc tính này có thể được sử dụng với thẻ <input> hoặc <textarea>. Nó cho phép thẻ chứa một đoạn text ẩn bên trong mà khi người dùng nhập vào một đoạn text bất kỳ, đoạn text ẩn kia sẽ biến mất.

- *Thuộc tính required*

```
<input type="text" name="search" required/>
```

Đúng như tên gọi, thuộc tính này sẽ không cho phép người dùng submit một form khi mà thẻ input chứa nó chưa được điền thông tin.

- *Thuộc tính step*

Thuộc tính step quy định dữ liệu nhập vào phải là bộ số của một số. Ví dụ: nếu step="3", số hợp lệ được nhập vào là -3, 0, 3, 6, ... Thuộc tính step được sử dụng trong thẻ input với các type có giá trị: number, range, date, datetime, datetime-local, month, time và week.

- **Thành phần <output> trong HTML5**

Thẻ này sử dụng tương tự như thẻ <input>. Tuy nhiên thay vì nhập giá trị vào thì nó được sử dụng để hiển thị ra kết quả (như khi thực thi một đoạn script).

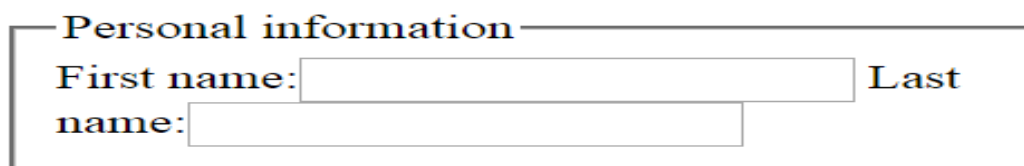
- **Thành phần <label> trong HTML5**

Là thành phần không bắt buộc tăng khả năng truy cập cho form

```
<label for="firstname"> First name:<input type="text" name="firstname" >
</label>
```

- **Thành phần <fieldset> trong HTML5:** Sử dụng để nhóm các thành phần liên quan của form vào một khung thể hiện. Nếu muốn thể hiện thông tin của khung cần sử dụng thẻ <legend>

```
<fieldset>
    <legend> Personal information </legend>
    <label>          First          name:<input          type="text"
    name="firstname" > </label>
    <label> Last name:<input type="text" name="lastname"
> </label>
</fieldset>
```



Hình 44. Thành phần fieldset

### 5.3. Làm việc với Javascript và query của HTML5

- **Làm việc với JAVASCRIPT:** cách sử dụng javascript đã được trình bày trong chương trước. Ngoài ra, Javascript còn cung cấp cho người dùng các framework với mục đích:
  - Là giải pháp tốt cho người thiết kế
  - Cung cấp một số thư viện có sẵn
  - Bao gồm nhiều hàm đã được xây dựng và kiểm tra bởi các nhà thiết kế và phát triển
  - Bao gồm nhiều hàm có sẵn và sử dụng được ngay
- **JQUERY**
  - jQuery là thư viện mới của javascript, dễ dàng tiếp cận đối với người thiết kế.
  - Thư viện jQuery có các tính năng sau:
    - Thao tác với HTML/DOM
    - Thao tác với CSS
    - Các phương thức, sự kiện HTML
    - Hiệu ứng và hoạt hình
    - AJAX
    - Các tiện ích (Utilities)
  - Khai báo jQuery:

```
<head>
<script src="jquery-3.1.1.min.js"></script>
</head>
```

Ví dụ: Ứng dụng jQuery để ẩn thành phần trên trang

```

<head>
  <script type="text/javascript"
src="jquery.js"></script>
  <script type="text/javascript">
    $(document).ready(function() {
      $("p").click(function() {
        $(this).hide();
      });
    });
  </script>
</head>
<body>
  <p>If you click on me, I will disappear.</p>
  <p>Click me away!</p>
  <p>Click me too!</p>
</body>

```



Hình 45. Ví dụ về jquery

Chú ý: Để sử dụng cần tải thư viện này về với hai phiên bản Product Version và Development Version. Cả hai phiên bản có thể tải tại địa chỉ website: <http://jquery.com/>

- Cú pháp của jQuery:
    - Chọn phần tử HTML để truy vấn
    - Thực hiện các “actions” tới các phần tử đó
- \$(selector).action()
- \$: xác định jQuery
- (selector): truy vấn tới thành phần HTML
- action: thể hiện hành động trên thành phần được chọn

Ví dụ:

\$(this).hide()	Thực hiện jQuery () ẩn, ẩn các yếu tố hiện HTML.
\$("#test").hide()	Thực hiện jQuery () ẩn, ẩn yếu tố có id= test
\$("p").hide()	Thực hiện jQuery () ẩn, ẩn tất cả các thành phần <p>
\$(".test").hide()	Thực hiện jQuery () ẩn, ẩn yếu tố có class= test



- jQuery selector:
  - Là thành phần quan trọng trong thư viện jQuery
  - Cho phép lựa chọn, thao tác tới các thành phần HTML như 1 nhóm hay yếu tố duy nhất
  - Cú pháp: `$()`

Jquery selectors	Giải nghĩa
<code>\$("*")</code>	Lựa chọn toàn bộ thành phần
<code>\$("p")</code>	Lựa chọn toàn bộ thành phần <code>&lt;p&gt;</code>
<code>\$("p.intro")</code>	Lựa chọn toàn bộ thành phần <code>&lt;p&gt;</code> có class là intro
<code>\$("p#intro")</code>	Lựa chọn thành phần <code>&lt;p&gt;</code> đầu tiên có id= intro
<code>\$(":animated")</code>	Lựa chọn toàn bộ thành phần hoạt hình
<code>\$(":button")</code>	Lựa chọn toàn bộ thành phần <code>&lt;input&gt;</code> có kiểu là "button"

- Sự kiện jQuery:
  - Các phương pháp xử lý sự kiện là chức năng cốt lõi của jQuery

`<head>`

```

<script type="text/javascript"
src=jquery.js"></script>

```

```

<script type="text/javascript">

```

```

    $(document).ready(function() {
        $("button").click(function() {
            $("p").hide();
        });
    });

```

```

    });
</script>

```

`</head>`

→ Sự kiện gọi 1 hàm được thực hiện khi có sự kiện nhấn chuột

- Một số sự kiện của jQuery:

Sự kiện	Giải nghĩa
<code>\$(document).ready(function)</code>	Liên kết tới hàm sự kiện (khi vừa load xong)
<code>\$(selector).click(function)</code>	Liên kết tới hàm gọi sự kiện nhấn chuột
<code>\$(selector).dblclick(function)</code>	Liên kết tới hàm gọi sự kiện nhấn đúp chuột
<code>\$(selector).focus(function)</code>	Liên kết tới hàm gọi sự kiện trọng tâm của thành phần được chọn
<code>\$(selector).mouseover(function)</code>	Liên kết tới hàm gọi sự kiện nhấn mouseover

- Hàm callback trong jQuery:
  - Được sử dụng để ngăn chặn các mã tiếp theo được chạy
  - Hàm có hiệu lực khi các hành động kết thúc
  - Cú pháp:



`$(selector).hide(speed,callback)`

*Ví dụ:*

```
$("#p").hide(1000,function(){
    alert("The paragraph is now hidden");
});
```

○ Thao tác với jQuery HTML:

- jQuery có phương pháp mạnh mẽ để thay đổi và thao tác với các phần tử HTML

```
<!DOCTYPE html>
```

```
<html>
```

```
<head>
```

```
    <script                                type="text/javascript"
src="jquery.js"></script>
```

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

```
        $(document).ready(function(){
```

```
            $("#button").click(function(){
```

```
                $("#p").html("W3Shools");
```

```
            });
```

```
        });
```

```
    </script>
```

```
</head>
```

```
<body>
```

```
    <h2>This is a heading</h2>
```

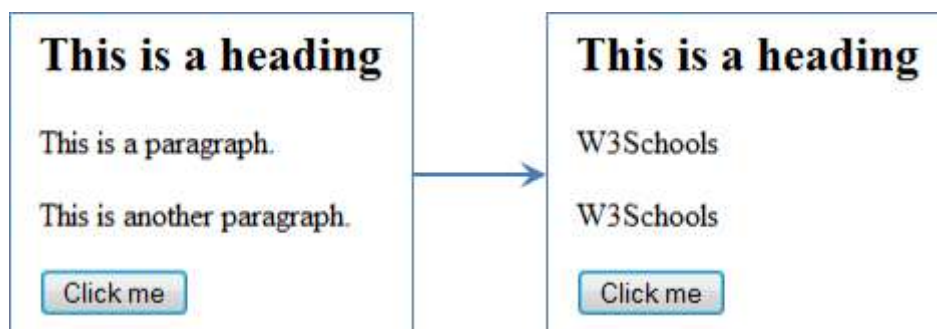
```
    <p>This is a paragraph.</p>
```

```
    <p>This is another paragraph.</p>
```

```
    <button>Click me</button>
```

```
</body>
```

```
</html>
```



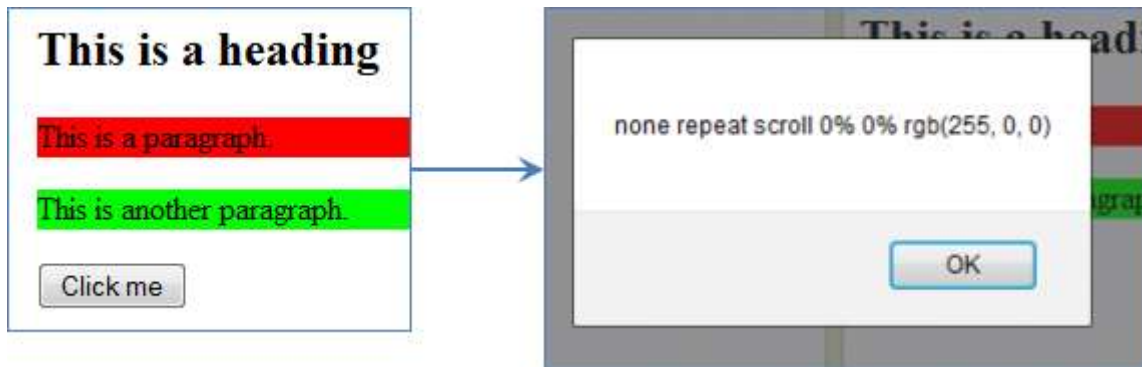
*Hình 46. Thao tác JQuery với HTML*

○ Chèn thêm nội dung HTML:

- `$("#p").append("W3School.");`
- `$("#p").prepend("W3School.");`
- `$("#p").after("W3School");`

- `$("p").before("W3School");`
- Thao tác với jQuery CSS:
  - Là phương thức quan trọng để thao tác với CSS
  - Bao gồm 3 cú pháp khác nhau, nhằm thực hiện các nhiệm vụ khác nhau
  - `css(property)`: trả về giá trị mặc định của CSS
 

```
$("p").css("background");
```



Hình 47. Thao tác JQuery với CSS với thuộc tính background

- `css(property, value)`: thiết lập giá trị và thuộc tính CSS
 

```
$("p").css("background", "yellow");
```
- `css({properties})`: thiết lập nhiều thuộc tính và giá trị CSS
 

```
$("p").css({"background": "yellow", "font-size": "200%"});
```

#### 5.4. Làm việc với các thành phần Video, Audio và Canvas của HTML5

Trước khi HTML5 ra đời, không có bất kỳ chuẩn nào quy định việc chạy audio trên website. Nếu muốn sử dụng audio trên trình duyệt, thông thường người dùng phải thêm chương trình hỗ trợ. Khi HTML5 ra đời đã cung cấp các thành phần HTML audio và HTML video chạy trong trình duyệt.

##### • Thành phần Audio trong HTML5:

Thẻ `<audio>` định nghĩa âm thanh, như nhạc hay trường audio khác. Thường dùng kèm với thẻ `<source>` để hiển thị được nhiều nội dung hơn.

Ví dụ:

```
<!DOCTYPE html>
<html>
<body>
  <audio controls>
    <source src="/nhac/song.ogg" type="audio/ogg">
    <source src="/nhac/song.mp3" type="audio/mpeg">
    Trình duyệt của bạn không hỗ trợ thẻ audio trong
    HTML5.
  </audio>
</body>
</html>
```

Thuộc tính **controls** cho phép hiển thị thanh điều khiển video như: bắt đầu, tạm dừng, âm lượng...

- **Thành phần Video trong HTML5:**

Thẻ `<video>` xác định một video, chẳng hạn như một đoạn phim hoặc một trường video. Đoạn text nằm bên trong `<video>` và `</video>` sẽ hiển thị khi trình duyệt không hỗ trợ thẻ `<video>`. Thường dùng kèm với thẻ `<source>` để hiển thị được nhiều nội dung hơn.

*Ví dụ:*

```
<!DOCTYPE html>
<html>
  <body>
    <video width="320" height="176" controls>
      <source src="/nhac/mov_bbb.mp4" type="video/mp4">
      <source src="/nhac/mov_bbb.ogg" type="video/ogg">
      Trình duyệt của bạn không hỗ trợ thẻ video trong HTML5
    .
  </video>
</body>
</html>
```

Thuộc tính **controls** cho phép hiển thị thanh điều khiển video như: bắt đầu, tạm dừng, âm lượng... Thuộc tính **height** và **width** được sử dụng để quy định chiều cao và độ rộng của video.

Nếu thuộc tính **height** và **width** không được thiết lập, trình duyệt sẽ không biết kích thước chính xác của video dẫn tới trang có thể bị co giãn khi tải video.

Đoạn thông báo nằm giữa thẻ `<video>` và `</video>` chỉ hiển thị đối với những trình duyệt không hỗ trợ thẻ `<video>`.

- **Thành phần Canvas trong HTML5:**

**Canvas** là một phần tử mới xuất hiện trong HTML5 và rất hữu ích trong việc vẽ đồ họa trên nền web. Nó lần đầu được giới thiệu bởi Apple, sau đó được ứng dụng nhiều vào trong trình duyệt Safari và bây giờ các trình duyệt phổ biến cũng đã hỗ trợ. Để canvas chạy được với IE ta cần thêm ExplorerCanvas. Thẻ `<canvas>` trong HTML5 dùng để vẽ 2D trên trang web. Vẽ một hình chữ nhật, hình chữ nhật gradient, hình chữ nhật nhiều màu và text nhiều màu trong canvas.

Canvas có thể giúp hiển thị trực quan một số hình ảnh dễ dàng trên trình duyệt như:

- Các sơ đồ đơn giản
- Trang trí thêm cho giao diện người dùng
- Hình ảnh động
- Biểu đồ và đồ thị
- Có thể nhúng các ứng dụng vẽ
- Hoạt động tốt với những hạn chế của CSS

Một canvas là một vùng hình vuông trong trang HTML và được xác định bằng thẻ `<canvas>`. Trạng thái mặc định của canvas là không đường viền và không nội dung.

Tag `<canvas>` sẽ có dạng:

```
<canvas id="myCanvas" width="200"
height="100"></canvas>
```

**Chú ý:** Luôn luôn xác định giá trị **id** (để JavaScript có thể tham chiếu tới) và **width, height** của canvas. Để thêm border, bạn có thể dùng CSS.

**Ví dụ:**

```
<canvas id="myCanvas" width="200" height="100"
style="border:1px solid #000000;"></canvas>
```

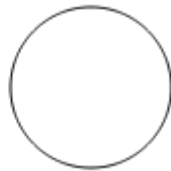
**Ví dụ về ứng dụng của Canvas**

Để vẽ một đường tròn trong Canvas, chúng ta dùng hàm sau.

`arc(x,y,r,start,stop)`

Để thực sự vẽ một đường tròn, chúng ta cũng phải sử dụng các hàm “mực” như là `stroke()` hay `fill()`.

Tạo một đường tròn với hàm **arc**



**JavaScript**

```
var
c=document.getElementById("myCanvas");
var ctx=c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
```

## 5.5. Làm việc CSS3

### 5.5.1. Tổng quan về CSS3

**CSS3** (Cascading Style Sheets Level 3) là phiên bản mới nhất của CSS. CSS3 không phải là một thành phần của HTMT5 nhưng lại có mối liên quan mật thiết với HTML5 và được phát triển song song với HTMT5.

Một số thành phần mới trong CSS3:

- CSS animation (CSS hoạt hình)
- CSS transition (CSS chuyển đổi)
- CSS 2D/#D transformation: transform
- CSS background, border, RGAA color, gradient, drop shadows, ....: border-radius, background-image, border-image
- Web font: @font-face.

Là tiêu chuẩn mới nhất của CSS nó hoàn toàn tương thích với các phiên bản trước. CSS3 được chia thành các module, các thành phần cũ được chia nhỏ và bổ sung các thành phần mới. Các module quan trọng trong css3 như Selectors, Box Model,

Backgrounds and Borders, Text Effects, 2D/3D Transformations, Animations, Multiple Column Layout, User Interface.

### 5.5.2. Thuộc tính mới *Border-radius* và *Border-image* trong CSS3

+ **Border-radius:** Tạo ra bốn góc bo tròn cho đường viền

**Cú pháp :**

```
tag {  
    border: bề-dày kiểu mã-màu;  
    border-radius: giá trị;  
    -moz-border-radius: giá trị;  
    -webkit-border-radius: giá trị;  
    -ms-border-radius: giá trị;  
    -o-border-radius: giá trị;  
}
```

Trong đó

-webkit-border-radius: hỗ trợ cho Google Chrome và Safari.  
-moz-border-radius: Giúp Firefox hỗ trợ.  
-ms-border-radius: hỗ trợ cho Internet Explorer.  
-o-border-radius: hỗ trợ cho Opera.

*Ví dụ:*

```
.box{  
border-radius:40px;  
-webkit-border-radius:40px;  
-moz-border-radius:40px;  
}
```



Hình 48. Thuộc tính *border-radius*

+ **Border-image:** Dùng để định dạng các dạng border bằng hình ảnh.

**Cú pháp :**

```
tag {  
    border-image: giá trị;  
    -moz-border-image: giá trị;  
    -webkit-border-image: giá trị;  
    -ms-border-image: giá trị;  
    -o-border-image: giá trị;  
}
```

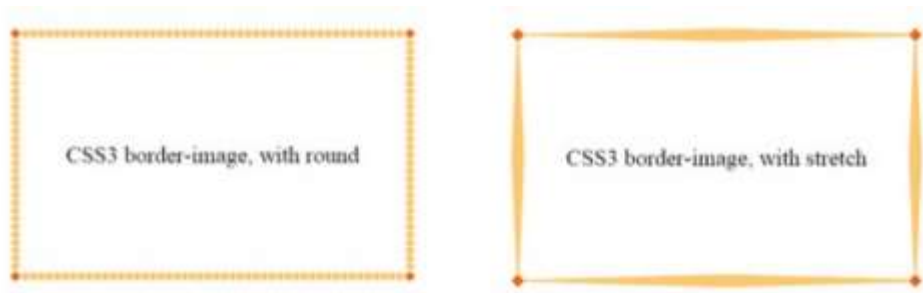
*Ví dụ:*

```
# borderimg{  
border-img:url(border.png) 30 round;  
-webkit-border- img: url(border.png) 30 round;  
-o-border- img: url(border.png) 30 round;
```

```

-moz-border-image: url(border.png) 30 round;
-ms-border-image: url(border.png) 30 round;
}

```



Hình 49. Ví dụ về thuộc tính border-radius

### 5.5.3. Transform, Transition và Animation

+ **Transform:** Thuộc tính transform xác định một chuyển đổi 2 chiều, 3 chiều, có thể là xoay, tỷ lệ, di chuyển, nghiêng, ...

**Cú pháp :**

```

tag {
    transform: giá trị;
    -moz- transform: giá trị;
    -webkit- transform: giá trị;
    -o- transform: giá trị;
}

```

Ví dụ:

```

p {
    background: #cc0000;
    height: 50px;
    width: 80px;
    transform: rotate(-15deg)
    -moz-transform: rotate(-15deg);
    -webkit-transform: rotate(-15deg);
    -o-transform: rotate(-15deg);
}

```



Hình 50. Thuộc tính transform

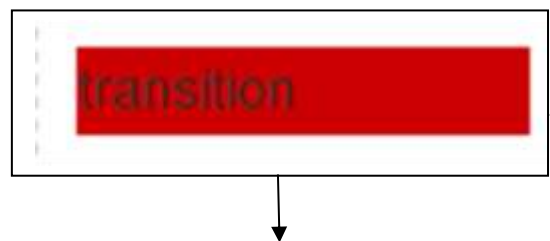
+ **Transition:** Thuộc tính transition xác định một quá trình chuyển đổi khi có một hành động.

**Cú pháp :**

```

tag {
    transition: giá trị;
    -moz- transition: giá trị;
    -webkit transition: giá trị;
    -o- transition: giá trị;
}

```



```
}
```

*Ví dụ:*

```
p {  
  background: #cc0000;  
  transition: height 2s;  
  -moz-transition: height 2s;  
  -webkit-transition: height 2s;  
  -o-transition: height 2s;  
  height: 23px;  
  width: 120px;  
}  
p:hover {  
  height: 100px;  
}
```

+ **Animation:** Thuộc tính animation xác định một chuyển động của một tag hay một hình ảnh, là sự tổng hợp của các thuộc tính: animation-name, animation-duration, animation-timing-function, animation-delay, animation-iteration-count, animation-direction.

**Cú pháp :**

```
tag {  
  animation: giá trị;  
  -moz-animation: giá trị;  
  -webkit-animation: giá trị;  
  -o-animation: giá trị;  
}
```

Trong đó:

- moz-animation hỗ trợ cho firefox.
- webkit-animation hỗ trợ cho Google Chrome và Safari.
- o-animation hỗ trợ cho Opera.

*Ví dụ:* Tạo liên kết “example” tới thẻ <div>. Các chuyển động sẽ kéo dài trong thời gian 4s, và nó sẽ thay đổi màu nền của phần tử <div> từ “red” thành “yellow”:

```
@keyframes example {  
  from {background-color: red;}  
  to {background-color: yellow;}  
}  
div {  
  width: 100px;  
  height: 100px;  
  background-color: red;  
  animation-name: example;
```

```

        animation-duration: 4s;
    }

```

#### 5.5.5. Font web

Thuộc tính font trong css3 dùng để định dạng các dạng font ngoài dạng đã được định nghĩa ở css2

**Cú pháp :**

```

@font-face {
    thuộc tính: giá trị;
}

```

*Ví dụ:*

```

@font-face {
    font-family: 'myFont';
    src: url('vcouri-webfont.eot');
    src: url('vcouri-webfont.eot?#iefix'), /* IE4+
*/
        url('vcouri-webfont.woff') format('woff'),
        url('vcouri-webfont.ttf') format('truetype'), /*
font chuan */
        url('vcouri-webfont.svg#vni-courinormal')
format('svg'); /* iphone, ipad*/
}
p.addFont {
    font-family: 'myFont';
}

```

### 5.6. Làm việc với thành phần mở rộng của CSS3

#### 5.6.1. CSS3 Media Queries

Với sự đa dạng của các thiết bị có chức năng duyệt web như ngày nay, các web-developer rất khó để website của mình hiển thị như ý muốn trên tất cả chúng. CSS3 ra đời đã làm cho điều đó dễ dàng hơn với Media Queries một chức năng giúp xác định và khoanh vùng chính xác thiết bị của người dùng. Kiểm tra khả năng của người dùng truy cập vào trang web. Nhận biết được chiều cao, chiều rộng của trình duyệt, độ phân giải.... Sử dụng CSS3 media queries để cung cấp các layout phù hợp với các layout mobile.

#### 5.6.2. CSS3 layout

CSS3 cung cấp các thuộc tính để thuận tiện cho việc thiết kế các layout dạng nhiều cột.

- Column-count: quy định cụ thể số lượng các cột một phần tử được chia thành.
- Column-width: quy định cụ thể chiều rộng của các cột
- Column-gap: quy định khoảng cách giữa các cột.



- Column-rule: là thuộc tính viết tắt, cho phép thiết lập tất cả các thuộc tính: chiều rộng, style, màu sắc giữa các cột.

### 5.6.3. CSS3 user interface(giao diện người dùng)

CSS3 cung cấp một số tính năng về phía người dùng: Thay đổi kích thước thành phần trên trang, thay đổi kích thước hộp.

- CSS3 resize: quy định một thành phần có thể hay không thay đổi kích thước
- CSS3 box sizing: Xác định lại chiều rộng và chiều cao cho thành phần.
- CSS3 outline-offset: Thuộc tính outline-offset property sẽ thêm khoảng trắng giữa outline và border của phần tử. Outlines khác với borders ở các điểm sau:
  - outline là đường vẽ xung quanh phần tử, bên ngoài border
  - outline sẽ không chiếm không gian khi hiển thị trên trình duyệt
  - outline có thể không là hình chữ nhật(non-rectangular)

**TỔNG KẾT CHƯƠNG 5:** Chương này thể hiện đầy đủ cách thức trình bày một trang HTML sử dụng công nghệ HTML5 kết hợp với việc sử dụng ngôn ngữ CSS3 nhằm giúp cho trang web thể hiện một cách tốt hơn.

### **BÀI TẬP:**

1. Hãy tạo ra một trang web kết hợp giữa HTML5 và CSS3 trong đó có thể hiện tương tác của người dùng khi di chuyển vào ảnh, bức ảnh được biến đổi.
2. Hãy sử dụng các thẻ HTML5 và CSS3 đã học để tạo ra một trang web được thể hiện trên mọi thiết bị (responsive) gồm có máy tính, máy tính bảng, điện thoại?

## TÀI LIỆU THAM KHẢO

- [1] Hướng dẫn thiết kế trang web tương tác bằng JavaScript  
Nguyễn Trường Sinh, Lê Minh Hoàng, Hoàng Đức Hải, Phạm Hoàng Dũng – NXB Giáo dục – 2001
- [2] Beginning HTML, XHTML, CSS and Javascript  
Jon Duckett – Wiley Publishing, Inc – 2010
- [3] HTML, XHTML and CSS – Fifth Edition  
Steven M. Schafer – Wiley Publishing, Inc – 2010
- [4] Beginning HTML5 and CSS3  
Christopher Murphy, Richard Clark, Oliver Studholme, Divya Manian – Paul Manning - 2012
- [5] HTML5 Và CSS3 - Thiết Kế Trang Web Thích Ứng Giàu Tính Năng  
Jermy Osborn & Nhóm AGI Creative – NXB Bách Khoa – 2015
- [6] Website: <http://www.w3schools.com/>