MỤC LỤC

LỜI NÓI ĐẦU

Môn học Thiết kế phát triển Website cung cấp cho sinh viên các kiến thức cơ bản về HTML/CSS, Javascript và một số kỹ thuật thiết kế web. Trong phần lý thuyết và thực hành của môn học trình bày về ngôn ngữ Javascript dùng để tương tác với các thành phần trên trang web, các kỹ thuật thiết kế website trên ngôn ngữ HTML, đồng thời hiểu và nắm được các kiến thức cơ bản về thiết kế bố cục trang web từ đơn giản đến phức tạp để thiết kế hoàn thiện trang website.

Để đạt được hiểu quả tốt cho môn học, sinh viên cần làm đầy đủ các bài thực hành trong cuốn bài tập thực hành do thầy cô hướng dẫn.

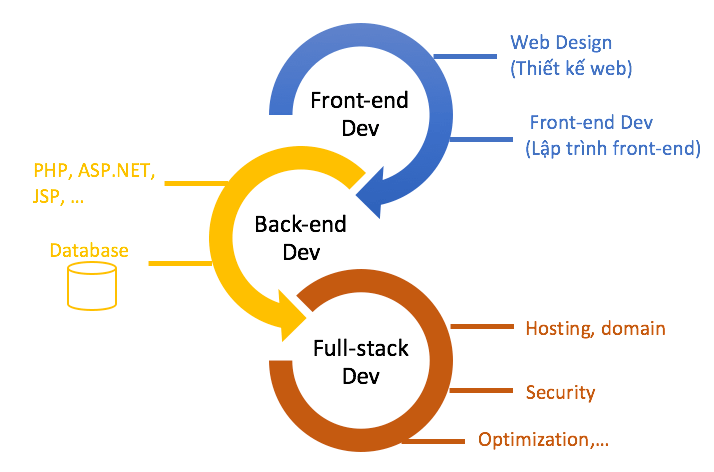
*Xin chân thành cảm ơn!*

# 

# CHƯƠNG I - HTML

## Tổng quan về một trang web

Trang web là một văn bản được sử dụng trên mạng toàn cầu (World Wide Web), văn bản này được viết sử dụng định dạng HTML hay XHTML. Các trang web liên kết với nhau bằng các siêu liên kết (Hyperlink) và được hiển thị thông qua một trình duyệt Browser (Chrome, FireFox …). Trang web có thể nằm ở máy tính cục bộ hoặc một máy tính ở xa (máy chủ) trên mạng internet và có thể truy cập đến nó thông qua một địa chỉ (URL). Trang web có thể hiển thị tất cả nội dung như văn bản, hình ảnh, âm thanh, video …



*Bức tranh tổng thể về ngành Web Development*

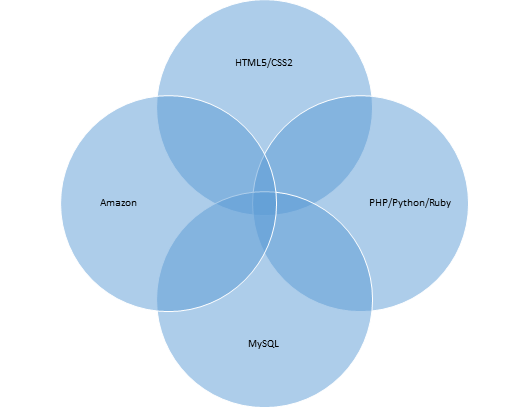
Phần front-end của một trang web là phần tương tác với người dùng. Tất cả mọi thứ mà người dùng nhìn thấy khi điều hướng trên internet, từ font chữ, màu sắc cho tới các menu xổ xuống và các thanh trượt, là một sự kết hợp của HTML, CSS và JavaScript được điều khiển bởi trình duyệt máy tính.

Các lập trình viên front-end chịu trách nhiệm cho giao diện của một trang web và kiến trúc những trải nghiệm cho người dung. Để thực hiện được những mục tiêu đó, các lập trình viên front-end phải sử dụng thành thạo 3 ngôn ngữ chính: HTML, CSS và ngôn ngữ lập trình JavaScript. Ngoài ra, các lập trình viên front-end cần phải làm quen với các framework như Bootstrap, Foundation, Backbone, Angular JS và EmberJS để đảm bảo nội dung luôn hiển thị tốt trên mọi thiết bị khác nhau, các thư viện như jQuery và LESS, đóng gói code vào trong một hình thức giúp tiết kiệm thời gian và hữu dụng hơn. Rất nhiều công việc dành cho lập trình viên front-end cũng yêu cầu kinh nghiệm với Ajax, một kỹ thuật được sử dụng rộng rãi bằng cách dung JavaScript để cho phép các trang load một cách tự động bằng cách tải dữ liệu máy chủ ở phần background.

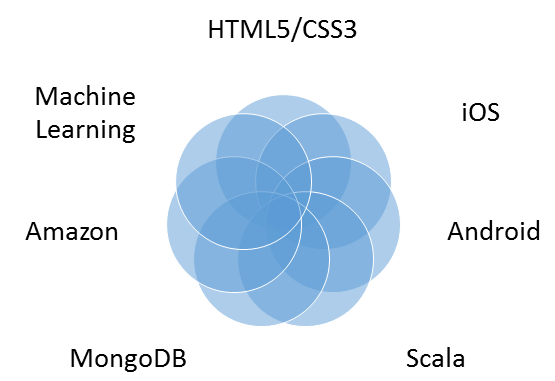
Thế nhưng điều gì giúp phần front-end của một trang web có thể hoạt động được? Tất cả dữ liệu sẽ được lưu trữ ở đâu? Đó là phần việc của back-end. Phần back-end của một trang web bao gồm một máy chủ, một ứng dụng và một cơ sở dữ liệu. Một lập trình viên back-end xây dựng và duy trì công nghệ mà sức mạnh của những thành phần đó, cho phép phần giao diện người dùng của trang web có thể tồn tại được.

Để khiến cho máy chủ, ứng dụng và cơ sở dữ liệu có thể giao tiếp được với nhau, các lập trình viên back-end sử dụng các ngôn ngữ server-side như PHP, Ruby, Python, Java và .Net để xây dựng một ứng dụng, và các công cụ như MySQL, Oracle và SQL Server để tìm kiếm, lưu trữ hoặc thay đổi dữ liệu và phục vụ trở lại tới người dung trong phần front-end. Các công việc tuyển dụng lập trình viên back-end cũng thường yêu cầu kinh nghiệm về các framework PHP như Zend, Symfony, CakePHP …; có kinh nghiệm với các phần mềm quản lý phiên bản như SVN, CVS hoặc Git; kinh nghiệm với Linux trong việc phát triển và triển khai hệ thống; kinh nghiệm với các hệ mã nguồn mở như Wordpress, Laravel…

Các lập trình viên full stack làm việc giống như các lập trình viên back-end ở phía máy chủ của lập trình web, nhưng họ cũng có thể thành thạo các ngôn ngữ front-end để điều khiển nội dung như thế nào ở phía giao diện của trang web. Để minh họa sự phức tạp ngày càng tang của phát triển full-stack, sau đây là một ví dụ về những yêu cầu mà một full-stack phải có trong vài năm về trước và so sánh với yêu cầu hiện nay.



*Yêu cầu kỹ năng cho một lập trình viên full-stack vài năm trước đây*



*Yêu cầu kỹ năng cho một lập trình viên full-stack hiện nay*

Bất kể là sử dụng công cụ xác định nào, tùy thuộc vào dự án và khách hang, các lập trình viên full-stack nên có kiến thức ở mọi cấp độ về cách web hoạt động: cài đặt và cấu hình các máy chủ Linux, viết các API server-side, nhảy vào phần JavaScript client-side của một ứng dụng và cũng cần có “con mắt thẩm mỹ” với CSS.

**1.2 Công cụ và môi trường**

Môn học này chúng ta sẽ tập trung nghiên cứu về lập trình front-end, như vậy để thiết kế được một trang web bắt mắt, hấp dẫn và có mã nguồn đạt chuẩn quốc tế, bố cục rõ ràng, giàu ngữ nghĩa (semantic). Đầu tiên để bắt mắt và hấp dẫn thì dựa vào Photoshop Web Designer (PSD Web Designer), đó là những người có óc về mỹ thuật, hiểu rõ cách phối màu, sắp xếp vị trí các hình ảnh, văn bản … sao cho lôi cuốn được thị giác người lướt web. Những PSD Web Designer phải sử dụng thành thạo công cụ Photoshop để hiện thực hóa ý tưởng giao diện web của mình. Và kết quả là 1 file PSD cụ thể để nhà thiết kế website (Web Designer) dựa vào đó để thiết kế y như mẫu.

Sau khi có bản thiết kế (file PSD) thì công việc còn lại sẽ dành cho nhà thiết kế website (Web Designer). Họ sẽ sử dụng HTML và CSS để xây dựng trang web dựa trên file PSD được cung cấp. Giao diện trang web phải giống như đúc giao diện trong PSD. Và để làm được điều đó thì Web Designer phải sử dụng công cụ Photoshop để trích xuất các đối tượng trong file PSD như: hình ảnh, font chữ, màu sắc,… và đo chính xác khoảng cách giữa các đối tượng này. Mỗi ngày website càng phát triển, nó cần được chuẩn hóa để làm sao các trình duyệt (browser) hiển thị cùng một kết quả và đặc biệt là những siêu website như Google, Bing,… có thể hiểu nội dung trang web đó đang nói về cái gì. Chính vì vậy, một trang web giàu tính ngữ nghĩa (semantic) đang được quan tâm hơn bao giờ hết.

Với web thì lập trình có 2 phần riêng biệt, đó là lập trình phía Client và lập trình phía Server. Công việc chính của lập trình front-end chủ yếu xử lý ở phía Client. Client chính là trình duyệt (browser): Chrome, Firefox, Safari, Opera,… Người lập trình front-end phải am hiểu rất rõ về thiết kế web, chính là 2 ngôn ngữ HTML và CSS. Ngoài ra còn phải sử dụng thành thạo ngôn ngữ lập trình web phía Client, mà hiện nay ngôn ngữ chính là JavaScript. Để tạo ra một trang web chúng ta cần một chương trình có thể chỉnh sửa được HTML gọi chung là text editor HTML như: Sublime Text, Notepad++, Dreamweaver,…

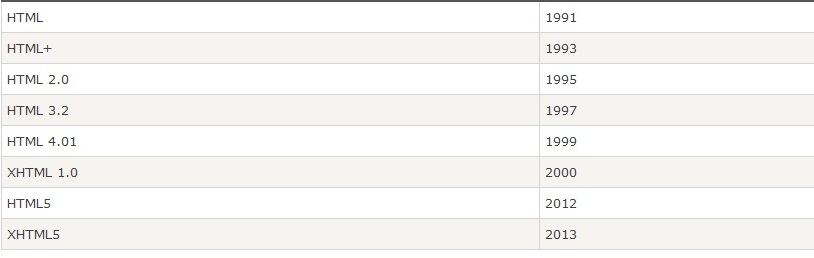
**1.3 Cấu trúc cơ bản của một file HTML**

HTML (tiếng Anh, viết tắt cho HyperText Markup Language, tức là "Ngôn ngữ Đánh dấu Siêu văn bản") là một ngôn ngữ đánh dấu được thiết kế ra để tạo nên các trang web, nghĩa là các mẩu thông tin được trình bày trên World Wide Web.

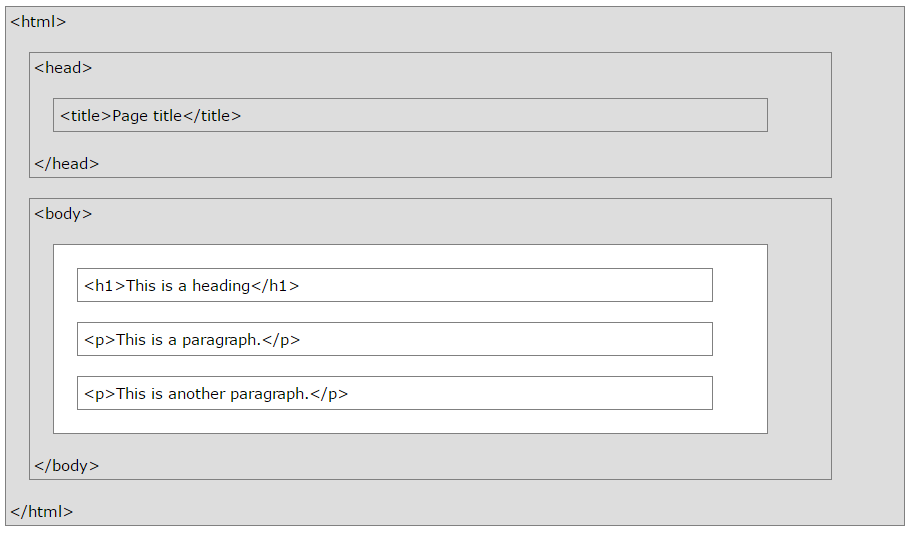
HTML không phải là một ngôn ngữ lập trình, do chúng chỉ có cấu trúc mã lệnh nhưng không có cấu trúc điều khiển.

Tuy HTML không phải là ngôn ngữ lập trình nhưng nó lại đóng vài trò rất quan trọng trong các ngôn ngữ viết web (PHP, ASP, JSP). Vì những gì chúng ta nhìn thầy (hình ảnh, kích thước, kiểu chữ, màu sắc) và những gì chúng ta tương tác (mẫu biểu, khung thoại,...) đều được xây dựng từ HTML.

Các phiên bản HTML và năm ra đời



Cấu trúc của một trang HTML



Các thành phần cơ bản

* Thẻ (tag) là một tập hợp ký hiệu được định nghĩa trong HTML có ý nghĩa đặc biệt, thẻ bắt đầu bởi “<” và kết thúc bởi “>”, thẻ mở có cú pháp <tên thẻ>, thẻ đóng có cú pháp </tên thẻ>
* Phần tử (element) có thể bao gồm thẻ mở, thẻ đóng và nội dung bên trong cặp thẻ mở, đóng. Cấu trúc của một phần tử:

<tên\_thẻ thuộc\_tính = “giá trị”> Nội dung </tên thẻ>

Ví dụ:

<font color = “red”> Chữ này có màu đỏ </font>

* Thuộc tính: mỗi thẻ có thể có một hay nhiều thuộc tính đi kèm, thuộc tính được nhập ngay trước ngoặc đóng “>” của thẻ mở, có thể có nhiều thuộc tính trong một thẻ, các thuộc tính phân cách nhau khoảng trắng. Nếu giá trị là chuỗi ký tự có khoảng trắng bên trong, bắt buộc phải dung “” hay ‘’ để bao chuỗi lại.

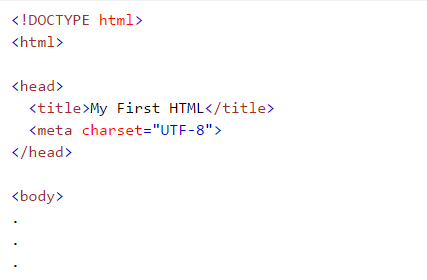
Một số điều cần lưu ý khi soạn thảo siêu văn bản bằng HTML:

* Nhiều dấu cách liền nhau cũng chỉ có tác dụng như một dấu cách. Phải sử dụng thẻ để thể hiện nhiều dấu giãn cách liền nhau.
* Gõ Enter để xuống dòng được xem như một dấu cách, để xuống hàng thì phải sử dụng thẻ tương ứng
* Có thể viết tên thẻ không phân biệt chữ in thường và in hoa.
* Bảng mã và tên gọi trong HTML của một số ký tự đặc biệt:

|  |  |
| --- | --- |
| **Mã ký tự** | **Ký tự** |
| &quot; | " |
| &amp; | & |
| &lt; | < |
| &gt; | > |
| &euro; | € |
| &nbsp; |  |
| &Aacute; | Á |
| &aacute; | á |
| &Acirc; | Â |
| &acirc; | â |
| &acute; | ´ |
| &AElig; | Æ |
| &aelig; | æ |
| &Agrave; | À |
| &agrave; | à |
| &Aring; | Å |
| &aring; | å |
| &Atilde; | Ã |
| &atilde; | ã |
| &Auml; | Ä |
| &auml; | ä |
| &brvbar; | ¦ |
| &Ccedil; | Ç |
| &ccedil; | ç |
| &cedil; | ¸ |
| &cent; | ¢ |
| &circ; | ˆ |
| &copy; | © |
| &curren; | ¤ |
| &deg; | ° |
| &divide; | ÷ |
| &Eacute; | É |
| &eacute; | é |
| &Ecirc; | Ê |
| &ecirc; | ê |
| &Egrave; | È |
| &egrave; | è |
| &ETH; | Ð |
| &eth; | ð |
| &Euml; | Ë |
| &euml; | ë |
| &euro; | € |
| &fnof; | ƒ |
| &frac12; | ½ |
| &frac14; | ¼ |
| &frac34; | ¾ |
| &Iacute; | Í |
| &iacute; | í |
| &Icirc; | Î |
| &icirc; | î |
| &iexcl; | ¡ |
| &Igrave; | Ì |
| &igrave; | ì |
| &iquest; | ¿ |
| &Iuml; | Ï |
| &iuml; | ï |
| &laquo; | « |
| &macr; | ¯ |
| &micro; | µ |
| &middot; | · |
| &not; | ¬ |
| &Ntilde; | Ñ |
| &ntilde; | ñ |
| &Oacute; | Ó |
| &oacute; | ó |
| &Ocirc; | Ô |
| &ocirc; | ô |
| &OElig; | Œ |
| &oelig; | œ |
| &Ograve; | Ò |
| &ograve; | ò |
| &ordf; | ª |
| &ordm; | º |
| &Oslash; | Ø |
| &oslash; | ø |
| &Otilde; | Õ |
| &otilde; | õ |
| &Ouml; | Ö |
| &ouml; | ö |
| &para; | ¶ |
| &plusmn; | ± |
| &pound; | £ |
| &raquo; | » |
| &reg; | ® |
| &Scaron; | Š |
| &scaron; | š |
| &sect; | § |
| &shy; |  |
| &sup1; | ¹ |
| &sup2; | ² |
| &sup3; | ³ |
| &szlig; | ß |
| &THORN; | Þ |
| &thorn; | þ |
| &tilde; | ˜ |
| &times; | × |
| &Uacute; | Ú |
| &uacute; | ú |
| &Ucirc; | Û |
| &ucirc; | û |
| &Ugrave; | Ù |
| &ugrave; | ù |
| &uml; | ¨ |
| &Uuml; | Ü |
| &uuml; | ü |
| &Yacute; | Ý |
| &yacute; | ý |
| &yen; | ¥ |
| &Yuml; | Ÿ |
| &yuml; | ÿ |
| &ensp; |  |
| &emsp; |  |
| &thinsp; |  |
| &zwnj; | ‌ |
| &zwj; | ‍ |
| &lrm; | ‎ |
| &rlm; | ‏ |
| &ndash; | – |
| &mdash; | — |
| &lsquo; | ‘ |
| &rsquo; | ’ |
| &sbquo; | ‚ |
| &ldquo; | “ |
| &rdquo; | ” |
| &bdquo; | „ |
| &lsaquo; | ‹ |
| &rsaquo; | › |
| &dagger; | † |
| &Dagger; | ‡ |
| &permil; | ‰ |
| &bull; | • |
| &hellip; | … |
| &Prime; | ″ |
| &prime; | ′ |
| &oline; | ‾ |
| &frasl; | ⁄ |
| &weierp; | ℘ |
| &image; | ℑ |
| &real; | ℜ |
| &trade; | ™ |
| &alefsym; | ℵ |
| &larr; | ← |
| &uarr; | ↑ |
| &rarr; | → |
| &darr; | ↓ |
| &harr; | ↔ |
| &crarr; | ↵ |
| &lArr; | ⇐ |
| &uArr; | ⇑ |
| &rArr; | ⇒ |
| &dArr; | ⇓ |
| &hArr; | ⇔ |
| &forall; | ∀ |
| &part; | ∂ |
| &exist; | ∃ |
| &empty; | ∅ |
| &nabla; | ∇ |
| &isin; | ∈ |
| &notin; | ∉ |
| &ni; | ∋ |
| &prod; | ∏ |
| &sum; | ∑ |
| &minus; | − |
| &lowast; | ∗ |
| &radic; | √ |
| &prop; | ∝ |
| &infin; | ∞ |
| &ang; | ∠ |
| &and; | ∧ |
| &or; | ∨ |
| &cap; | ∩ |
| &cup; | ∪ |
| &int; | ∫ |
| &there4; | ∴ |
| &sim; | ∼ |
| &cong; | ≅ |
| &asymp; | ≈ |
| &ne; | ≠ |
| &equiv; | ≡ |
| &le; | ≤ |
| &ge; | ≥ |
| &sub; | ⊂ |
| &sup; | ⊃ |
| &nsub; | ⊄ |
| &sube; | ⊆ |
| &supe; | ⊇ |
| &oplus; | ⊕ |
| &otimes; | ⊗ |
| &perp; | ⊥ |
| &hArr; | ⇔ |
| &sdot; | ⋅ |
| &lceil; | ⌈ |
| &rceil; | ⌉ |
| &lfloor; | ⌊ |
| &rfloor; | ⌋ |
| &lang; | ⟨ |
| &rang; | ⟩ |
| &loz; | ◊ |
| &spades; | ♠ |
| &clubs; | ♣ |
| &hearts; | ♥ |
| &diams; | ♦ |
| &Alpha; | Α |
| &alpha; | α |
| &Beta; | Β |
| &beta; | β |
| &Gamma; | Γ |
| &gamma; | γ |
| &Delta; | Δ |
| &delta; | δ |
| &Epsilon; | Ε |
| &epsilon; | ε |
| &Zeta; | Ζ |
| &zeta; | ζ |
| &Eta; | Η |
| &eta; | η |
| &Theta; | Θ |
| &theta; | θ |
| &thetasym; | ϑ |
| &Iota; | Ι |
| &iota; | ι |
| &Kappa; | Κ |
| &kappa; | κ |
| &Lambda; | Λ |
| &lambda; | λ |
| &Mu; | Μ |
| &mu; | μ |
| &Nu; | Ν |
| &nu; | ν |
| &Xi; | Ξ |
| &xi; | ξ |
| &Omicron; | Ο |
| &omicron; | ο |
| &Pi; | Π |
| &pi; | π |
| &piv; | ϖ |
| &Rho; | Ρ |
| &rho; | ρ |
| &Sigma; | Σ |
| &sigma; | σ |
| &sigmaf; | ς |
| &Tau; | Τ |
| &tau; | τ |
| &Upsilon; | Υ |
| &upsilon; | υ |
| &upsih; | ϒ |
| &Phi; | Φ |
| &phi; | φ |
| &Chi; | Χ |
| &chi; | χ |
| &Psi; | Ψ |
| &psi; | ψ |
| &Omega; | Ω |
| &omega; | ω |

**1.4 Các thẻ HTML cơ bản**

***Thẻ head*** chứa thông tin về website cho trình duyệt và các Search Engine (Google, Bing …). Thẻ title nằm trong cặp thẻ head nhằm tạo tiêu đề cho website



Thẻ body chứa toàn bộ nội dung của website, một số thuộc tính của thẻ body:

|  |  |  |
| --- | --- | --- |
| Thuộc tính | Ý nghĩa | Ví dụ |
| bgcolor=“color” | Thiết lập màu nền cho trang web | <body bgcolor=“#00FF00”> |
| background=“url” | Thiết lập ảnh nền cho trang web | <body background=“image.jpg”> |
| leftmargin=“num”  topmargin=“num” | Thiết lập lề trái và lề trên cho trang web | <body leftmargin=“0” topmargin=“0”> |
| text=“color” | Thiết lập màu chữ của văn bản, kể cả các đề mục | <body text=“#FF0000”> |
| alink=“color”  vlink=“color”  link=“color” | Xác định màu sắc cho các siêu liên kết trong văn bản | <body alink=“#FF0000” vlink=“#00FF00” link=“#0000FF”> |

***Các thẻ định dạng văn bản:***

|  |  |  |
| --- | --- | --- |
| Thẻ + Thuộc tính | Ý nghĩa | Ví dụ |
| <font face=“fontname”  size=“fontsize”  color=“color”> | Thiết lập font chữ cho văn bản | <font face=“Arial”  size=“5”  color=“#FF00FF”> |
| <p align=“?”></p>  “?” = “center” | “justify” | “left” | “right” | Định dạng một đoạn văn bản | <p align=“justify”>Đoạn văn bản này được canh đều hai biên trái phải</p> |
| <b></b> | Định dạng chữ in đậm | <b>**Đoạn văn bản in đậm<**/b> |
| <i></i>, <em></em> | Định dạng chữ in nghiêng | <i>*Đoạn văn bản in nghiêng<*/i> |
| <u></u> | Định dạng chữ gạch dưới | <u>Đoạn văn bản gạch dưới</u> |
| <strike></strike> | Định dạng chữ gạch ngang chữ | <strike>~~Đoạn văn bản bị gạch ngang chữ~~</strike> |
| <sup></sup> | Định dạng chỉ số trên | ax<sup>2</sup> → ax2 |
| <sub></sub> | Định dạng chỉ số dưới | H<sub>2</sub>O → H2O |
| <h1> , … , <h6> | Định dạng tiêu đề từ kích thước 1 đến 6 (tiêu đề 1 có kích thước lớn nhất) | <h1>Tiêu đề</h1> |
| <pre></pre> | Bỏ qua định dạng của các thẻ HTML bên trong (không bỏ qua ký tự khoảng trắng) | <pre><b>Xin chào</b> các bạn</pre> → <b>Xin chào</b> các bạn |
| <br> | Kết thúc dòng hiện tại và chuyển sang dòng mới | <p>Dòng 1<br>Dòng 2</p>  → Dòng 1  Dòng 2 |

Thẻ <hr> tạo ra một đường kẻ ngang trên website, các thuộc tính của thẻ <hr>:

|  |  |
| --- | --- |
| Thuộc tính | Ý nghĩa |
| size | Độ dày của đường kẻ, đơn vị pixel |
| width | Độ dài của đường kẻ, đơn vị pixel |
| color | Màu của đường kẻ |
| align | Căn chỉnh vị trí của đường kẻ với 3 giá trị: left, right và center |

Ví dụ:

<hr size=“7” width=“300” color=“green” align=“right” />

***Thẻ hình ảnh <img>***

Cú pháp: <img src=“Đường\_dẫn\_ảnh” />

|  |  |
| --- | --- |
| Thuộc tính | Ý nghĩa |
| src=“url” | Đường dẫn của ảnh cần hiển thị |
| border=“giá\_trị” | Đường viền của ảnh |
| width=“giá\_trị” | Độ rộng của ảnh, đơn vị pixel |
| height=“giá\_trị” | Độ cao của ảnh, đơn vị pixel |
| alt=“giá\_trị” | Nội dung hiển thị thay thế khi ảnh không được hiển thị (do sai đường dẫn, ảnh không được tải về …) |
| title=“” | Tiêu đề của ảnh (khi di chuột qua vspace ảnh sẽ hiển thị tiêu đề) |
| vspace=“giá\_trị” | Khoảng cách từ ảnh đến thành phần khác bên ngoài theo chiều dọc |
| hspace=“giá\_trị” | Khoảng cách từ ảnh đến thành phần khác bên ngoài theo chiều ngang |

Lưu ý: Khi sử dụng thẻ <img /> nên đặt thuộc tính alt và title trong mọi trường hợp, 2 thuộc tính này có tác dụng SEO cho website, đồng thời chuẩn web mà W3C đưa ra yêu cầu mọi ảnh phải có.

Ví dụ:

<img src="banner.png" border="2" alt="Nội dung khi ảnh không hiển thị" title="Tiêu đề của ảnh" />

***Thẻ liên kết <a>***

Cú pháp: <a href=“đường\_dẫn”>Nội dung</a>

|  |  |
| --- | --- |
| Thuộc tính | Ý nghĩa |
| href=“url” | Đường dẫn tới liên kết |
| target=“giá\_trị” | Phương thức khi mở liên kết. Gồm các giá trị \_self(mở liên kết ở tab hiện tại), \_blank(mở với 1 cửa sổ mới), \_parent(mở với frame), \_top(khác với \_parent, \_topmở với cả trang) |
| title=“” | Tiêu đề của liên kết |

Ví dụ:

<a href=“ http://ictu.edu.vn” target=“\_blank” title=“DHTN”> Trang chủ Trường DHCNTT</a>

***Liên kết hình ảnh***

Trong trường hợp liên kết là 1 hình ảnh, chúng ta có thể thay “Nội dung” bằng cú pháp hình ảnh như trong ví dụ sau:

<a href=“ http://ictu.edu.vn” target=“\_blank” title=“Truong DH CNTT&TT”>

<img src=“ banner1.jpg“ />

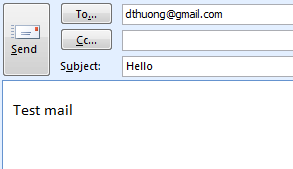
</a>

*Bản đồ ảnh*: Là một ảnh trong trang web được chia ra làm nhiều vùng, mỗi vùng khi click vào sẽ liên kết đến một địa chỉ URL.

*Liên kết tạo email*: Là chức năng tạo liên kết email.

Ví dụ:

<a href="mailto:dthuong@gmail.com?subject=Hello &body=Test mail">Click vào đây để gửi Email</a>



***Danh sách***

Việc liệt kê danh sách sẽ làm cho văn bản trở nên sáng sủa và dễ theo dõi. Trong HTML có một số kiểu danh sách như danh sách đánh số thứ tự, danh sách không đánh số thứ tự, danh sách định nghĩa.

*Danh sách không đánh số thứ tự*

Danh sách không đánh số thứ tự liệt kê các mục bằng một chấm tròn ở đầu mỗi mục.

Cú pháp:

<ul>

<li>Mục thứ 1</li>

<li>Mục thứ 2</li>

…

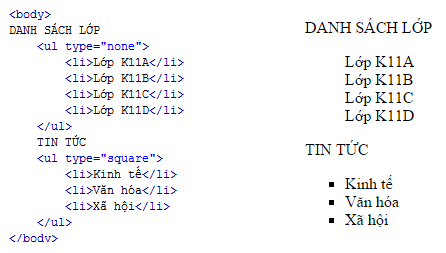
</ul>

Cặp thẻ <ul> … </ul> đánh dấu bắt đầu và kết thúc danh sách, nó có tác dụng lùi lề trái danh sách vào sâu hơn.

Các thẻ <li> … </li> có tác dụng thể hiện bắt đầu một dòng mới với một chấm tròn.

Theo mặc định, mỗi mục thuộc mức thứ nhất đánh thứ tự bắt đầu bằng một chấm tròn, còn mỗi mục thuộc mức thứ hai sẽ bắt đầu bằng một vòng tròn, ta có thể thay đổi bằng cách sử dụng thuộc tính TYPE.

Ví dụ:



*Danh sách đánh số thứ tự*

Danh sách đánh số thứ tự giống như danh sách không đánh số thứ tự nhưng các chấm tròn ở đầu mỗi mục sẽ được thay bởi việc đánh số thứ tự các mục đó.

Cú pháp:

<ol>

<li>Mục thứ 1</li>

<li>Mục thứ 2</li>

</ol>

Cặp thẻ <ol> … </ol> đánh dấu bắt đầu và kết thúc danh sách, nó có tác dụng lùi lề trái danh sách vào sâu hơn.

Các thẻ <li> … </li> có tác dụng thể hiện bắt đầu một dòng mới với một số thứ tự tương ứng.

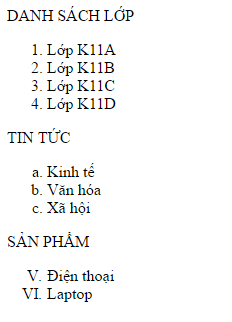
Có thể dùng thuộc tính TYPE để lựa chọn các cách đánh thứ tự khác nhau khi bắt đầu mỗi mục trong một danh sách có đánh số thứ tự:

|  |  |
| --- | --- |
| Thuộc tính | Ý nghĩa |
| type=“A” | Hiển thị theo dạng ký tự hoa A, B, C… |
| type=“a” | Hiển thị theo kiểu ký tự thường a, b, c… |
| type=“I” | Hiển thị dạng La Mã I, II, III, IV, … |
| type=“i” | Hiển thị dạng La Mã thường i, ii, iii, iv,… |
| type=“1” | Hiển thị dạng số thứ tự 1, 2, 3, 4, … (đây là lựa chọn mặc định) |
| start=“n” | Với n là giá trị bắt đầu của danh sách để bắt đầu với 1 giá trị khác 1 |

Ví dụ:



Kết quả hiển thị:



*Danh sách các định nghĩa*

Danh sách các định nghĩa trông giống như một bảng từ vựng hay bảng giải thích các thuật ngữ lùi vào trong.

Cú pháp:

<dl>

<dt>Mục thứ 1</dt>

<dd>Nội dung mục thứ 1</dd>

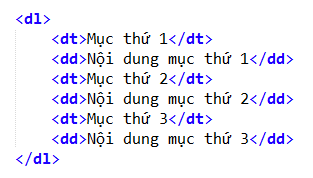
<dt>Mục thứ 2</dt>

<dd>Nội dung mục thứ 2</dd>

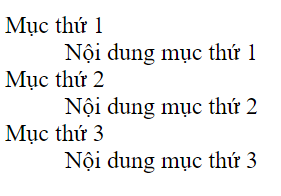
…

</dl>

Ví dụ:



Kết quả hiển thị:



***Các thẻ đặc biệt***

*Meta tags*

Mọi thẻ meta đều đặt trong cặp thẻ <head></head>, cung cấp thông tin về website cho trình duyệt và các Search Engine.

*Keyword*

Tạo từ khóa cho website, cung cấp cho các search engine.

Cú pháp:

<meta name=“keywords” content=“các từ khóa cách nhau bởi dấu phảy” />

*Description*

Thẻ mô tả về website, cung cấp thông tin về web.

Cú pháp:

<meta name=“description” content=“mô tả ngắn gọn về website” />



*Refresh*

Tự động tải lại trang trong một khoảng thời gian được định nghĩa.

Cú pháp:

<meta http-equiv=“refresh” content=“time” />

*Author*

Cung cấp thông tin về tác giả website.

Cú pháp:

<meta name=“author” content=“” />

*Charset*

Xác định kiểu charset mà website sử dụng.

Cú pháp:

<meta charset=“utf-8” />

*Style tags*

Cặp thẻ <style></style> được đặt trong cặp thẻ <head></head>, xác định thông tin định dạng website. Trong một tài liệu HTML có thể sử dụng nhiều cặp thẻ <style></style>

*Script tags*

Bên trong cặp thẻ <head></head>, chứa các thông tin về mã kịch bản Javascript

Ví dụ:

<script type=“text/javascript”> document.write(“Xin chào”);

</script>

**1.5 Phương pháp tạo bảng**

***1.5.1 Khung cấu trúc***

Giới hạn bảng: <table> … </table>

Định nghĩa một hàng: <tr> … </tr>

Định nghĩa một ô: <td> … </td>

Ô chứa tiêu đề: <th> … </th>

Cú pháp tạo khung cấu trúc của một bảng:

<table>

<tr>

<th>Tên cột 1</th>

<th>Tên cột 2</th>

…

</tr>

<tr>

<td>Ô 1 dòng 1</td>

<td>Ô 2 dòng 1</td>

…

</tr>

…

</table>

Ví dụ:

<table>

<tr>

<th>Họ và tên</th>

<th>Lớp</th>

<th>Quê quán</th>

</tr>

<tr>

<td>Bùi Thanh Hằng</td>

<td>CNTTK13B</td>

<td>Bắc Ninh</td>

</tr>

<tr>

<td>Nguyễn Văn Quyền</td>

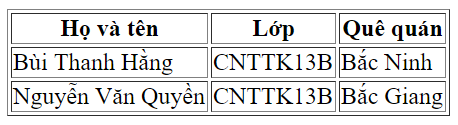
<td>CNTTK13B</td>

<td>Bắc Giang</td>

</tr>

</table>

Kết quả thu được bảng như sau:



***1.5.2 Một số lưu ý về bảng***

Ô rỗng thì không có đường bao, muốn ô rỗng có đường bao phải dung dấu cách hoặc xuống dòng: <td>&nbsp;</td> hoặc <td><br></td>

Có thể sử dụng rowspan và colspan để tạo bảng có ô chồng chéo lên nhau

***Thẻ <table>*** có các thuộc tính sau:

|  |  |
| --- | --- |
| Thuộc tính | Ý nghĩa |
| width | Độ rộng của bảng, thường sử dụng giá trị kích thước có đơn vị % hoặc px |
| height | Chiều cao bảng, thường sử dụng giá trị kích thước có đơn vị % hoặc px |
| bgcolor | Định màu nền của bảng |
| background | Định ảnh nền của bảng |
| border | Độ lớn đường viền của bảng |
| bordercolor | Màu của đường viền |
| align | Căn vị trí của bảng so với toàn website. Giá trị: left, right, center |
| cellspacing | Định độ dày của khung |
| cellpadding | Định khoảng cách từ nội dung đến đường bao của bảng |

***Thẻ <tr>*** có các thuộc tính sau:

|  |  |
| --- | --- |
| Thuộc tính | Ý nghĩa |
| height | Khai báo chiều cao của dòng (độ lớn dòng) |
| align | Định vị trí nội dung trong dòng (theo chiều ngang): left (trái), right (phải), center (giữa) |
| valign | Định vị trí nội dung theo chiều dọc: top (trên), middle (giữa), bottom (dưới) |

***Thẻ <td>*** có các thuộc tính sau:

|  |  |
| --- | --- |
| Thuộc tính | Ý nghĩa |
| width | Khai báo độ rộng của cột |
| height | Khai báo chiều cao của cột |
| align | Định vị trí nội dung trong cột( theo chiều ngang): left (trái), right (phải), center (giữa) |
| valign | Định vị trí nội dung theo chiều dọc: top (trên), middle (giữa), bottom (dưới) |

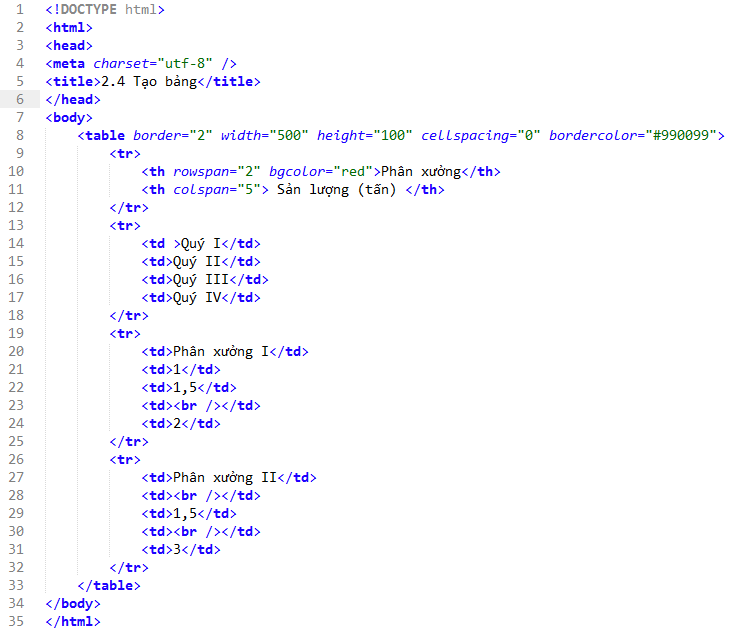
***Thẻ <th>*** tương tự cặp thẻ <td></td> tuy nhiên <th> dùng trong trường hợp nếu ô đó chứa tiêu đề của cột. Nội dung trong cặp thẻ này được in đậm và căn giữa tự động.

***Gộp cột, gộp dòng*** trong bảng là việc tùy biến bảng bằng việc gộp các cột hay dòng lại với nhau:

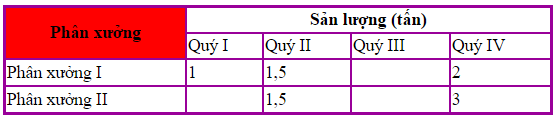
|  |  |
| --- | --- |
| Thuộc tính | Ý nghĩa |
| colspan=“x” | Gộp x cột tính từ cột đang đặt thuộc tính colspan |
| rowspan=“y” | Gộp y hàng tính từ hàng đang đặt thuộc tính rowspan |

Lưu ý: Sau khi gộp phải loại bỏ số dòng (hoặc cột) để cân đối bảng

***1.5.3 Ví dụ***



Kết quả hiển thị:



**1.6 Form và các điều khiển trên form**

***1.6.1 Chức năng của form***

Để mở rộng khả năng phục vụ người sử dụng, dịch vụ Web cần phải sử dụng các ứng dụng khác bên ngoài. Ví dụ người sử dụng yêu cầu tìm kiếm trong cơ sở dữ liệu, lấy các thông in tức thời, luôn được cập nhật… Để là được điều này dịch vụ web phải chuyển yêu cầu của người sử dụng đến một ứng dụng khác. Ứng dụng này sẽ thực hiện yêu cầu và trả lại kết quả cho web server để chuyển tiếp đến người sử dụng.

Form là một cách để chuyển dữ liệu từ người sử dụng đến web server xử lý. Form được sử dụng rộng rãi trên www. Các form rất tiện lợi cho người dùng điền các yêu cầu tìm kiếm, các biểu mẫu điều tra, nhập dữ liệu đầu vào cho các ứng dụng.

Có nhiều thành phần khác nhau (gọi là các điều khiển control) trong một form. Tùy theo yêu cầu giao tiếp với người sử dụng cần chọn thành phần thích hợp nhất.

Khi tạo form cần phải chỉ rõ cho máy chủ dịch vụ biết cách xử lý form. Có nhiều loại chương trình ứng dụng khác nhau trong máy chủ dịch vụ để làm việc này: các chương trình CGI, ISAPI, các script ASP, JSP, Java Bean, Servlet …

Tạo form là khâu đầu tiên trong việc xây dựng giao tiếp giữa người sử dụng với các ứng dụng Internet/Intranet.

***1.6.2 Các thành phần trong form***

Cặp thẻ để tạo form là <form> … </form>

Mọi thành phần của form đều phải nằm trong phạm vi giới hạn của cặp thẻ này.

Công thức khung để tạo form là:

<form name=“tên\_form” method=“POST” action=“url”>

Toàn bộ các thành phần bên trong form nằm ở vị trí này

</form>

Thuộc tính method chỉ ra phương thức trao đổi dữ liệu giữa trình duyệt web và máy chủ web. Có hai phương thức là POST và GET, đối với form phương thức thường là POST.

Thuộc tính action để thiết lập địa chỉ URL của chương trình sẽ nhận và xử lý dữ liệu gửi từ form.

*a) Textbox*

Tạo nên đối tượng cho phép nhập dữ liệu văn bản

Cú pháp:

<input type=“text” name=“” value=“” size=“” />

Thuộc tính name: Tên của textbox, đặt tùy ý, được kết hợp với PHP hay Javascript để tạo tính tương tác trên website.

Thuộc tính value: Giá trị ban đầu của textbox.

Thuộc tính size: Độ rộng của textbox.

Nếu thẻ input không sử dụng thuộc tính type=“text” thì trình duyệt tự hiểu dạng Textbox

*b) Password*

Đối với các form như đăng nhập chúng ta dùng loại này để ẩn đối tượng password

Cú pháp:

<input type=“password” value=“ ” name=“” size=“” />

Thuộc tính name: Tên của password, đặt tùy ý, được kết hợp với PHP hay Javascript để tạo tính tương tác trên website.

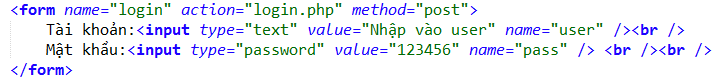
Thuộc tính value: Giá trị ban đầu của password.

Thuộc tính size: Độ rộng của password.

Password khác Textbox ở chỗ khi gõ các ký tự từ bàn phím thì nó không hiển thị ký tự tương ứng trong khung mà thay bằng các dấu \*.

Lưu ý: việc hiển thị ký tự dấu \* để thay thế chỉ có tác dụng che giấu với người nhìn tại máy cục bộ. Password vẫn được gửi vào mạng dưới dạng rõ, không mã hóa, có thể bị xem trộm. Nhiệm vụ mã là của giao thức mạng.

Ví dụ:



Kết quả hiển thị:



*c) Checkbox*

Ô đánh dấu để đánh dấu một hoặc vài lựa chọn.

Cú pháp:

<input type=“checkbox” name=“” value=“” checked=“checked”/>

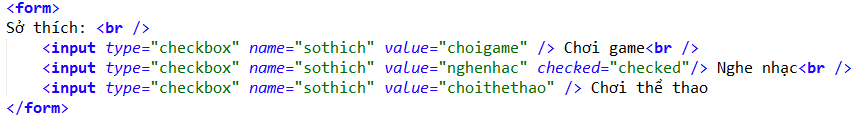
Thuộc tính name: Tên của checkbox, đặt tùy ý, được kết hợp với PHP hay Javascript để tạo tính tương tác trên website.

Thuộc tính value: Giá trị ban đầu của checkbox.

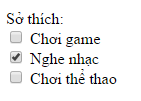
Thuộc tính checked=“checked”: Để lựa chọn tích ban đầu.

Lưu ý: Đặt name cùng giá trị với các đối tượng trong cùng nhóm

Ví dụ:



Kết quả hiển thị:



*d) Radio*

Các nút chọn radio là một nhóm nút tròn, chỉ cho phép được chọn một trong nhiều nút.

Cú pháp:

<input type=“radio” name=“” value=“” checked=“checked”/>

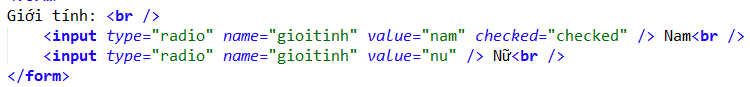
Thuộc tính name: Tên của radio, đặt tùy ý, được kết hợp với PHP hay Javascript để tạo tính tương tác trên website.

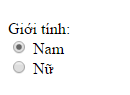
Thuộc tính value: Giá trị ban đầu của radio.

Thuộc tính checked=“checked”: Để lựa chọn tích ban đầu.

Lưu ý: Đặt name cùng giá trị với các đối tượng trong cùng nhóm

Ví dụ:





*e) Upload*

Tạo chức năng duyệt file từ máy tính.

Cú pháp:

<input type=“file” name=“” value=“” checked=“checked”/>

Thuộc tính name: Tên của file.

Thuộc tính value: Giá trị ban đầu của file.

Ví dụ:



Kết quả hiển thị:



*f) Submit*

Tạo nút gửi dữ liệu tới file xử lý (Tới link khai báo ở thuộc tính Action)

Cú pháp:

<input type=“submit” name=“” value=“” />

Thuộc tính name: Tên của submit, đặt tùy ý, được kết hợp với PHP hay Javascript để tạo tính tương tác trên website.

Thuộc tính value: Nội dung hiển thị trên nút submit.

*g) Button*

Tương tự như Submit tuy nhiên khi sử dụng Button dữ liệu sẽ không được gửi tới file xử lý (Khai báo phần action) mà thường kết hợp với JS để tạo tương tác

Cú pháp:

<input type=“button” name=“” value=“” />

Thuộc tính name: Tên của button, đặt tùy ý, được kết hợp với PHP hay Javascript để tạo tính tương tác trên website.

Thuộc tính value: Nội dung hiển thị trên nút button.

*h) Reset*

Trả lại giá trị ban đầu khi chưa nhập liệu cho tất cả các đối tượng trong form

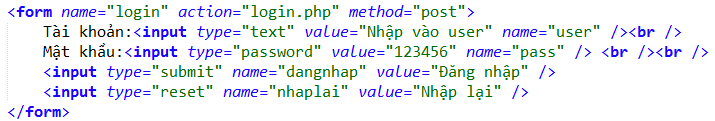
Cú pháp:

<input type=“reset” name=“” value=“” />

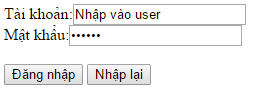
Thuộc tính name: Tên của reset, đặt tùy ý, được kết hợp với PHP hay Javascript để tạo tính tương tác trên website.

Thuộc tính value: Nội dung hiển thị trên nút reset.

Ví dụ:



Kết quả hiển thị:



*i) Select tag*

Tạo danh sách lựa chọn

Cú pháp:

<select name=“” size=“”>

<option>Item 1</option>

<option selected>Item 2</option>

…

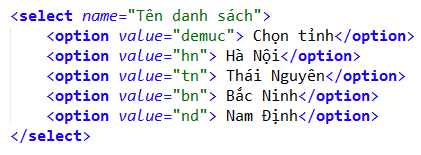
</select>

Thuộc tính name: Tên của select, đặt tùy ý, được kết hợp với PHP hay Javascript để tạo tính tương tác trên website.

Thuộc tính size xác định số mục chọn có thể nhìn thấy trong cửa sổ cuộn, mặc định danh sách chọn select chỉ cho phép một mục chọn, thuộc tính multiple cho phép đánh dấu chọn nhiều mục.

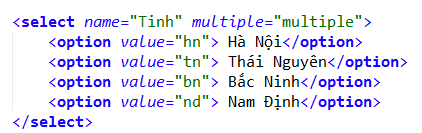
Thuộc tính option selected xác định lựa chọn mặc định ban đầu.

Ví dụ:

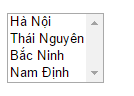


Kết quả hiển thị:





Kết quả hiển thị:



*k) Textarea*

Vùng văn bản để hiển thị nhiều dòng văn bản.

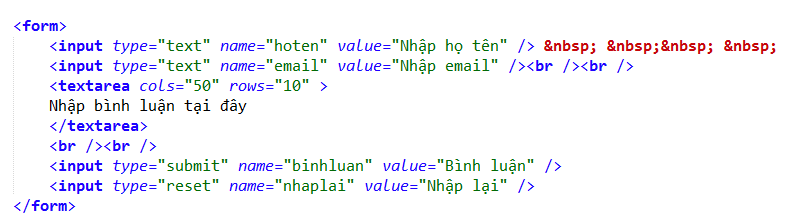
Cú pháp:

<textarea cols=“số cột” rows=“số dòng” maxlength=“số ký tự tối đa cho phép”>

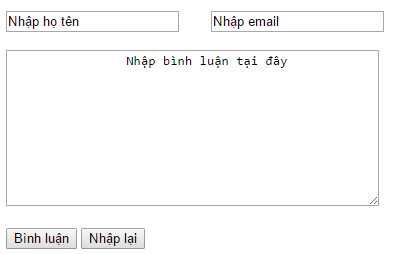
Nội dung

</textarea>

Ví dụ:



Kết quả hiển thị:



# CHƯƠNG II - CSS

**2.1 Giới thiệu về CSS**

***2.1.1 Giới thiệu***

CSS (Cascading Style Sheets mà Pearl tạm dịch là tờ mẫu theo Style Sheets) là một ngôn ngữ quy định cách trình bày cho các tài liệu viết bằng HTML, XHTML, XML, SVG, hay UML, …

HTML cũng hỗ trợ một số thuộc tính định dạng cơ bản cho text, picture, table, … nhưng nó không thật sự phong phú và chính xác như nhau trên mọi hệ thống. CSS cung cấp các thuộc tính trình bày dành cho các đối tượng với sự sáng tạo cao trong kết hợp các thuộc tính giúp mang lại hiệu quả. Ngoài ra, hiện tại CSS đã được hỗ trợ bởi tất cả các trình duyệt, nên trang web có thể hiển thị hầu như “như nhau” dù trên một hệ thống sử dụng Windows, Linux hay trên một máy Mac miễn là phiên bản trình duyệt mới nhất đang được sử dụng.

Sử dụng các mã định dạng trực tiếp trong HTML tốn hao nhiều thời gian thiết kế cũng như dung lượng lưu trữ trên đĩa cứng. Trong khi đó CSS đưa ra phương thức “tờ mẫu ngoại” giúp áp dụng một khuôn mẫu chuẩn từ một file CSS ở ngoài. Nó thật sự có hiệu quả đồng bộ khi tạo một website có hàng trăm trang hay cả khi muốn thay đổi một thuộc tính trình bày nào đó. Khi chúng ta có một website với hàng trăm trang và muốn thay đổi font chữ hay màu chữ cho một thành phần nào đó. Đó thật sự sẽ là một công việc tốn nhiều thời gian và rất khó để kiểm soát được hết, nhưng sử dụng CSS việc đó là hoàn toàn đơn giản.

Ngoài ra, CSS còn cho phép định dạng những kiểu trình bày thích hợp hơn cho các phương tiện khác nhau như màn hình máy tính, máy in, điện thoại,…

***2.1.2 Cú pháp CSS***

Cú pháp CSS cơ bản:

Selector{ property: value;}

Trong đó:

*Selector:* Các đối tượng mà chúng ta sẽ áp dụng các thuộc tính trình bày. Nó là các tag HTML, class hay id (chúng ta sẽ học về hai thành phần này ở phần sau).

Ví dụ: body, h2, p, img, #title, #content, .username,…

Trong CSS ngoài viết tên selector theo tên tag, class, id. Chúng ta còn có thể viết tên selector theo phân cấp như để chỉ các ảnh ở trong #entry, chúng ta viết selector là #entry img, như vậy thì các thuộc tính chỉ định sẽ chỉ áp dụng riêng cho các ảnh nằm trong #entry.

Khi viết tên cho class, đôi khi sẽ có nhiều thành phần có cùng class đó, ví dụ như thẻ img và thẻ a cùng có class tên vistors nhưng đây lại là hai đối tượng khác nhau, 1 cái là ảnh của người thăm, 1 cái là liên kết tới trang người thăm. Nên nếu khi viết CSS ta ghi là .visitors { width:50 } thì sẽ ảnh hưởng tới cả hai thành phần. Nên trong trường hợp này, nếu dùng CSS đó chỉ riêng phần ảnh thì chỉ nền ghi là img .visitors thôi.

Một cách viết tên selector nữa đó là dựa trên tên các thuộc tính có trong HTML. Ví dụ trong HTML ta có đoạn mã: <input name=”Search” type=”Text” value=”Key Word”>. Để áp dụng thuộc tính CSS cho riêng ô tìm kiếm này chúng ta sẽ dùng selector input[name=”Search”].

Ngoài việc viết tên selector cụ thể, chúng ta cũng có thể dùng một selector 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.

*Property:* 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 selector 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 selector.

Ví dụ:

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

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

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;

}

*Value:* Giá trị của thuộc tính.

Đối với một giá trị có khoảng trắng, 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.

Ví dụ: width:100 px. Nó sẽ làm CSS 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 \*/

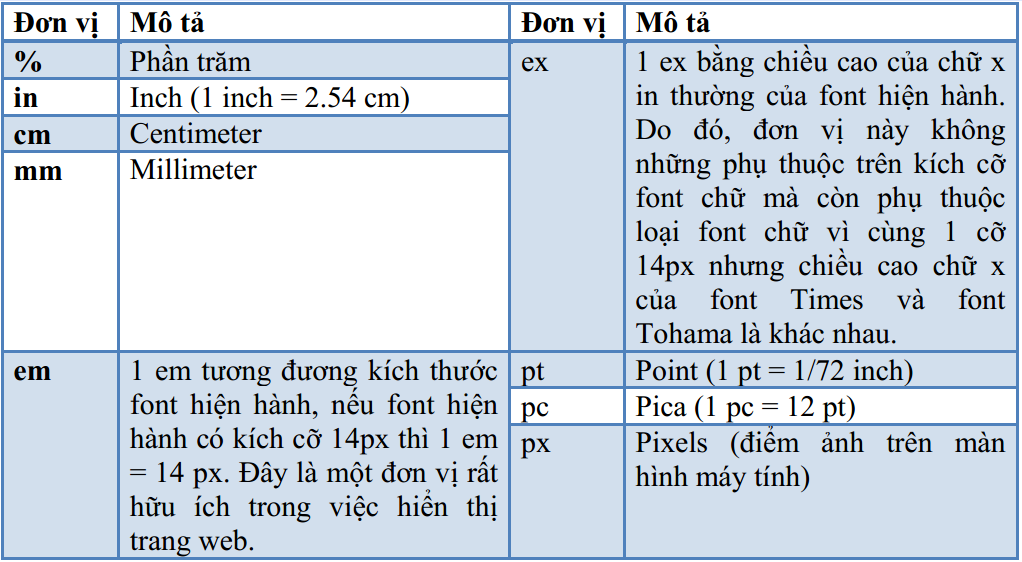
body {

color: red;

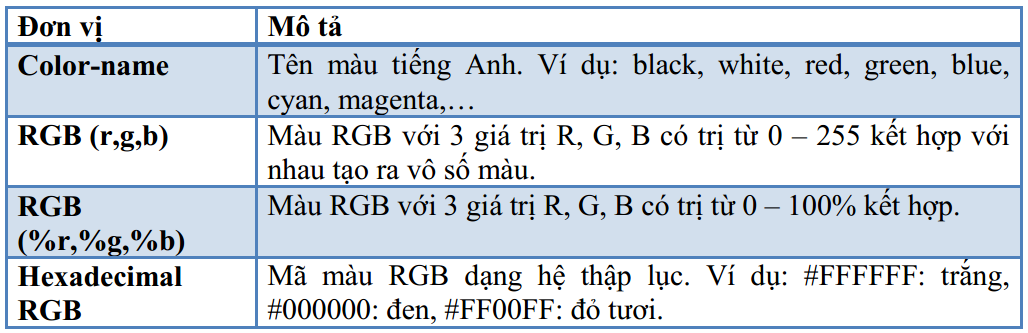
}

***2.1.3 Đơn vị CSS***

*Đơn vị chiều dài*

**

*Đơn vị màu sắc*

**

***2.1.4 Vị trí đặt CSS***

Có ba cách nhúng khác nhau để nhúng CSS vào trong một tài liệu HTML

*Cách 1: Nội tuyến (kiểu thuộc tính)*

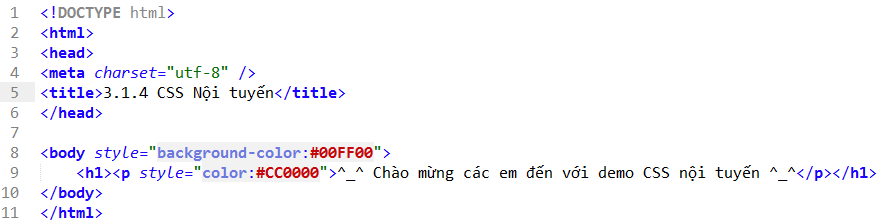
Đây là một phương pháp nguyên thủy nhất để nhúng CSS vào một tài liệu HTML bằng cách nhúng vào từng thẻ HTML muốn áp dụng. Và trong trường hợp này chúng ta sẽ không cần selector trong cú pháp.

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

Cú pháp:

<tên\_thẻ style=“thuộc\_tính: giá\_trị;”>Nội dung</tên\_thẻ>

Ví dụ:



Kết quả hiển thị:



*Cách 2: Bên trong (thẻ style)*

Thật 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 quản lý. Lưu ý: thẻ style nên đặt trong thẻ head.

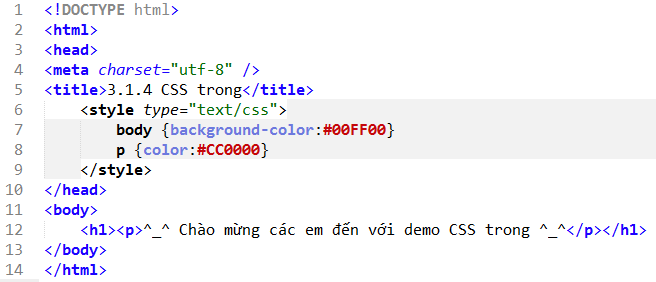
Cú pháp:

<style type=“text/css”>

selector { property: value;}

</style>

Ví dụ:



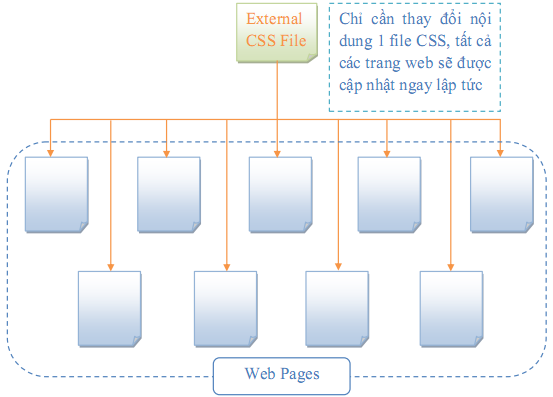
Kết quả hiển thị:



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

Tương tự như cách 2 nhưng thay vì đặt tất cả các mã CSS trong thẻ style chúng ta sẽ đưa chúng vào trong một file CSS (có phần mở rộng .css) bên ngoài và liên kết nó vào trang web bằng thuộc tính href trong thẻ link.

Đây là cách nên sử dụng vì nó đặc biệt hữu ích cho việc đồng bộ hay bảo trì một website lớn sử dụng cùng một kiểu mẫu. Các ví dụ trong cuốn bài giảng này cũng được trình bày theo kiểu này.



Cú pháp liên kết được đặt trong thẻ head:

<link rel=“stylesheet” type=“text/css” href=“liên\_kết\_tới\_file\_css” />

Cú pháp nội dung định dạng trong file có phần đuôi .css:

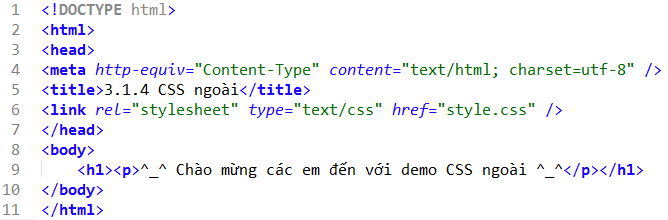
<style type=“text/css”>

selector { property: value;}

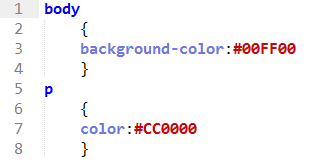
</style>

Ví dụ:

.html



.css



Kết quả hiển thị:



*Sự ưu tiên:*

Trước khi thực thi CSS cho một 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 liên kết vào trang web, CSS nhúng trong thẻ <style> và các CSS nội tuyến. Sau đó, trình duyệt sẽ tổng hợp toàn bộ CSS này vào một CSS ảo, và nếu có các thuộc tính CSS giống nhau thì thuộc tính CSS nào nằm sau sẽ được ưu tiên sử dụng. Theo nguyên tắc đó trình duyệt sẽ ưu tiên cho các CSS nội tuyến > CSS bên trong > CSS bên ngoài > CSS mặc định của trình duyệt.

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 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.

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

**2.2 Các thuộc tính trong CSS**

***2.2.1 Font và text***

a) *Font chữ*

***Thuộc tính font-family***

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 chúng 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 dự phòng như Arial, Tohama hay Times New Roman và cũng được đề nghị đặt vào danh sách font một generic families (thường thì nó sẽ có độ ưu tiên thấp nhất). Thực hiện theo cách này thì sẽ đảm bảo trang web có thể hiển thị tốt trên bất kỳ hệ thống nào.

Ví dụ sau chúng ta 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.

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***

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 thực hiện áp dụng kiểu in nghiêng cho thành phần h1 và kiểu xiên cho h2.

h1 {

font-style:italic;

}

h2 {

font-style:oblique;

}

***Thuộc tính font-variant***

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.

Ví dụ:

h1 {

font-variant:small-caps

}

***Thuộc tính font-weight***

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 thì có thể cân nhắc sử dụng các đơn vị quy 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>

*b) Text*

***Màu chữ (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 chúng ta 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.

body {

color:#000

}

h1 {

color:#0000FF

}

h2 {

color:#00FF00

}

***Thuộc tính text-indent***

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 chúng ta 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>.

p {

text-indent:30px

}

***Thuộc tính text-align***

Thuộc tính text-align giúp 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 chúng ta 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***

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***

Thuộc tính text-decoration giúp 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 chúng ta 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***

Text-transform là thuộc tính quy 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 chúng ta 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

}

***2.2.2 Liên kết trong CSS***

Một thành phần rất quan trọng trong mọi website chính là liên kết.Cũng như một đối tượng văn bản thông thường, chúng ta hoàn toàn có thể áp dụng các thuộc tính định dạng đã học ở phần 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 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).

Ví dụ 1: Ví dụ này chúng ta 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 mall-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 một số hiệu ứng: các liến 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;

}

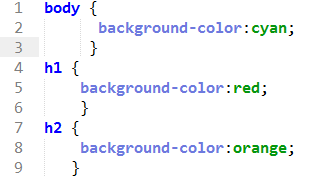
***2.2.3 Border và background trong CSS***

*a) Background*

***Màu nền (thuộc tính background-color)***

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.



***Ảnh nền (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.

Ví dụ:

body{

background-image: url(img1.jpg);

}

Chúng ta phải chỉ định đường dẫn của ảnh trong cặp ngoặc đơn sau url. Do ảnh đặt trong cùng thư mục với file .css nên chúng ta chỉ cần ghi img1.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/img1.jpg) hoặc để chắc chắn chúng ta có thể sử dụng đường dẫn tuyệt đối.

***Lặp lại ảnh nền (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.

Ví dụ:

body{

background-image: url(img1.jpg);

background-repeat: no-repeat;

}

***Khóa ảnh nền (thuộc tính background-attachment)***

Background-attachment là một thuộc tính cho phép 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 cuộn trang web.

***Định vị ảnh nền (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 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. 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.

Ví dụ:

|  |  |
| --- | --- |
| Giá trị | Ý nghĩa |
| Background-position: 10px 5px | Ảnh được định vị 10px từ trái qua và 5px 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.

Ví dụ: Chúng ta có thể nhóm lại đoạn CSS sau

background-color:transparent;

background-image: url(img1.jpg);

background-repeat: no-repeat;

background-attachment: fixed;

background-position: right bottom;

thành một dòng ngắn gọn:

background:transparent url(logo.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.

Ví dụ: Chúng ta sẽ bỏ qua hai thuộc tính background-attachment và background-position ở dòng mã trên đi:

background:transparent url(img1.jpg) no-repeat;

Hai thuộc tính không được chỉ định sẽ được thiết lập tới giá trị mặc định là scroll và top left.

*b) Border*

Border là một thành phần quan trọng trong một trang web. Nó thường được dùng trong trang trí, đóng khung cho một đối tượng cần nhấn mạnh, phân cách các đối tượng giúp trang web trông dễ nhìn hơn,…

Border có các thuộc tính sau:

* Thuộc tính border-width

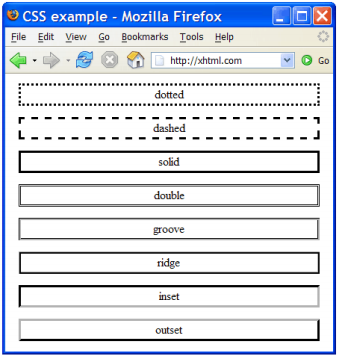
Border-width là một thuộc tính CSS quy định độ rộng cho viền của một đối tượng web. Thuộc tính này có các giá trị: thin (mảnh), medium (vừa), thick (dày), hay là một giá trị đo cụ thể như pixels.

* Thuộc tính border-color

Border-color là thuộc tính CSS quy định màu viền cho một đối tượng web. Thuộc tính này nhận tất cả đơn vị màu CSS hỗ trợ.

* Thuộc tính border-style

Border-style là thuộc tính CSS quy định kiểu viền thể hiện của một đối tượng web. CSS cung cấp tất cả 8 kiểu viền tương ứng với 8 giá trị: dotted, dashed, solid, double, groove, ridge, inset và outset. Ngoài ra, hai giá trị none hay hidden dùng để ẩn đường viền.



Ví dụ sau sẽ định viền cho 3 thành phần h1, h2, p

h1 { border-width:thin;

border-color:#FF0000;

border-style:solid }

h2 { border-width:thick;

border-color:#CCC;

border-style:dotted }

p { border-width:5px;

border-color:#FF00FF;

border-style:double }

Ngoài ra, chúng ta cũng có thể dùng riêng các thuộc tính border-top, border-right, border-bottom hay border-left để chỉ định viền riêng cho các đối tượng.

Ví dụ:

h1 {

border-top-width:thin;

border-top-color:#FF0000;

border-top-style:solid;

border-right-width:thick;

border-right-color:#AFAFAF;

border-right-style:dotted;

border-bottom-width:2px;

border-bottom-color:blue;

border-bottom-style:double;

border-left-width:5px;

border-left-color:violet;

border-left-style:groove

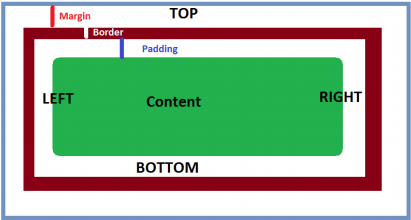
}

Thuộc tính border rút gọn:

border:<border-width> |<border-color> |<border-style>

***2.2.4 Padding và margin***

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, border, margin và các tùy chọn. Hình bên dưới mô tả cấu trúc minh họa mô hình hộp cho một thành phần web.



Ví dụ:

<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;

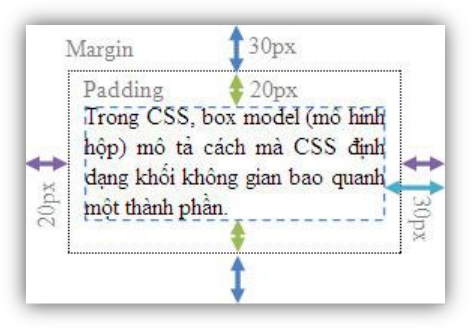
padding:20px 10px;

border:1px solid #000;

text-align:justify

}

Kết quả được mô tả như sau:



*a) Margin*

Thuộc tính margin trong CSS 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.

Ví dụ:

body {

margin-top:80px;

margin-bottom:40px;

margin-left:50px;

margin-right:30px;

border:1px dotted #FF0000

}

Hoặc viết ngắn gọn như sau:

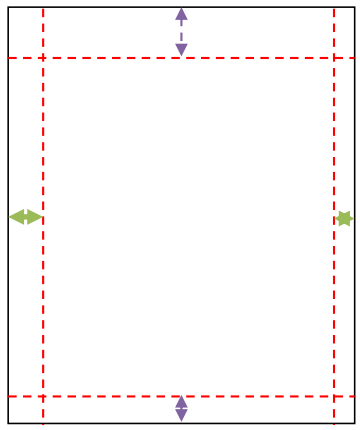
body {

margin:80px 30px 40px 50px;

border:1px dotted #FF0000

}

Kết quả được mô tả như sau:



Cú pháp chung:

margin:<margin-top> | <margin-right> | <margin-bottom> | <margin-left>

Hoặc

margin:<value1>|< value2>

với value1 là giá trị margin-top và margin-bottom và value2 là giá trị margin-left và margin-right.

*b) Padding*

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.

Cú pháp:

padding:<margin-top> | <margin-right> | <margin-bottom> | <margin-left>

Ví dụ:

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

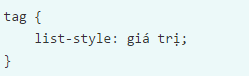
}

***2.2.5 Danh sách và thuộc tính display trong CSS***

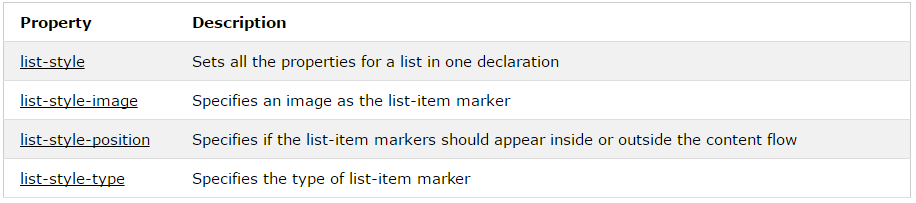
*a) Danh sách*

Thuộc tính list-style thiết lập kiểu cho một danh sách

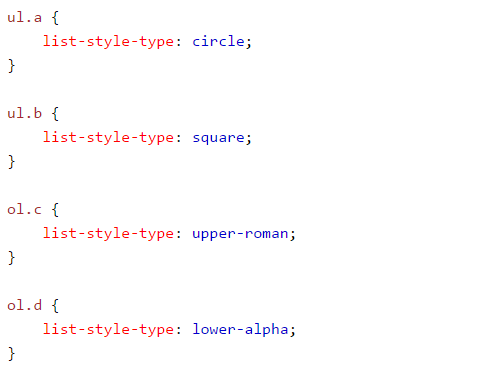
Cú pháp:

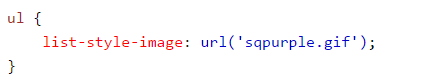


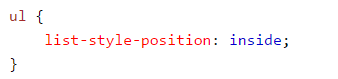
Với giá trị như sau:

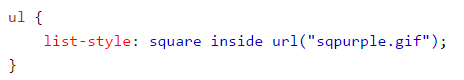


Ví dụ:





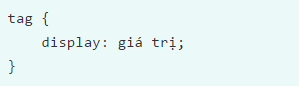




*b) Display*

Thuộc tính display xác định loại hiển thị của thành phần.

Cú pháp:



Với giá trị như sau:

|  |  |  |  |
| --- | --- | --- | --- |
| **Thuộc tính** | **giá trị** | **Ví dụ** | **Mô tả** |
| [display](http://hocwebchuan.com/reference/cssSection/example/ex_display.php) | [block](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr01) | display: block; | Thành phần hiển thị như một khối, khi sử dụng giá trị block thành phần sẽ đứng một hàng độc lập so với thành phần trước và sau nó. |
| [inline](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr02) | display: inline; | Thành phần sẽ hiển thị như một nội tuyến (inline, không ngắt dòng), đây là dạng mặc định. |
| [inline-block](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr03) | display: inline-block; | Thành phần sẽ hiển thị như một khối, nhưng là một khối nội tuyến. |
| [inline-table](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr04) | display: inline-table; | Thành phần sẽ hiển thị như một khối nội tuyến, đối xử tương tự [<table>](http://hocwebchuan.com/reference/tag/tag_table.php), không ngắt dòng trước và sau thành phần. |
| [list-item](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr05) | display: list-item; | Thành phần sẽ hiển thị như một khối và một nội tuyến cho các điểm đánh dấu danh sách. |
| [none](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr06) | display: none; | Thành phần không hiển thị. |
| run-in | display: run-in; | Thành phần sẽ hiển thị như một khối hoặc một nội tuyến, tùy thuộc vào bối cảnh. |
| [table](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr07) | display: table; | Thành phần sẽ đối xử như một table, ngắt dòng trước và sau thành phần. |
| [table-caption](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr08) | display: table-caption; | Thành phần sẽ đối xử như một [<caption>](http://hocwebchuan.com/reference/tag/tag_caption.php) của table. |
| [table-cell](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr08) | display: table-cell; | Thành phần sẽ đối xử như một ô trong table. |
| [table-column](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr08) | display: table-column; | Thành phần sẽ đối xử như một cột trong table. |
| [table-column-group](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr08) | display: table-column-group; | Thành phần sẽ đối xử như một nhóm cột ([<colgroup>](http://hocwebchuan.com/reference/tag/tag_colgroup.php)) trong table. |
| [table-footer-group](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr08) | display: table-footer-group; | Thành phần sẽ đối xử như một nhóm footer ([<tfoot>](http://hocwebchuan.com/reference/tag/tag_tfoot.php)) trong table. |
| [table-header-group](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr08) | display: table-header-group; | Thành phần sẽ đối xử như một nhóm header ([<thead>](http://hocwebchuan.com/reference/tag/tag_thead.php)) trong table. |
| [table-row](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr08) | display: table-row; | Thành phần sẽ đối xử như một hàng trong table. |
| [table-row-group](http://hocwebchuan.com/reference/cssSection/example/ex_display.php#pr08) | display: table-row-group; | Thành phần sẽ đối xử như một nhóm hàng trong table. |
| inherit | display: inherit; | Xác định thừa hưởng thuộc tính từ thành phần cha (thành phần bao ngoài). |

**2.3 Xây dựng giao diện website với HTML và CSS**

***2.3.1 Class và ID***

*a) Class*

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>Thái Nguyên</li>

<li>Khánh Hòa</li>

<li>Quảng Ninh</li>

<li>Bắc 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 như sau:

<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”>Thái Nguyên</li>

<li class=”tinh”>Khánh Hòa</li>

<li class=”tinh”>Quảng Ninh</li>

<li class=”tinh”>Bắc 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

}

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

*b) Id*

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”>Thái Nguyên</li>

<li class=”tinh”>Khánh Hòa</li>

<li class=”tinh”>Quảng Ninh</li>

<li class=”tinh”>Bắc 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 với ký tự đầu là chữ số, nó sẽ không làm việc cho Firefox.

Qua hai ví dụ trên chúng ta có thể rút ra những kết luận sau:

* Class dùng để nhóm các đối tượng có cùng thuộc tính, do tính chất đó nó có thể được sử dụng nhiều lần.
* Id dùng để nhận dạng một đối tượng đặc trưng, id có tính duy nhất

***2.3.2 Span và Div***

*a) Span*

Thẻ span trong HTML là một công cụ đánh dấu, nó không thêm hay bớt bất cứ một cái gì vào một tài liệu HTML, qua đó chúng ta có thể viết CSS định đạng cho các phần tử mong muốn.

Ví dụ:

<p>Ví dụ về thẻ span trong CSS</p>

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

<p>Ví dụ về thẻ <span class=“indam”>span</span> trong CSS</p>

Sau đó chúng ta viết CSS cho yêu cầu trên:

.indam{

font-weight: bold;

}

*b) Div*

Thẻ div giống như thẻ span được thêm vào tài liệu HTML với mục đích nhóm các phần tử lại để 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.

Xét lại ví dụ về danh sách tỉnh, 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>Thái Nguyên</li>

<li>Khánh Hòa</li>

<li>Quảng Ninh</li>

<li>Bắc Giang</li>

</div>

</ul>

Và đoạn CSS như sau:

#tp {

color:#FF0000

}

#tinh {

color:0000FF

}

***2.3.3 Float và Clear***

*a) Float*

Float là một thuộc tính CSS dùng để cố định một thành phần web về bên trái hay bên phải không gian bao quanh nó. Đây là một thuộc tính rất cần thiết khi dàn trang (như tạo các trang web 2 column layout hay 3 column layout), hiển thị văn bản thành cột, hay thực hiện việc định vị trí ảnh và text.



Thuộc tính float có 3 giá trị:

* Left: Cố định phần tử về bên trái.
* Right: Cố định phần tử về bên phải.
* None: Bình thường.

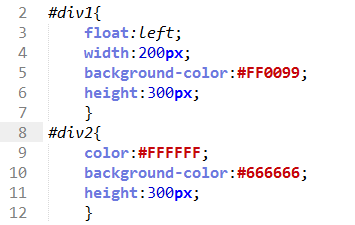
Nhìn vào hình minh họa trên chúng ta thấy là ban đầu trong box lớn có hai thành phần là Box B và phần Content. Lúc đầu Box B nằm bên trên và Content nằm bên dưới, nhưng khi chúng ta đặt thuộc tính float cho Box B thì Box B bị cố định về bên trái và chừa lại khoảng trống bên phải nó. Còn phần Content thì vốn nằm bên dưới sẽ tự động tràn lên để lắp đầy khoảng trống do Box B tạo ra.

Ví dụ:

.html



.css



Kết quả hiển thị:



*b) Clear*

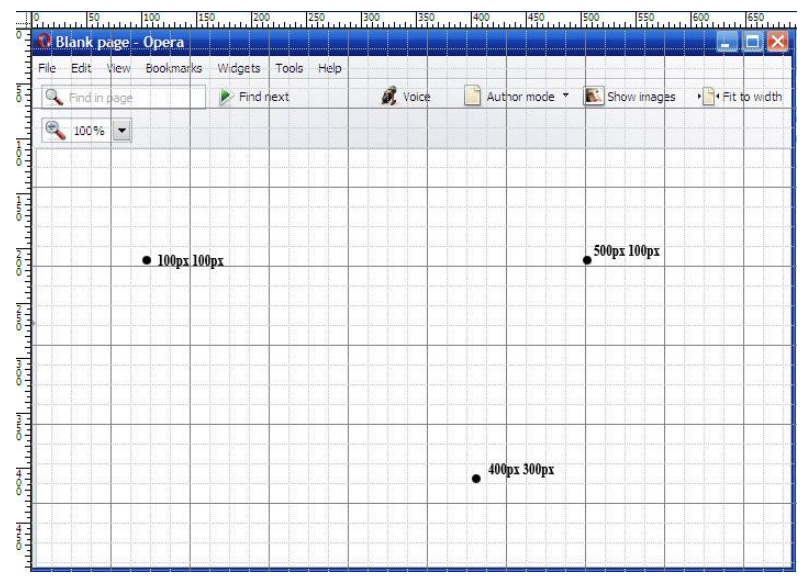
Đi cùng với thuộc tính float, trong CSS còn có một thuộc tính là clear. Thuộc tính clear là một thuộc tính thường được gán vào các phần tử liên quan tới phần tử đã được float để quyết định hướng xử sự của phần tử này.

Ở ví dụ trên, khi chúng ta float tấm ảnh qua trái thì mặc nhiên văn bản sẽ được tràn lên để lắp vào chỗ trống. Nhưng khi chúng ta đặt vào văn bản thuộc tính clear thì chúng ta có quyền quyết định xem phần văn bản đó có được tràn lên hay không.

Thuộc tính clear có tất cả 4 thuộc tính: left (tràn bên trái), right (tràn bên phải), both (không tràn) và none (mặc định).

***2.3.4 Position***

Nguyên lý hoạt động của position: Coi cửa sổ trình duyệt giống như một hệ tọa độ và với position chúng ta có thể đặt một đối tượng web ở bất cứ vị trí nào trên hệ tọa độ này

****

Giả sử chúng ta muốn định vị một ảnh ở vị trí 70px cách đỉnh và 90px từ bên trái tài liệu, chúng ta sẽ viết CSS như sau:

img { position:absolute; top:70px; left:90px }

*a) 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.

Ví dụ sau sẽ chỉ cho chúng ta cách đặt bốn ảnh ở bốn góc tài liệu bằng định vị tuyệt đối.

#logo1 {

position:absolute;

top:50px;

left:70px

}

#logo2 {

position:absolute;

top:0;

right:0

}

#logo3 {

position:absolute;

bottom:0;

left:0

}

#logo4 {

position:absolute;

bottom:70px;

right:50px

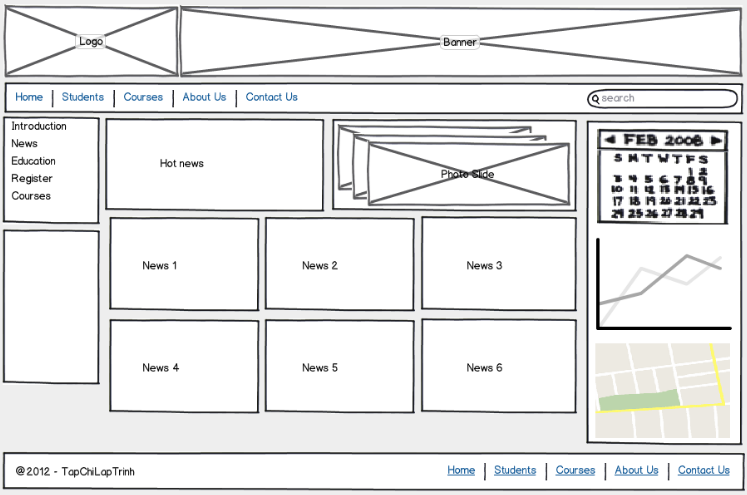
}

*b) Relative position*

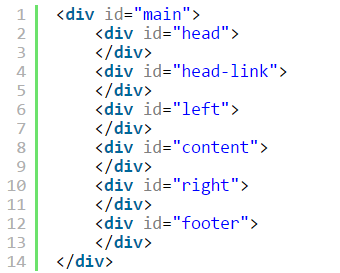
Sự định vị tương đối cho một thành phần là sự định vị được tính từ vị trí gốc trong tài liệu. Các thành phần đã được định vị tương đối sẽ để lại khoảng không trong tài liệu. Các thành phần được định vị tương đối sẽ nhận giá trị position là relative.

***2.3.5 Cách xây dựng bố cục một trang web***

Việc sử dụng div kết hợp với css để làm layout (bố cục) cho một trang web đã trở nên phổ biến và đạt hiệu quả cao. Ví dụ cần thiết kế một trang có giao diện được thiết kế layout như sau :



Trước hết tạo mới một trang HTML với cấu trúc chuẩn (gồm đầy đủ các phần html, head, title, body). Phần body của trang HTML:



Cặp thẻ trong đoạn mã trên tạo ra các phân vùng khác nhau trên trang HTML, mỗi phân vùng được đánh ID riêng biệt như sau:

m*ain:* phân vùng chứa toàn bộ nội dung của trang.

*head:* phân vùng chứa nội dung đầu tiên của trang (theo VD trên là phần logo + banner)

*head-link:* phân vùng chứa các liên kết đầu trang (ở VD trên: Home, Student, Course,…)

*left:* phân vùng chứa các nội dung bên trái (ở VD trên là các liên kết trái)

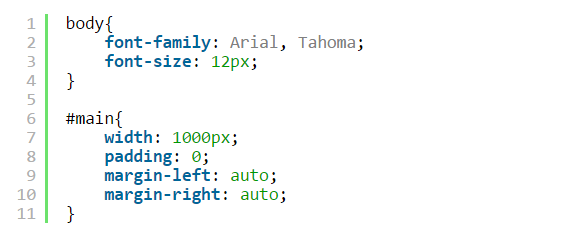
*content:* phân vùng chứa các nội dung chính của trang (ở VD trên là các phân vùng thông tin như: hot news, Photo Slide, News 1, …)

*right:* phân vùng chứa các thông tin bên phải trang (ở VD trên là lịch, biểu đồ, bản đồ)

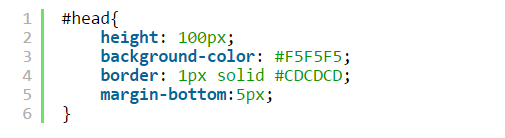
*footer:* phân vùng chứa các thông tin cuối trang (ở VD trên là thông tin về chủ sở hữu và các liên kết).

Tiếp đến là công đoạn code các bộ chọn css để phân vùng rõ ràng cho các div đã tạo ở trên:

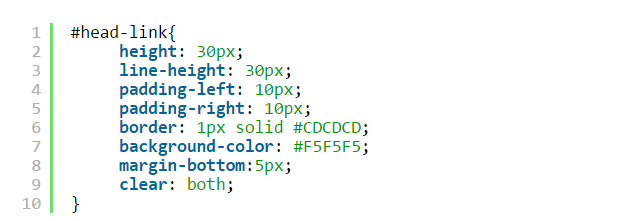
Tạo bộ chọn body để cố định font chữ cho cả trang và bộ chọn #main để cố định độ rộng vùng thông tin và căn chỉnh vùng này nằm giữa màn hình.



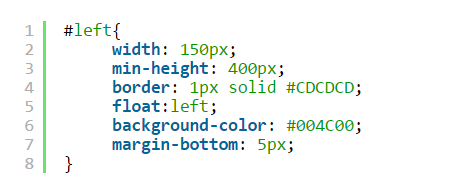
Tạo bộ chọn #head xác định chiều cao, màu nền, đường viền và khoảng cách so với vùng khác



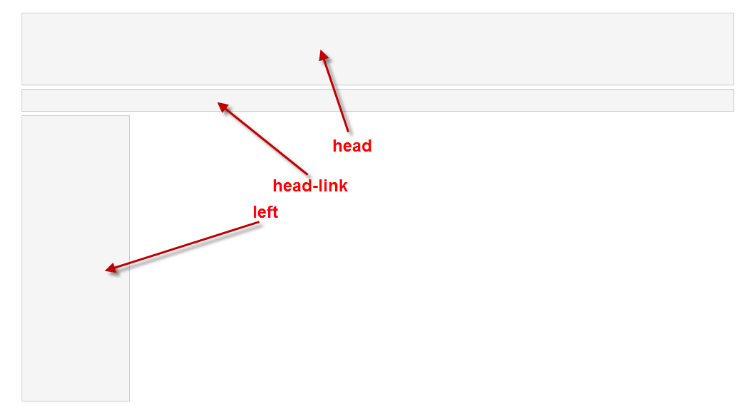
Tiếp tục với #head-link



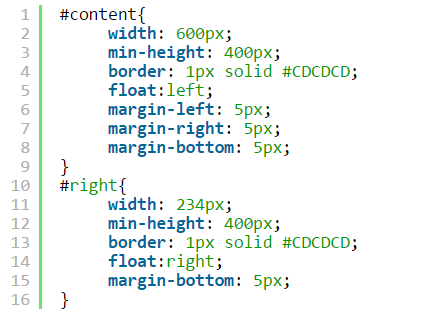
#left

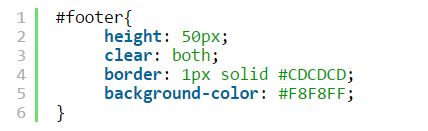


Kết quả trên trình duyệt:

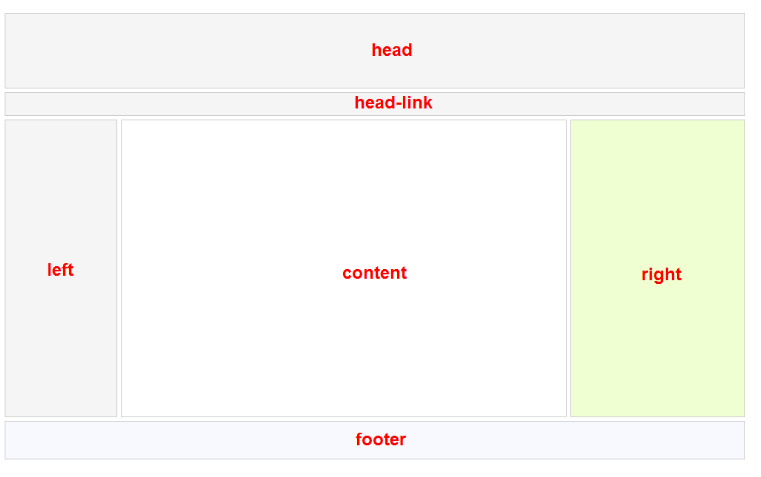


Tiếp đến #content, #right và #footer:





Quan sát lại trình duyệt:



**CHƯƠNG III - JAVASCRIPT**

**3.1. Tổng quan về JavaScript**

***3.1.1. Giới thiệu JavaScript***

Javascript là ngôn ngữ kịch bản phổ biến nhất trên thế giới hiện nay. Nó là ngôn ngữ chuẩn được sử dụng trong các trang web cũng như cho các ứng dụng desktop, mobile. Javascript được thiết kế thêm các tương tác tới các trang html.

Vậy thế nào là ngôn ngữ kịch bản? Ngôn ngữ kịch bản là ngôn ngữ lập trình nhẹ hỗ trợ việc viết kịch bản. Kịch bản ở đây là các dòng code có thể được thông dịch và thực thi nhanh chóng với các bước liên kết và biên dịch rõ ràng.

Javascript có thể làm những gì?

- Javascript cung cấp 1 công cụ lập trình cho các nhà thiết kế web html.

- Javascript có thể thao tác html (đọc và thay đổi nội dung của các thẻ html).

- Javascript có thể thao tác với CSS (đọc và thay đổi định dạng của các thẻ html).

- Javascript có thể xác định tính hợp lệ của dữ liệu (validate data).

- Javascript có thể lưu trữ dữ liệu.

- Javascript có thể thực hiện lại các sự kiện

Javascript và Java

Javascript và java là 2 ngôn ngữ hoàn toàn khác biệt cả về khái niệm và thiết kế. Java do Sun phát triển và là ngôn ngữ lập trình phức tạp giống như C và C++. Javascript là ngôn ngữ đơn giản được tạo ra bởi Brendan Eich ở Netscape, hiện được tiếp tục phát triển.

***3.1.2. Cách import và sử dụng JavaScript***

Nội dung của javascript được đặt trong cặp thẻ <script></script>.

Nội dung của javascript có thể được đặt trong thẻ <body> và phần <head> của trang web, hay có thể được đặt riêng ra 1 file .js

Ví dụ:

- *Nội dung của javascript được đặt trong phần <head> của trang html*:

<html>

<head>  
<script>  
function myFunction(){

document.getElementById("demo").innerHTML= "Xin chào";

}  
</script>

</head>

<body>

<h1>My Web Page</h1>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

</body>  
</html>

- *Nội dung của javascript được đặt trong phần <body> của trang html*:

<html>  
<body>

<h1>My Web Page</h1>

<p id="demo">A Paragraph</p>

<button type="button" onclick="myFunction()">Try it</button>

<script>  
function myFunction()  
{

document.getElementById("demo").innerHTML= "Xin chào";

}  
</script>

</body>  
</html>

*- Nội dung được lưu vào file .js và được nhúng vào trang html:*

<html>  
<body>  
<script src="myScript.js"></script>

</body>  
</html>

***3.1.3. Biến và đối tượng***

*✓ Khai báo không tường minh: ten\_bien = bieu\_thuc;*

*✓ Khai báo tường minh:*

* Khai báo biến và không khởi tạo giá trị: *var ten\_bien;*
* Khai báo biến và khởi tạo giá trị ban đầu: *var ten\_bien = bieu\_thuc;*

*Quy tắc đặt tên biến:*

* Ký tự bắt đầu phải là một chữ cái hoặc dấu gạch dưới “\_”
* Trong tên biến không chứa khoảng trắng và các kỹ tự đặc biệt
* Các biến phân biệt chữ hoa, chữ thường.

Ví dụ:

var answer1="He is called 'Johnny'";  
var answer2='He is called "Johnny"';

var pi=3.14;  
var x=123;  
var y=123e5;  
var z=123e-5;

var cars=new Array("Saab","Volvo","BMW");

var person={firstname:"John", lastname:"Doe", id:5566};

***3.1.4. Các toán tử***

- Gán (=)

Ví dụ: x = a+b

Kiểu gán thông thường Kiểu gán rút gọn

x = x+y x+=y

x = x-y x-=y

x = x/y x/=y

x = x%y x%=y

- So sánh

|  |  |
| --- | --- |
| Toán tử | Ý nghĩa |
| = = | So sánh bằng |
| != | So sánh không bằng |
| > | So sánh lớn hơn |
| >= | So sánh lớn hơn hoặc bằng |
| < | So sánh nhỏ hơn |
| <= | So sánh nhỏ hơn hoặc bằng |

- Số học

✓Cộng(+), trừ (-), nhân (\*), chia(/), lấy phần dư (%)

✓ bien++ (hoặc: ++bien) tăng giá trịcủa bien lên 1

✓bien - -(hoặc: - -bien) giảm giá trị của biến đi 1

- Chuỗi

Để nối chuỗi sử dụng toán tử “+”

Ví dụ: “Hà”+” ”+”Nội” được “Hà Nội”

- Logic

* Phép && (phép và)
* Phép || (phép hoặc)
* Phép ! (phủ định)

- Đối tượng

new: toán tử khai báo biến kiểu đối tượng

- Các toán tử trên bit (Bitwise)

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

& Toán tử bitwise AND, trả lại giá trị 1 nếu cả hai bit cùng là 1.

| Toán tử bitwise OR, trả lại giá trị 1 nếu một trong hai bit là 1.

^ Toán tử bitwise XOR, trả lại giá trị 1 nếu hai bit có giá trị khác nhau

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

<< Toán tử dịch trái. Dịch chuyển toán hạng trái sang trái một số lượng bit bằng toán hạng phải. Các bit bị chuyển sang trái bị mất và 0 thay vào phía bên phải. Ví dụ: 4<<2 trở thành 16 (số nhị phân 100 trở thành số nhị phân 10000)

>> Toán tử dịch phải. Dịch chuyển toán hạng trái sang phải một số lượng bit bằng toán hạng phải. Các bit bị chuyển sang phải bị mất và dấu của toán hạng bên trái được giữ nguyên. Ví dụ: 16>>2 trở thành 4 (số nhị phân 10000 trở thành số nhị phân 100)

>>> Toán tử dịch phải có chèn 0. Dịch chuyển toán hạng trái sang phải một số lượng bit bằng toán hạng phải. Bit dấu được dịch chuyển từ trái (giống >>). Những bit được dịch sang phải bị xoá đi. Ví dụ: -8>>>2 trở thành 1073741822 (bởi các bit dấu đã trở thành một phần của số). Tất nhiên với số dương kết quả của toán tử >> và >>> là giống nhau.

Có một số toán tử dịch chuyển bitwise rút gọn:

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

***3.1.5. Hàm và gọi hàm***

Cú pháp:

function ten\_ham(các tham số){

//các câu lệnh trong hàm

[return bieu\_thuc;] //kết thúc và trả về giá trị cho hàm

}

### Các hàm có sẵn

- eval(st)

- Number(st)

- parseInt(st, radius)

- parseFloat(st, radius)

***3.1.6. Các cấu trúc lệnh***

#### Câu lệnh rẽ nhánh

1. *Câu lệnh if*

Cú pháp:

if (dieu\_kien)

{

//các câu lệnh nếu dieu\_kien đúng;

}

else

{

//các câu lệnh nếu điều kiện sai;

}

*Ví dụ:*

var a=3;b=2;

var max;

if (a>b)

{

document.write("Số lớn là: " + a);

}

else

{

document.write("Số lớn là: " + b);

}

1. *Câu lệnh lựa chọn switch*Cú pháp:

switch (bieu\_thuc){

case gia\_tri\_1:

//mã lệnh thực hiện khi bieu\_thuc= =gia\_tri\_1

break;

case gia\_tri\_2:

//mã lệnh thực hiện khi bieu\_thuc= =gia\_tri\_2

break;

…….

case gia\_tri\_n:

//mã lệnh thực hiện khi bieu\_thuc= =gia\_tri\_n

break;

default:

//mã lệnh thực hiện khi không có giá trị nào phù hợp

}

#### Câu lệnh lặp

1. *Câu lệnh for (số lần lặp xác định trước)*

Cú pháp:

for (khởi tạo biến đếm; điều kiện; thay đổi giá trị biến đếm){

//các câu lệnh thực hiện trong khi lặp

}

*Ví dụ:*

for(i=1;i<=10;i++)

{

document.writeln(i+"<br>");

}

1. *Câu lệnh lặp while (số lần lặp không xác định trước)*

Cú pháp:

while (dieu\_kien){

//các câu lệnh thực hiện trong khi lặp

}

Ví dụ:

var i=1;

while(i<=10)

{

document.write(i +"<br>");

i++;

}

***3.1.7. Các đối tượng cơ bản***

Window

Textaear

Text

FileUpload

Password

Hidden

Submit

Reset

Radio

Checkbox

Button

Select

Plugin

Mime Type

Frame

document

Location

History

Layer

Link

Image

Area

Anchor

Applet

Plugin

Form

navigator

Option

*Sơ đồ phân cấp đối tượng Navigator*

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

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

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

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

#### Đối tượng navigator

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

*Các thuộc tính*

|  |  |
| --- | --- |
| *appCodeName* | Xác định tên mã nội tại của trình duyệt (Atlas). |
| *AppName* | Xác định tên trình duyệt. |
| *AppVersion* | Xác định thông tin về phiên bản của đối tượng navigator. |
| *userAgent* | Xác định header của user - agent. |

*Ví dụ*

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

<HTML>

<HEAD>

<TITLE> Navigator Object Exemple </TITLE>

<SCRIPT LANGUAGE= "JavaScript">

document.write("appCodeName = "+navigator.appCodeName + "<BR>");

document.write("appName = "+navigator.appName + "<BR>");

document.write("appVersion = "+navigator.appVersion + "<BR>");

document.write("userAgent = "+navigator.userAgent + "<BR>");

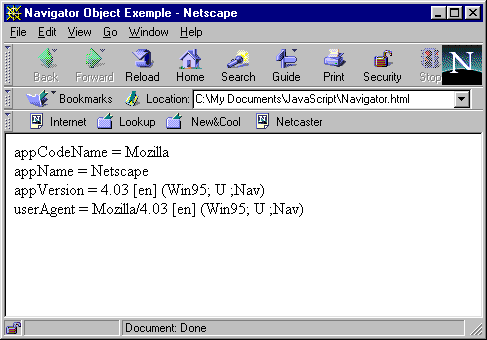
</SCRIPT>

</HEAD>

<BODY>

</BODY>

</HTML>



\

#### Đối tượng window

Đối tượng window như đã nói ở trên là đối tượng ở mức cao nhất. Các đối tượng document, frame, vị trớ đều là thuộc tính của đối tượng window.

CÁC THUỘC TÍNH

* defaultStatus - Thông báo ngầm định hiển thị lên trên thanh trạng thái của cửa sổ
* Frames - Mảng xác định tất cả các frame trong cửa sổ.
* Length - Số lượng các frame trong cửa sổ cha mẹ.
* Name - Tên của cửa sổ hiện thời.
* Parent - Đối tượng cửa sổ cha mẹ
* Self - Cửa sổ hiện thời.
* Status - Được sử dụng cho thông báo tạm thời hiển thị lên trên thanh thạng thái cửa sổ. Đựơc sử dụng để lấy hay đặt lại thông báo trạng thái và ghi đè lên defaultStatus.
* Top - Cửa sổ ở trên cùng.
* Window - Cửa sổ hiện thời.

CÁC PHƯƠNG THỨC

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

Ví dụ: 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 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 dựng để đóng của 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')">

<P>

<A HREF="doc2.html" TARGET="window2">

Load a file into window2 </A>

</P>

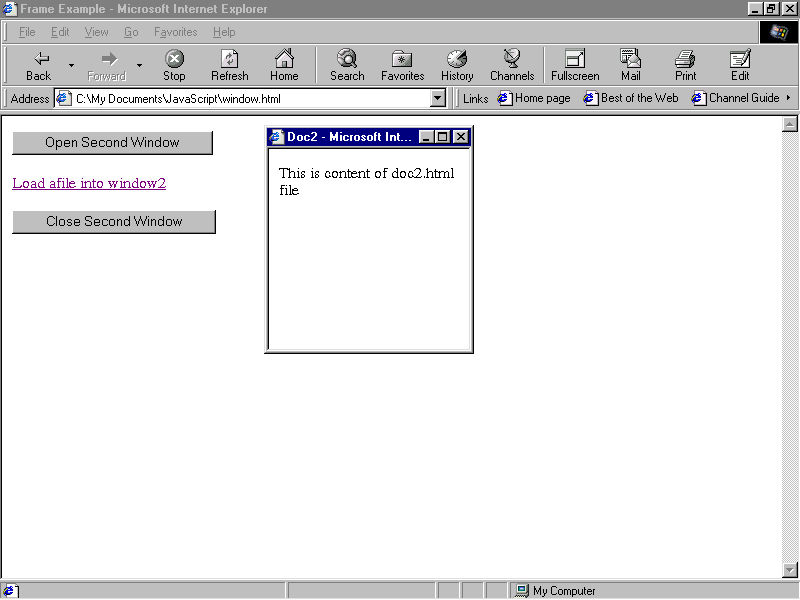
<INPUT TYPE="button" VALUE="Close Second Window"

onClick="msgWindow.close()">

</FORM>

</BODY>

</HTML>



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.

#### Đối tượng location

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

http:// [www.abc.com/](http://www.abc.com/) chap1/page2.html#topic3

*Các thuộc tính*

* hash - Tên anchor của vị trí hiện thời (ví dụ topic3).
* Host - Phần hostname:port của URL (ví dụ [www.abc.com](http://www.abc.com)) ). Chú ý rằng đây thường là cổng ngầm định và ít khi được chỉ ra.
* 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).
* Port - Cổng truyền thông được sử dụng cho máy tính host, thường là cổng ngầm định.
* Protocol - Giao thức được sử dụng (cùng với dấu hai chấm) (ví dụ http:).
* Search - Câu truy vấn tìm kiếm có thể ở cuối URL cho các script CGI.

#### Đối tượng frame

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

CÁC THUỘC TÍNH

* + frames - Mảng tất cả các frame trong cửa sổ.
  + Name - Thuộc tính NAME của thẻ <FRAME>
  + Length - Số lượng các frame con trong một frame.
  + Parent - Cửa sổ hay frame chứa nhóm frame hiện thời.
  + self - frame hiện thời.
  + Window - frame hiện thời.

CÁC PHƯƠNG THỨC

* + clearTimeout (timeoutID) - Xoá timeout do setTimeout lập. SetTimeout trả lại timeoutID.
  + TimeoutID = setTimeout (expression,msec) - Đánh giá expression sau khi hết thời gian msec.

SỬ DỤNG FRAME

* Tạo một frame (create)

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

*Ví dụ1:* tạo frame ( hình 17)

<HTML>

<HEAD>

<TITLE>Frame Example </TITLE>

<FRAMESET ROWS="90%,10%">

<FRAMESET COLS="30%,70%">

<FRAME SRC=CATEGORY.HTM NAME="ListFrame">

<FRAME SRC=TITLES.HTM NAME="contentFrame">

</FRAMESET >

<FRAME SRC=NAVIGATOR.HTM NAME="navigateFrame">

</FRAMESET >

</HEAD>

<BODY></BODY>

</HTML>

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

Top

listFrame (category.html)

contentFrame (titles.html)

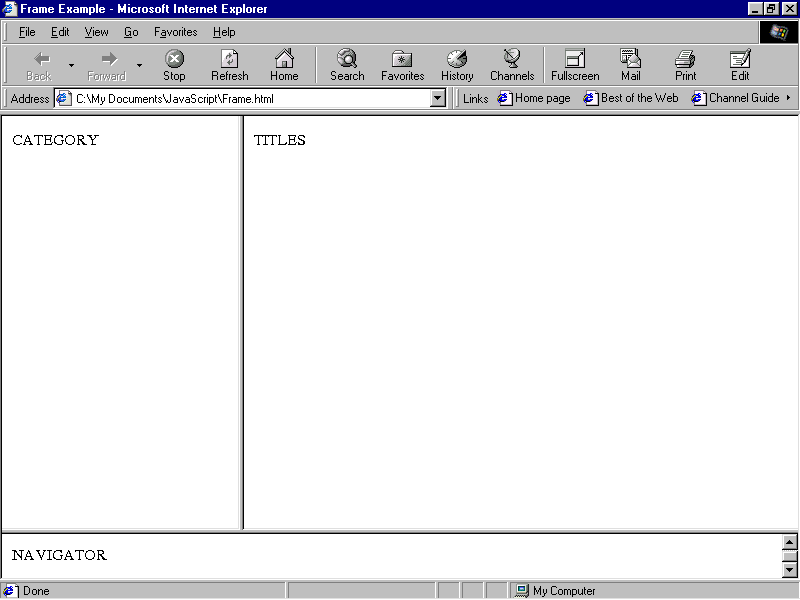
navigatorFrame (navigator.html)

Chúng ta có thể gọi tới những frame trước đó bằng cách sử dụng thuộc tính frames như sau:

listFrame chính là top.frames[0]

contentFrame chính là top.frames[1]

navigatorFrame chính là top.frames[2]



*Ví dụ 2*: Cũng giống như một sự lựa chọn, có thể tạo ra một cửa sổ giống như ví dụ trước nhưng trong mỗi đỉnh của hai frame lại có một cửa sổ cha riêng từ navigateFrame. Mức frameset cao nhất có thể được định nghĩa như sau:

<HTML>

<HEAD>

<TITLE>Frame Example </TITLE>

<FRAMESET ROWS="90%,10%">

<FRAME SRC=muske13.HTML NAME="upperFrame">

<FRAME SRC=NAVIGATOR.HTM NAME="navigateFrame">

</FRAMESET >

</HEAD>

<BODY>

</BODY>

</HTML>

Trong file muske13.html lại tiếp tục đặt một frameset:

<HTML>

<HEAD>

<TITLE>Frame Example </TITLE>

<FRAMESET COLS="30%,70%">

<FRAME SRC=CATEGORY.HTM NAME="ListFrame">

<FRAME SRC=TITLES.HTM NAME="contentFrame">

</FRAMESET >

</HEAD>

<BODY>

</BODY>

</HTML>

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

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

top

upperFrame (muske13.html)

navigatorFrame (navigator.html)

listFrame (category.html)

contentFrame (titles.html)

upperFrame chính làtop.frames[0]

navigatorFrame chính là top.frames[1]

listFrame chính là upperFrame.frames[0]

hoặctop.frames[0].frames[0]

contentFrame chính là upperFrame.frames[1]

hoặctop.frames[0].frames[1]

* Cập nhật một frame (update)

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

Trong ví dụ trên, nếu thêm một dòng sau vào navigatorFrame:

<INPUT TYPE="button" VALUE="Titles only" onClick="top.frames[0].location='artist.html'">

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

#### Đối tượng DOCUMENT

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

Các 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

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

#### Đối tượng anchors

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

document.anchors [index]

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

document.anchors.length.

#### Đối tượng FORMS

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

button

checkbox

hidden

password

radio

reset

select

submit

text

textarea

Các phần tử này sẽ được trình bày sau.

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

document.forms.length.

Mỗi một form có thể được tham chiếu như sau:

document.forms[index]

CÁC THUỘC TÍNH

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

CÁC PHƯƠNG THỨC

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

CÁC CHƯƠNG TRÌNH XỬ LÝ SỰ KIỆN

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

Ví dụ:

<!DOCTYPE html>

<head>

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

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

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

document.forms['adduser'].name.focus();

function set\_focus()

{

document.forms['adduser'].name.focus();

}

function kt()

{

var frm = window.document.adduser;

if(frm.user.value=='')

{

alert('Xin vui lòng nhập Username !');

document.forms['adduser'].user.focus();

return false;

}

else if(frm.pass.value=='')

{

alert('Xin vui lòng nhập Password !');

document.forms['adduser'].pass.focus();

return false;

}

else

return true;

}

</script>

<title>Untitled Document</title>

</head>

<body>

<form name="adduser" action="form1.html" method="post" onsubmit="return kt()">

<fieldset>

<legend>Member Register</legend>

<label>User Name:</label> <input type="text" name="user" size="25" /><br />

<label>Password:</label> <input type="password" name="pass" size="25" /><br />

<label>Re-Password:</label> <input type="text" name="re\_pass" size="25" /><br />

<label>E-Mail:</label> <input type="text" name="email" size="25" /><br />

<label>Address :</label> <input type="text" name="address" size="25" /><br />

<label></label> <input type="submit" name="ok" value="Add New" />

</fieldset>

</form>

</body>

</html>

#### Đối tượng HISTORY

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

CÁC THUỘC TÍNH

* length - Số lượng các URL trong đối tượng.

CÁC PHƯƠNG THỨC

* history.back() - Được sử dụng để tham chiếu tới URL mới được thăm trước đây.
* history.forward() - Được sử dụng để tham chiếu tới URL kế tiếp trong danh sách. Nó sẽ không gây hiệu ứng gì nếu đã đến cuối của danh sách.
* history.go (delta | "location") - Được sử dụng để chuyển lên hay chuyển xuống delta bậc hay di chuển đến URL xác định bởi location trong danh sách. Nếu delta được sử dụng thì việc dịch chuyển lên phía trên khi delta dương và xuống phía dưới khi delta âm. nếu sử dụng location, URL gần nhất có chứa location là chuỗi con sẽ được tham chiếu.

#### Đối tượng LINKS

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

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

document.links.length()

Có thể tham chiếu tới một liên kết qủa

document.links [index]

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

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

CÁC THUỘC TÍNH

* hash - Tên anchor của vị trí hiện thời (ví dụ topic3).
* Host - Phần hostname:port của URL (ví dụ [www.abc.com)](http://www.abc.com)). Chú ý rằng đây thường là cổng ngầm định và ít khi được chỉ ra.
* Hostname - Tên của host và domain (ví dụ ww.abc.com).
* href - Toàn bộ URL cho document hiện tại.
* Pathname - Phần đường dẫn của URL (ví dụ /chap1/page2.html).
* port - Cổng truyền thông được sử dụng cho máy tính host, thường là cổng ngầm định.
* Protocol - Giao thức được sử dụng (cùng với dấu hai chấm) (ví dụ http:).
* Search - Câu truy vấn tìm kiếm có thể ở cuối URL cho các script CGI.
* Target - Giống thuộc tính TARGET của <LINK>.

CÁC CHƯƠNG TRÌNH XỬ LÝ SỰ KIỆN

* onClick - Xảy ra khi người sử dụng nhấn vào link.
* onMouseOver - Xảy ra khi con chuột di chuyển qua link.

#### Đối tượng MATH

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

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

CÁC THUỘC TÍNH

* E - Hằng số Euler, khoảng 2,718.
* LN2 - logarit tự nhiên của 2, khoảng 0,693.
* LN10 - logarit tự nhiên của 10, khoảng 2,302.
* LOG2E - logarit cơ số 2 của e, khoảng 1,442.
* PI - Giá trị của π, 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 nămg giữa -1 và 1.
* Math.asin (number) - Trả lại giá trị arc sine (theo radian) của number. Giá trị của number phải nămg giữa -1 và 1.
* Math.atan (number) - Trả lại giá trị arc tan (theo radian) của number.
* Math.ceil (number) - Trả lại số nguyên nhỏ nhất lớn hơn hoặc bằng number.
* Math.cos (number) - Trả lại giá trị cosine của number.
* Math.exp (number) - Trả lại giá trị e^ number, với e là hằng số Euler.
* Math.floor (number) - Trả lại số nguyên lớn nhất nhỏ hơn hoặc bằng number.
* Math.log (number) - Trả lại logarit tự nhiên của number.
* Math.max (num1,num2) - Trả lại giá trị lớn nhất giữa num1 và num2
* Math.min (num1,num2) - Trả lại giá trị nhỏ nhất giữa num1 và num2.
* Math.pos (base,exponent) - Trả lại giá trị base luỹ thừa exponent.
* Math.random (r) - Trả lại một số ngẫu nhiên giữa 0 và 1. Phwong thức này chỉ thực hiện được trên nền tảng UNIX.
* Math.round (number) - Trả lại giá trị của number làm tròn tới số nguyên gần nhất.
* Math.sin (number) - Trả lại sin của number.
* Math.sqrt (number) - Trả lại căn bậc 2 của number.
* Math.tan (number) - Trả lại tag của number.

#### 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 chuyểnbằng phút của giờ địa phương hiện tại so với giờ quốc tế GMT.
* dateVar.getYear()-Trả lại năm cho dateVar.
* Date.parse (dateStr) - Phân tích chuỗi dateStr và trả lại số lượng các mili giây tính từ 00:00:00 ngày 01/01/1970.
* dateVar.setDay(day) - Đặt ngày trong tháng là day cho dateVar.
* dateVar.setHours(hours) - Đặt giờ là hours cho dateVar.
* dateVar.setMinutes(minutes) - Đặt phút là minutes cho dateVar.
* dateVar.setMonths(months) - Đặt tháng là months cho dateVar.
* dateVar.setSeconds(seconds) - Đặt giây là seconds cho dateVar.
* dateVar.setTime(value) - Đặt thời gian là value, trong đó value biểu diễn số lượng mili giây từ 00:00:00 ngày 01/01/10970.
* 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.

#### 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.index0f(srchStr [,index]) - Trả lại vị trí trong chuỗi str vị trí xuất hiện đầu tiên của chuỗi srchStr. Chuỗi str được tìm từ trái sang phải. Tham số index có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.
* str.italics() - Kết quả giống như thẻ <I> trên chuỗi str.

str.lastIndex0f(srchStr [,index]) - Trả lại vị trí trong chuỗi str vị trí xuất hiện cuối cùng của chuỗi srchStr. Chuỗi str được tìm từ phải sang trái. Tham số index có thể được sử dụng để xác định vị trí bắt đầu tìm kiếm trong chuỗi.

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

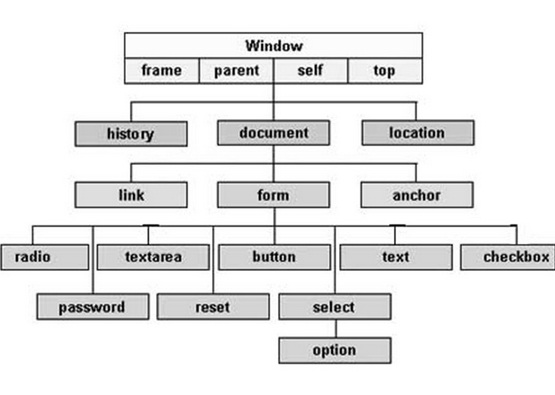
**3.2. Lập trình DOM và quản lý của sổ sử dụng Javascript**

***3.2.1. Giới thiệu mô hình DOM***

DOM (Document Object Model): Là một mô hình chuẩn cho phép ngôn ngữ lập trình có thể truy xuất và thay đổi động nội dung, cấu trúc, định dạng của 1 văn bản

Mỗi trang web bên trong một trình duyệt đều được coi là một đối tượng. Khi một trang web được tải, trình duyệt sẽ tạo ra một mô hình đối tượng về trang web (DOM – Document Object Model). Nói một cách khác, DOM đại diện cho những gì chứa trong trang web và cách mà trang web được truy cập và sửa đổi.

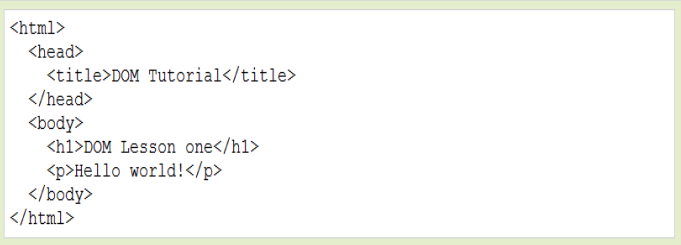
***3.2.2. DOM HTML***



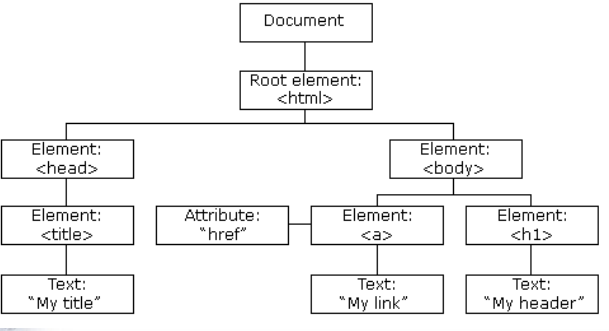
HTML DOM : là mô hình chuẩn cho văn bản HTML

* Toàn bộ trang là document node
* Mỗi thẻ là 1 HTML node
* Văn bản trong 1 thẻ là text node
* Các thuộc tính trong thẻ là các attribute
* Thông qua HTML DOM, ta có thể lấy, thay đổi, thêm hay xóa bất kỳ 1 element nào trong trang web

►Ví dụ HTML DOM



HTML DOM coi trang html là 1 cây



*► Quan hệ giữa các node*

Các node trong cây HTML có mối quan hệ phân cấp với nhau

* Các từ cha, con, anh em dùng để mô tả các mối quan hệ này. Các node cha có các node con, các node anh em là các node có cùng cấp
* Trong 1 cây HTML, node trên cùng là root (gốc)
* Mọi node (trừ root) đề có duy nhất 1 node cha
* 1 node có thể có nhiều node con
* Node lá là node không có node con
* Node anh em là node có cùng node cha

*► Lấy 1 node*

Mỗi node trong HTML là 1 đối tượng. Ta có thể lấy đối tượng bằng cách :

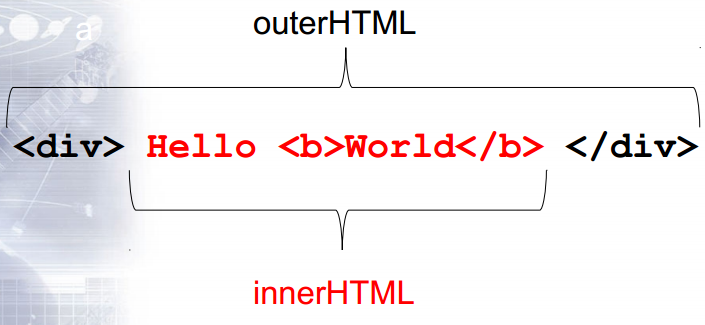
* X.getElementById ( id) : lấy element có id cung cấp trong node X
* X.getElementByTagName( name) : lấy danh sách các element có name cung cấp trong node X

*► Thuộc tính 1 node*

* X.innerHTML : văn bản trong X
* X.nodeName : tên của X
* X.nodeValue : giá trị của X
* X.parentNode : node cha của X
* X.attributes : các thuộc tính của X
* X.firstChild : node con đầu tiên của X
* X.lastChild : node con cuối của X

Trong đó X là 1 node trong HTML DOM

►innerHTML và outerHTML



►*Thêm, xóa 1 node*

* X.appendChild ( Y) : thêm node Y vào làm con node X
* X.removeChild ( Y ) : xóa node Y ra khỏi con node X
* document.createTextNode(“Text”) : tạo 1 nút văn bản từ nút gốc
* document.createElement(TagName) : tạo 1 nút phần tử từ nút gốc

► *Làm việc với thuộc tính*

Ta có thể thêm, xóa, cập nhật thuộc tính của 1 node như sau :

* X.getAttribute(“Attribute name”) : Lấy giá trị 1 attribute
* X.setAttribute(“Attribute name”,”Attribute value”) : Thêm hay set lại giá

trị 1 attribute

* X.removeAttribute(“Attribute name”) : Xóa 1 attribute



► *Định dạng node*

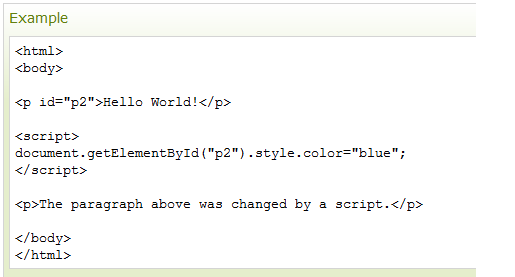
* Các node biểu diễn 1 thẻ đều có thuộc tính style dùng để định dạng
* Cách thay đổi định dạng 1 node :

***3.2.3. DOM CSS***

Cây phần tử HTML DOM cho phép javascript thay đổi định dạng của các phần tử HTML.Để thay đổi định dạng của phần tử HTML sử dụng cú pháp:



Ví dụ sau thay đổi các định dạng của phần tử <p>



Sử dụng các sự kiện (Events)

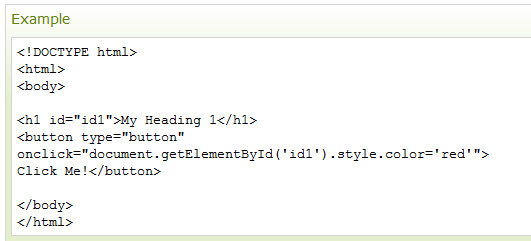
Cây phần tử HTML DOM cho phép thực thi mã lệnh khi một sự kiện sảy ra. Các sự kiện có thể là:

● Một phần tử được click

● Trang web được tải lên

● Các trường <input> được thay đổi

Ví dụ sau sẽ thay đổi định dạng của 1 phần tử HTML với id=”id1”, khi người dùng click vào nút.

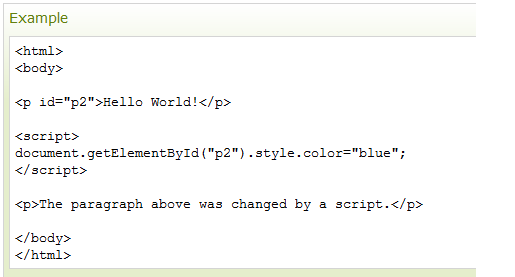


***3.2.4. DOM event***

Cây phần tử HTML DOM cho phép javascript thay đổi định dạng của các phần tử HTML.Để thay đổi định dạng của phần tử HTML sử dụng cú pháp:



Ví dụ sau thay đổi các định dạng của phần tử <p>



Sử dụng các sự kiện (Events)

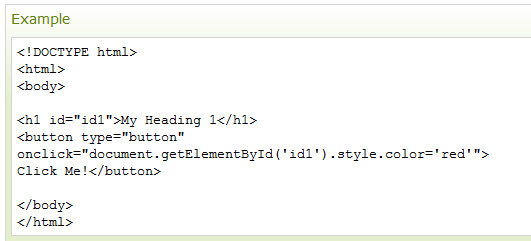
Cây phần tử HTML DOM cho phép thực thi mã lệnh khi một sự kiện sảy ra. Các sự kiện có thể là:

● Một phần tử được click

● Trang web được tải lên

● Các trường <input> được thay đổi

Ví dụ sau sẽ thay đổi định dạng của 1 phần tử HTML với id=”id1”, khi người dùng click vào nút.



***3.2.5. DOM notes***

Trong HTML DOM (Document Object Model) mọi thứ đều được coi là các nút (node):

● Tài liệu bản thân nó là 1 node tài liệu

● Tất cả các phần tử HTML là các node phần tử

● Tất cả các thuộc tính HTML là các node thuộc tính

● Văn bản bên trong các phần tử HTML là các node văn bản

● Các chú thích là các node chú thích

**4.3 Thư viện jQuery**

***4.3.1 Giới thiệu về jQuery***

jQuery là một Javascript Framework, hỗ trợ các nhà lập trình web tạo ra các tương tác trên website một cách nhanh nhất. jQuery được khởi xướng bởi John Resig (hiện là trưởng dự án của Mozzila) vào năm 2006, jQuery có mã nguồn mở và hoàn toàn miễn phí. jQuery có một cộng đồng sử dụng đông đảo và được rất nhiều lập trình tham gia hoàn thiện, phát triển và viết Plugin.

**jQuery** đơn giản hóa cách viết Javascript và tăng tốc độ xử lý các sự kiện trên trang web, giúp tiết kiệm thời gian và công sức so với cách viết javascript thông thường.

Bên cạnh đó, việc sử dụng **jQuery** giúp giải quyết tốt vấn đề xung đột giữa các trình duyệt web.

Học**jQuery** rất đơn giản, nếu như nắm vững CSS, chúng ta có thể tiếp cận và sử dụng jQuery nhanh chóng.

* Hỗ trợ tốt việc xử lí các vấn đề thường gặp **(DOM, AJAX…)**
* Tương thích nhiều trình duyệt web phổ biến.
* Nhỏ gọn, dễ dùng, có nhiều tài liệu hướng dẫn sử dụng chi tiết.
* Ít xung khắc với các thư viện Javascript khác.
* Plugin phong phú.

***Hướng tới các thành phần trong HTML*:** nếu không sử dụng thư viện Javascript này, chúng ta phải viết rất nhiều dòng code mới có thể đạt được mục tiêu là di chuyển trong cấu trúc cây (Document Object Model) của một tài liệu **HTML** và chọn ra các thành phần liên quan. **jQuery** cho phép chọn bất cứ thành phần nào của tài liệu một cách dễ dàng dựa vào [jQuery selector](http://www.ewebvn.com/tag/jquery-selector).

***Thay đổi giao diện của một trang web***: CSS là công cụ rất mạnh để định dạng một trang web nhưng nó có một nhược điểm là không phải tất cả các trình duyệt đều hiển thị giống nhau. Cho nên **jQuery** ra đời để lấp chỗ trống này, vì vậy có thể sử dụng nó để giúp trang web có thể hiển thị tốt trên hầu hết các trình duyệt. Hơn nữa **jQuery** cũng có thể thay đổi class hoặc những định dạng **CSS** đã được áp dụng lên bất cứ thành phần nào của **HTML** ngay cả khi trang web đó đã được trình duyệt load thành công.

***Thay đổi nội dung của tài liệu*:** jQuery không chỉ  thay đổi bề ngoài của trang web, mà còn  có thể thay đổi nội dung của chính tài liệu đó. Nó có thể thêm hoặc bớt nội dung trên trang, thậm chí cả cấu trúc **HTML** của một trang web cũng có thể được viết lại và mở rộng.

***Tương tác với người dùng*: jQuery** cho nhiều phương thức để tương tác với người dùng và tối giản các mã Event trong code HTML.

***Tạo hiệu ứng động*: jQuery** cho phép sử dụng rất nhiều hiệu ứng động như mờ dần, slideUp, slideDown()…

***Hỗ trợ Ajax*:** đây chính là công nghệ ngày càng trở nên phổ biến Asynchronous JavaScript And XML (**AJAX**), nó giúp người thiết kế web tạo ra những trang web tương tác cực tốt và nhiều tính năng. Thư viện **jQuery** loại bỏ sự phức tạp của trình duyệt trong quá trình này và cho phép người [phát triển web](http://www.ewebvn.com/phat-trien-web)có thể tập trung vào các tính năng đầu cuối, đơn giản hoá các tác vụ javascript.

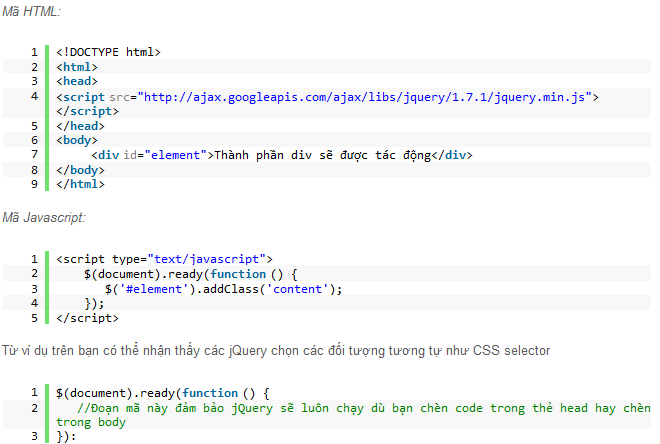
***3.3.2 Đưa jQuery vào trang web***

Để sử dụng jQuery phải có thư viện do jQuery cung cấp bằng cách truy cập vào [http://jquery.com](http://jquery.com/) để Download phiên bản mới. Hoặc có thể sử dụng trực tiếp jQuery lưu trữ tại jQuery hay Google từ 2 địa chỉ sau:

<http://code.jquery.com/jquery-1.7.2.min.js>

<http://ajax.googleapis.com/ajax/libs/jquery/1.7.1/jquery.min.js>

Ví dụ sau sẽ tìm trong tài liệu HTML thành phần div có id là element và thêm vào thành phần này 1  class content

****

***Cú pháp cơ bản của jQuery:***

*Cú pháp để chọn đối tượng*

jQuery sử dụng cùng một cách như CSS để lựa chọn những phần tử HTML (thẻ HTML) và thực hiện vài việc gì đó lên những đối tượng được chọn. Việc chọn đối tượng có thể thực hiện theo kiểu chọn một đối tượng duy nhất (đơn lẻ) hoặc một tập hợp các đối tượng có cùng đặc điểm( nhiều thành phần). Nguyên tắc để chọn cụ thể như sau:

Cú pháp: **$(doi\_tuong)**

$ : Ký tự bắt đầu của jQuery trong việc xác định đối tượng

doi\_tuong: tiêu chí để xác định đối tượng. Liệt kê cụ thể những trường hợp như bên dưới.

|  |  |
| --- | --- |
| $(this) | Chọn đối tượng hiện tại (đối tượng đặc biệt) | đối tượng đơn lẻ |
| $(document) | Chọn đối tượng toàn bộ tài liệu |
| $(window) | Chọn đối tượng về cửa sổ hiện hành |
| $(‘p’) | Chọn tất cả các thẻ <p> |
| $(‘.ClassName’) | Chọn tất cả các thẻ HTML có class=”ClassName” |
| $(‘#IdValue’) | Chọn thẻ HTML có id=”IdValue” |

*Điều khiển css của đối tượng*

■ Lấy giá trị của một thuộc tính

Khi dùng jQuery, việc tác động trực tiếp và lấy giá trị một thuộc tính CSS của đối tượng đang được chọn vô cùng đơn giản vì Chúng ta sẽ sử dụng tên thuộc tính và giá trị lấy được sẽ có định dạng y như khi định nghĩa CSS thuần.

**Cú pháp: $(doi\_tuong).css(‘ten\_css’);**

Doi\_tuong:là tiêu chuẩn xác định đối tượng như mô tả ở trên

Ten\_css:là tên của thuộc tính css như font-size, color, background-color,...

Ví dụ: để lấy giá trị thuộc tính CSS font-familycủa tất cả các thẻ <p> ta có câu lệnh jQuery sau:

var value= $(‘p’).css(‘font-family’);

Giá trị trả về của câu lệnh trên là chuỗi tên của font đang dùng như “Verdana” hay “Arial”. Chú ý: Trong trường hợp thẻ <p> trong toàn trang có nhiều font-family khác nhau thì giá trị trả về là của thẻ <p> xuất hiện đầu tiên.

■ Đặt giá trị mới cho CSS

Tương tự như lấy giá trị của thuộc tính CSS, việc gán giá trị cho nó cũng khá đơn giản.

**Cú pháp: $(doi\_tuong).css(‘ten\_css’, ‘gia\_tri’);**

doi\_tuong:là tiêu chuẩn xác định đối tượng như mô tả ở trên

ten\_css:là tên của thuộc tính css như: font-size, color, background-color,...

gia\_tri:là giá trị muốn gán cho thuộc tính css

Ví dụ: muốn gán giá trị thuộc tính font-family thành Verdana cho toàn bộ thẻ <p>. Câu lệnh jQuery sẽ là:

$(‘p’).css(‘font-family’,’Verdana’);

Câu lệnh trên sẽ thay thế toàn bộ thuộc tính font-family của tất cả các thẻ <p> thành Verdana

*Điều khiển thuộc tính của đối tượng*

■ Lấy giá trị thuộc tính của đối tượng

*Cú pháp:* $(doi\_tuong).attr(‘thuoc\_tinh’);

*doi\_tuong:*tiêu chuẩn xác định đối tượng như mô tả ở trên

*thuoc\_tinh:*tên của thuộc tính của thẻ html

*Ví dụ1: ta có thẻ* <img id=”vidu\_image” src=”simple.jpg” /> *Nếu muốn lấy thuộc tính src của thẻ này ta dùng:*

var value = $(‘#vidu\_image’).attr(‘src’);

*Ví dụ 2: có thẻ* <input type=’txtUID’ id=’vidu’/>, *nếu muốn lấy giá trị thuộc tính value của thẻ này ta sử dụng câu lệnh sau:*

var value = $(‘#txtUID’).attr(‘value’);

*Hoặc:* var value=$(‘#txtUID’).val();

*Chú ý: đối với các thẻ nhập dữ liệu của FORM người ta thường hay dùng hàm val() để lấy giá trị của thuộc tính value của chúng. Không có sự khác biệt nào về kết quả giữa cách dùng attr() và val()*

■ Gán giá trị mới cho thuộc tính

Cú pháp: **$(doi\_tuong).attr(‘thuoc\_tinh’,’gia\_tri’);**

*doi\_tuong:*tiêu chuẩn xác định đối tượng như mô tả ở trên

*thuoc\_tinh:*tên của thuộc tính của thẻ html

*gia\_tri:*giá trị muốn gán cho thuộc tính

Ví dụ: ta có thẻ <img id=”vidu\_image” src=”simple.jpg” /> ta muốn đổi thuộc tính src thành new\_image.jpg thực hiện như sau:

$(‘#vidu\_image’).attr(‘src’,’new\_image.jpg’);

Gán sự kiện và xử lý sự kiện

**►** Gán sự kiện vào đối tượng

*Cú pháp*: **$(doi\_tuong).ten\_su\_kien(ham\_dieu\_khien);**

*doi\_tuong:*tiêu chuẩn xác định đối tượng như mô tả ở trên

*ten\_su\_kien:*Tên của sự kiện

*ham\_dieu\_khien:*hàm điều khiển của sự kiện khi sự kiện phát sinh

*Ví dụ1:*  Khi muốn website load lên sẽ hiển thị hộp thoại thông báo dòng chữ “Xin chào”.

Code javascript thông thường: <body onload=”alert(‘Xin chào’)”>

Code sử dụng jQuery: $(document).ready(function(){alert(‘Xin chào’);});

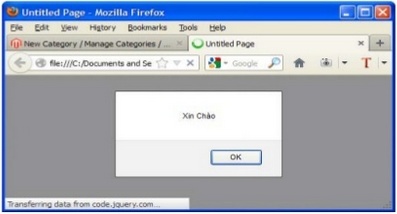
*Ví dụ 2: sử dụng hàm điều khiển định nghĩa riêng (không có tham số)*

function Display\_Message(){

Alert(“Xin chào”);

}

$(document).ready(Display\_Message);

**

*Ví dụ 3: Sử dụng hàm điều khiển định nghĩa riêng (có tham số)*

function Display\_Message(thong\_diep){

alert(thong\_diep);

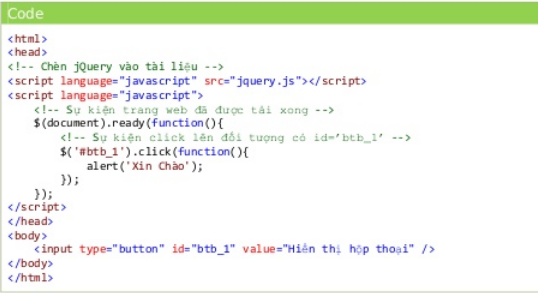
}

$(document).ready(Display\_Message(‘Xin chào’));

► Hủy sự kiện đã gán vào đối tượng

Cú pháp: **$(doi\_tuong).unbind(‘ten\_su\_kien’);**

Ví dụ 1: Gán sự kiện click lên 1 button, khi người dùng click vào button thì bật hộp thoại có dòng chữ “Xin chào ” lên.

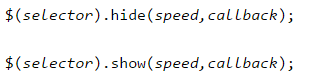


Sự kiện cho nút btb\_1 nằm trong sự kiện ready của document vì: khi trang web được tải xong thì đối tượng btb\_1 mới tồn tại. Nếu không đặt lệnh gán sự kiện cho btb\_1 bên trong sự kiện ready của document thì sẽ không có tác dụng gì hết khi đoạn script xuất hiện trước đoạn định nghĩa btb\_1 ở ví dụ trên. Trong trườn hợp không muốn sử dụng sự kiện ready của document ta cần chèn đoạn script định nghĩa sự kiện xuất hiện sau đoạn HTML định nghĩa đối tượng btb\_1.

***3.3.3 jQuery Effects***

*a) Hide and show*

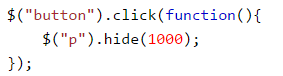
Cú pháp:



*speed:* tốc độ diễn ra hiệu ứng, với các giá trị: slow, fast, giá trị cụ thể

*callback:* hàm thực thi khi kết thúc hiệu ứng

Ví dụ:



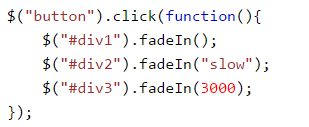
*b)Fading*

*fadeIn()* được sử dụng để ẩn đi một phần tử.

Cú pháp:



Ví dụ:

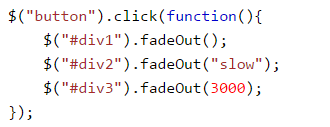


*fadeOut()* được sử dụng để làm mờ đi một phần tử.

Cú pháp:



Ví dụ:



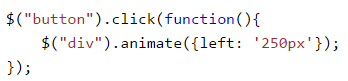
*c) Animations*

*animate()* được sử dụng để tạo hiệu ứng tùy chỉnh.

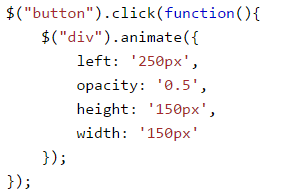
Cú pháp:



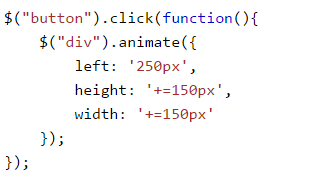
Ví dụ 1:



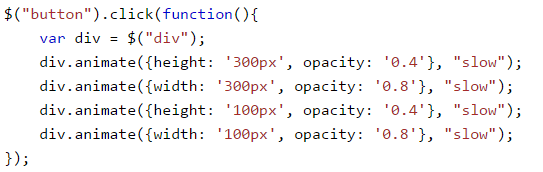
Ví dụ 2:



Ví dụ 3:



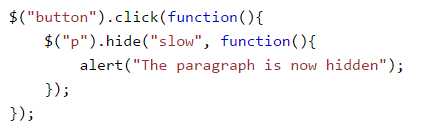
Ví dụ 4:



*d)Callback Function*

Hàm callback được thực hiện sau khi hiệu ứng kết thúc.

Ví dụ:

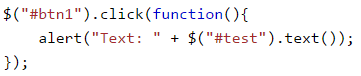


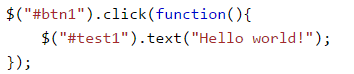
***3.3.4 jQuery HTML***

*a) Get và Set*

*text():* thiết lập hoặc trả về nội dung text của phần tử được chọn

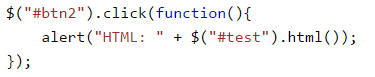
Ví dụ:





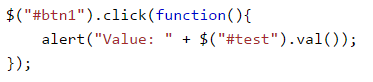
*html():* thiết lập hoặc trả về nội dung html của phần tử được chọn bao gồm cả phần nội dung html

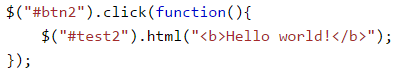
Ví dụ:



*val():* lấy giá trị hiện tại của thành phần hoặc thay đổi giá trị cho thành phần, val() sẽ lấy giá trị đầu tiên nếu thành phần chọn là một danh sách.

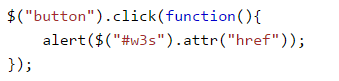
Ví dụ:

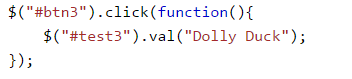




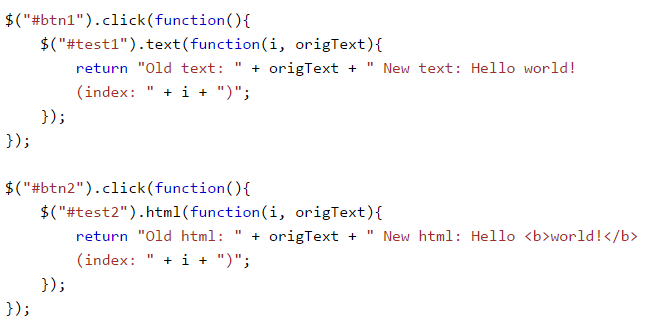
*attr():* lấy giá trị của thuộc tính

Ví dụ:





*Callback Function*:



*b) Add*

*append():* chèn nội dung vào cuối phần tử được chọn

Ví dụ:



*prepend():* chàn nội dung vào đầu phần tử được chọn

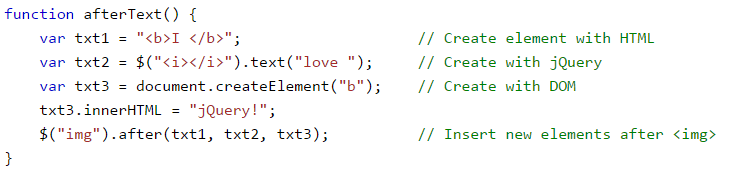
Ví dụ:



*after():* chèn nội dung vào sau phần tử HTML được chọn

*before():* chèn nội dung vào trước phần tử HTML được chọn

Ví dụ:



*c) Remove*

*remove():* loại bỏ phần tử được chọn bao gồm cả phần tử con

Ví dụ:



*empty():* loại bỏ các phần tử con của phần tử được chọn

Ví dụ:



*d) css*

*css()* đặt hoặc trả về một hoặc nhiều thuộc tính cho các phần tử đã chọn

Cú pháp:







Ví dụ:





