Nội dung ôn tập thi cuối kỳ - 90 phút – thi tự luận trên máy tính (làm bài vào tập tin Microsoft Word] – Không sử dụng tài liệu, không dùng trình duyệt

– Các nội dung căn bản (các phần chữ màu đen) cần biết, hiểu và áp dụng được vào việc làm web

– Các nội dung nâng cao (các phần chữ màu đỏ), chỉ cần biết ở mức tổng quan

Domain, hosting

DNS, DNS server, domain name, hosting, XAMPP, *WAMP*, Apache, *IIS*

HTML

webkit engine browser, gecko engine browser, www, web, webpage, website, static website, dynamic website, web app, browser, text editor, code editor, IDE, HTML, HTTP, HTML tag, empty tag, HTML element, attribute, doctype, markup validation, inline element, block element, URL, absolute path, relative path, web client, web server, render, DOM

CSS

Cascading, rule, selector, declaration, styling text, styling color, styling background color, pseudo-class, pseudo-element, box model, floating, positioning, flex, grid, layout, reset CSS, CSS sprite, *SASS, LESS, UI/UX, bootstrap*.

JavaScript

Variable, data types, array, logic, loop, function, object, event, DOM, callback function,

*jquery, react, angular, vue*.

Mục lục

[Domain, hosting 4](#_Toc44362802)

[DNS - Domain Name System 4](#_Toc44362803)

[DNS server – Domain Name System server 4](#_Toc44362804)

[Domain name 4](#_Toc44362805)

[XAMPP 5](#_Toc44362806)

[WAMP 5](#_Toc44362807)

[Apache 5](#_Toc44362808)

[IIS 5](#_Toc44362809)

[HTML 5](#_Toc44362810)

[DOM – Document Object Model 5](#_Toc44362811)

[WWW – World Wide Web 6](#_Toc44362812)

[Webkit 7](#_Toc44362813)

[Internet 8](#_Toc44362814)

[Web page 9](#_Toc44362815)

[Website 9](#_Toc44362816)

[Static website 9](#_Toc44362817)

[Dynamic website 9](#_Toc44362818)

[Web application 9](#_Toc44362819)

[Web browser 10](#_Toc44362820)

[Text editor 10](#_Toc44362821)

[Code editor 11](#_Toc44362822)

[IDE - Integrated Development Environment 11](#_Toc44362823)

[HTML – Hypertext Markup Language 11](#_Toc44362824)

[HTTP – Hyper Text Transfer Protocol 11](#_Toc44362825)

[Tag 11](#_Toc44362826)

[Empty tag 11](#_Toc44362827)

[Element 12](#_Toc44362828)

[Attribute 12](#_Toc44362829)

[Doctype 13](#_Toc44362830)

[Inline element 13](#_Toc44362831)

[Block element 13](#_Toc44362832)

[URI – Uniform Resource Identifier 13](#_Toc44362833)

[URN – Uniform Resource Name 13](#_Toc44362834)

[URL – Uniform Resource Locator 13](#_Toc44362835)

[Path 15](#_Toc44362836)

[Web client 15](#_Toc44362837)

[CSS 15](#_Toc44362838)

[Cascading 16](#_Toc44362839)

[Specificity 17](#_Toc44362840)

[Order 18](#_Toc44362841)

[Rule 18](#_Toc44362842)

[Selector 18](#_Toc44362843)

[Declaration 22](#_Toc44362844)

[Inhertitance 22](#_Toc44362845)

[Box model 23](#_Toc44362846)

[Styling text 24](#_Toc44362847)

[Styling background 25](#_Toc44362848)

[Typeface, font-family, @font-face 25](#_Toc44362849)

[Pseudo-class 27](#_Toc44362850)

[Pseudo-element 28](#_Toc44362851)

[Normal flow 29](#_Toc44362852)

[Float 29](#_Toc44362853)

[Clear 30](#_Toc44362854)

[Positioning 30](#_Toc44362855)

[Layout 31](#_Toc44362856)

[Flexbox 32](#_Toc44362857)

[Grid system 35](#_Toc44362858)

[Reset CSS 35](#_Toc44362859)

[Sprite 36](#_Toc44362860)

[JavaScript 36](#_Toc44362861)

## Domain, hosting

### DNS - Domain Name System

Domain Name System: Hệ thống tên miền hay Hệ thống phân giải (resolve) tên miền. DNS dùng để lưu trữ các tên miền (domain name, ví dụ www.google.com), cung cấp dịch vụ chuyển đổi từ địa chỉ dạng chuỗi (tên miền) sang địa chỉ dạng số (IP) và ngược lại (ví dụ www.google.com <> 216.58.221.228).

### DNS server – Domain Name System server

Máy chủ hệ thống tên miền: thông thường, để tìm ra chính xác địa chỉ IP của 1 website cần sự phối hợp hoạt động của 4 máy chủ DNS (Nếu không có quá trình cache).

* DNS Resolver: Máy chủ đầu tiên, nhận truy vấn DNS từ máy khách, sau khi nhận được truy vấn, nó tiến hành truy vấn đến 3 máy chủ DNS còn lại để tìm đúng địa chỉ IP của tên miền.
* Root nameserver: Máy chủ này phản hồi thông tin về phần mở rộng (ví dụ .net, .com) chứa trong tên miền đến DNS Resolver.
* TLD server - Top Level Domain server: Sau khi nhận phản hồi từ Root nameserver, DNS Resolver gửi truy vấn đến TLD server, TLD phản hồi thông tin về địa chỉ IP của máy chủ Authoritative DNS.
* Authoritative nameserver: đây là máy chủ cuối cùng lưu trữ địa chỉ IP của tên miền cần phân giải, tại đây địa chỉ IP tương ứng của tên miền sẽ được phản hồi lại cho DNS Resolver, và DNS Resolver phản hồi địa chỉ IP này về máy khách. Máy khách dùng địa chỉ IP này để tương tác với server chứa trang web.

### Domain name

Tên miền là 1 chuỗi kí tự, giúp con người thuận tiện hơn trong việc truy cập các tài nguyên trên internet. Thay vì phải nhớ địa chỉ IP của 1 thiết bị trên internet, chúng ta chỉ cần nhớ tên miền của nó.

**Tên miền cấp 1, cấp 2, cấp 3 là gì? cho ví dụ mỗi loại?**

Tên miền cấp 1 cũng là tên miền quốc tế, được dùng chung cho nhiều quốc gia, mỗi tên miền đại diện cho một lĩnh vực, một ngành nghề, hay một khu vực địa lý. Ví dụ: .com, .net, .org, .mil, .edu, .gov, .asia, .eu

Tên miền cấp 2 cũng là tên miền quốc gia, thông thường mỗi quốc gia sẽ có một tên miền riêng, gồm hai kí tự. Ví dụ: .vn (Việt Nam), .cn (Trung Quốc), .uk (Anh), .us (Mỹ).

Tên miền cấp 3 là tên miền kết hợp giữa tên miền cấp 2 và tên miền cấp 1. Ví dụ: .com.vn, .edu.vn, .edu.uk, .com.us

**Tên miền quốc tế là gì? cho 3 ví dụ?**

– Tên miền quốc tế: do Trung tâm quản lý tên miền quốc tế cấp, ví dụ thường có đuôi là .com, .net, .biz, .info, .org

**Tên miền “nội địa” hay quốc gia là gì? cho 3 ví dụ?**

– Tên miền quốc gia (nội địa): do Trung tâm quản lý tên miền của mỗi quốc gia quản lý. Ví dụ tên miền của Việt Nam có đuôi dạng .vn, .com.vn, edu.vn, gov.vn. Các tên miền này do VNNIC quản lý.

### XAMPP

Là 1 chương trình mã nguồn mở chạy đa nền tảng, dùng để tạo web server. XAMPP được dùng bởi các lập trình viên để phát triển dự án website theo ngôn ngữ PHP. XAMPP được viết tắt từ các từ: X – nói về ứng dụng chạy đa nền tảng, A – Apache, M – MariaDB sau này được thay thế bởi MySQL, P – PHP, P – Perl.

### WAMP

### Apache

Là 1 phần mềm web server mã nguồn mở miễn phí. Nó giúp chủ website đưa nội dung lên web – vì vậy có tên gọi là “web server”.

### IIS

## HTML

### DOM – Document Object Model

DOM là viết tắt của Document Object Model (Mô hình đối tượng tài liệu). DOM là một chuẩn của W3C, dùng để biểu diễn tài liệu HTML, XML; cho phép người lập trình có thể truy cập và xử lý nội dung của tài liệu HTML, XML bằng ngôn ngữ lập trình, ví dụ ngôn ngữ JavaScript. DOM gồm ba loại là Core DOM, XML DOM và HTML DOM. Trong phần này chỉ quan tâm tới HTML DOM, để tiện trình bày, gọi tắt là DOM.

Ở dạng DOM, mỗi phần tử (element) của tài liệu HTML đều được xem là một đối tượng, có thuộc tính và phương thức. Do đó, người lập trình có thể lấy, thay đổi, thêm, xóa các phần tử của tài liệu HTML. Mỗi phần tử được biểu diễn bằng một nút (node) trong cây DOM.

Tài liệu HTML sẽ được chuyển thành một trong các loại nút sau:

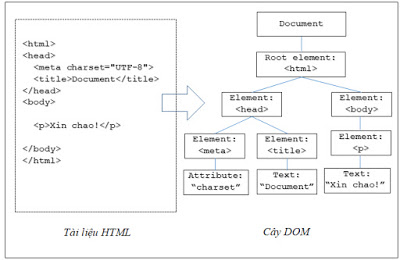
– Document: là nút gốc, đại diện cho toàn bộ tài liệu HTML

– Element: tất cả các phần tử của HTML đều thuộc loại nút này

– Text: phần nội dung văn bản trong tài liệu HTML thuộc loại nút này

– Attribute: thuộc tính của các phần tử trong tài liệu HTML thuộc loại nút này

­– Comment: các chú thích trong tài liệu HTML thuộc loại nút này



### WWW – World Wide Web

Web hay WWW là viết tắt của World Wide Web.

Web là không gian thông tin toàn cầu.

Web là một hệ thống bao gồm các đặc điểm sau:

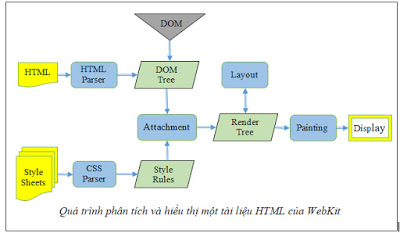
– Chứa thông tin dưới dạng tài liệu (document) và các dạng tài nguyên khác, gọi chung là tài nguyên web

– Các tài nguyên web được xác định (định danh) bằng URL

– Các tài nguyên web được gắn kết với nhau bởi các liên kết (link) trong các siêu văn bản (hypertext)

– Có thể truy cập tới các tài nguyên web thông qua Internet

### Webkit



Ở hình trên:

– HTML: nội dung tài liệu HTML

– Style Sheets: nội dung CSS

– HTML Parser: bộ phận phân tích mã HTML

– CSS Parser: bộ phận phân tích CSS

– DOM: mô hình DOM

– DOM Tree: cây DOM

– Style Rules: thông tin định dạng, và hiển thị các phần tử HTML

– Attachment: bộ phận kết hợp các thông tin định dạng(CSS) vào các nút của cây DOM

– Layout: khung kết xuất các phần tử HTML

– Render Tree: cây kết xuất

– Painting: Bộ phận hiển thị

– Display: trang web kết quả

Quá trình biên dịch và hiển thị nội dung gồm bốn bước sau:

– Phân tích (parsing) mã HTML để tạo cây DOM (DOM tree)

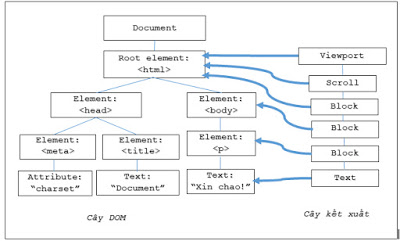
– Tạo cây kết xuất (render tree)

Sau khi đã có cây DOM, bộ phận kết xuất tiếp tục phân tích nội dung CSS, kết hợp với một số thông tin khác trong tài liệu HTML để tạo ra cây kết xuất.

Cây kết xuất chứa thông tin để hiển thị các phần tử của tài liệu HTML lên cửa sổ trình duyệt. Các thông tin gồm: chiều rộng, chiều cao, vị trí, màu, hướng của phần tử.

Một cách trực quan, cây kết xuất chứa các hình chữ nhật cùng với thông tin về màu và hướng. Đây là các hình chữ nhật (hay các khung) sẽ được hiển thị trong cửa sổ trình duyệt.

Cây kết xuất có cấu trúc gần giống với cây DOM. Tuy nhiên, các nút trên cây DOM nếu không được hiển thị sẽ không có trong cây kết xuất, ví dụ nút [Element:<head>] và các nút con của nó. Các nút được đánh dấu là “không hiển thị” cũng không xuất hiện trong cây kết xuất. Ngược lại, một số nút trên cây DOM sẽ có nhiều nút tương ứng trên cây kết xuất.



– Dựng khung kết xuất (layout)

Sau khi đã có cây kết xuất, bộ phận kết xuất sẽ xác định tọa độ cho các nút trên cửa sổ trình duyệt. Quá trình này tạm gọi là dựng khung (layout).

– Hiển thị (paint)

Bước cuối cùng, bộ phận kết xuất sẽ chuyển các dữ liệu cần thiết cho bộ phận UI Backend để xuất nội dung lên cửa sổ trình duyệt.

### Internet

Nhiều người thường xem Internet và web là một. Tuy nhiên, đây là hai cái khác nhau.

Internet là một hệ thống kết nối toàn cầu của các mạng máy tính (computer network).

Trong khi, web là không gian thông tin toàn cầu, là tập hợp có liên kết của các tài liệu và các tài nguyên web khác.

Hiểu nôm na, Internet cung cấp hạ tầng, còn web là nội dung nằm trên hạ tầng Internet.

### Web page

Trang web (web page hay webpage) là một tài liệu, được sử dụng trong hệ thống web, hoặc trong trình duyệt. Nhắc lại, tài liệu là một vật mang tin. Trang web là một siêu văn bản. Ví dụ, mở trình duyệt gõ vào địa chỉ “https://www.w3.org”, gõ phím Enter, khi đó mọi thông tin xuất hiện trên màn hình chính là nội dung của một trang web.

Trang web là các trang đơn vị để cấu thành website. Ví dụ, một website có thể được cấu thành từ các trang web sau: trang chủ, trang tin, trang quản trị, trang đăng nhập.

Mỗi trang web thường phục vụ cho một mục đích cụ thể. Đây là địa chỉ của một trang web, “www.congty.com/index.html”. Một website có thể chỉ bao gồm một trang web.

### Website

Website là một tập hợp nhiều trang web có liên quan đến nhau. Thông thường, website có một cái tên duy nhất, gọi là tên miền (domain name). Website được đặt trên ít nhất một máy chủ web (web server). Người dùng có thể truy cập tới website bằng mạng công cộng (Internet) hoặc mạng nội bộ (LAN) thông qua địa chỉ của nó (URL).

### Static website

Trang web tĩnh (static web page, flat page hoặc stationary page): là trang web chứa nội dung cố định, nội dung này là giống nhau với mọi người dùng, mọi ngữ cảnh. Người dùng không có khả năng tương tác với nó.

Mã nguồn của trang web tĩnh thường là một tài liệu HTML, được lưu dưới dạng một tập tin.

Trang web tĩnh thích hợp cho các nội dung ít hoặc không cần cập nhật.

### Dynamic website

Trang web động (dynamic web page, live web page, hoặc interactive web page): là trang web mà nội dung của nó có thể thay đổi tùy thuộc vào ngữ cảnh và điều kiện khác nhau. Việc thay đổi có thể được thực hiện tại phía server (server-side) bằng các ngôn ngữ lập trình phía server (ví dụ: C#, PHP, Java, Python, Ruby) hoặc tại phía client (client-side) bằng ngôn ngữ lập trình phía client (ví dụ JavaScript), hoặc kết hợp cả phía client và phía server bằng kĩ thuật Ajax.

### Web application

Ứng dụng web[[8]](file:///D:\\Liv\\TuHoc\\Hoc%20lam%20web%201_HTML.docx" \l "_ftn8" \o ") (web application hoặc web app): là một chương trình máy tính hoạt động dựa trên mô hình client-server, trong đó chương trình được lưu trên server, giao diện người dùng được chạy trên một trình duyệt web, tạm hiểu là, chương trình để ở một nơi (server), người dùng thì lại ở một nơi khác (client), hay chương trình để ở máy tính này, nhưng khi sử dụng thì lại dùng ở một máy tính khác. Nó khác so với ứng dụng trên máy tính đơn.

Ứng dụng web hoạt động dựa trên nền tảng web. Người dùng có thể chạy ứng dụng thông qua Internet, Intranet. Để tạo ra ứng dụng web, người ta thường sử dụng các ngôn ngữ lập trình kết hợp với các framework chuyên dụng.

Chưa có sự phân biệt rõ ràng giữa trang web động và ứng dụng web, nói chung cứ website nào có chức năng tương tự như một ứng dụng máy tính hoặc ứng dụng di động thì được xem như một ứng dụng web.

Một số ví dụ về các ứng dụng web: thư điện tử, bán hàng trực tuyến, diễn đàn, nhật kí mạng, bản đồ, quản lý nhân lực, quản lý y tế và chăm sóc sức khỏe.

So sánh sự khác nhau giữa website động và ứng dụng web:

|  |  |
| --- | --- |
| Website động | Ứng dụng web |
| - Cho phép người dùng tương tác và thêm nội dung, tuy nhiên phần lớn nội dung vẫn là do nhà phát triển tạo ra.  - Ít phức tạp và dễ phát triển hơn so với ứng dụng web. | - Hoạt động dựa trên sự tương tác từ người dùng, hầu hết nội dung và dữ liệu là do người dùng đóng góp  - Giống như một ứng dụng máy tính  - Phức tạp và cần nhiều kĩ năng hơn để phát triển so với website động |

### Web browser

Trình duyệt web (gọi tắt là trình duyệt) là một phần mềm ứng dụng, cho phép người dùng xem và tương tác với: văn bản, hình ảnh, đoạn phim, nhạc, trò chơi và các thông tin khác ở trên một trang web. Trang web có thể nằm trên WWW hoặc trong mạng nội bộ.

Văn bản và hình ảnh trên trang web có thể chứa liên kết tới các trang web khác ở cùng hoặc khác website. Trình duyệt web cho phép người dùng truy cập các thông tin trên các trang web nhanh chóng và dễ dàng thông qua các liên kết đó. Trình duyệt sẽ biên dịch mã HTML để hiển thị, do vậy, một trang web có thể được hiển thị khác nhau trên các trình duyệt khác nhau.

### Text editor

Text editor là trình soạn thảo văn bản thô, cho phép lập trình viên có thể viết mã nguồn và lưu lại thành một tập tin. Tuy nhiên, hiện nay các text editor rất tiện dụng, linh hoạt và cho phép tích hợp thêm rất nhiều các chức năng nhằm hỗ trợ việc lập trình tốt hơn. Ưu điểm của text editor là gọn, nhẹ.

### Code editor

Là 1 trình soạn thảo văn bản (text editor), nhưng được thiết kế chuyên biệt để soạn thảo mã nguồn của các ngôn ngữ lập trình. Là 1 công cụ lập trình cơ bản. Trình soạn thảo mã nguồn có 1 số chức năng giúp đẩy nhanh quá trình soạn thảo mã nguồn như: tô màu cú pháp, căn thụt lề đầu dòng, tự động nhắc từ,…1 số code editor phổ biến như visual studio code, sublime text,…

### IDE - Integrated Development Environment

IDE (Integrated Development Environtment – môi trường phát triển tích hợp) là một phần mềm được sử dụng để viết mã, tuy nhiên ngoài chức năng của một text editor, nó còn được tính hợp thêm nhiều chức năng khác, ví dụ như:

– Trình biên/thông dịch

– Tìm và sửa lỗi (debug)

– Tự động chèn mã (Autocompletion)

– Tự động báo lỗi khi viết mã

– Gợi ý các hàm, biến

### HTML – Hypertext Markup Language

Là 1 ngôn ngữ đánh dấu văn bản, được thiết kế để tạo văn bản có cấu trúc và hiển thị trên trình duyệt web.

### HTTP – HyperText Transfer Protocol

Là giao thức truyền thông tin trên môi trường world wide web. Dùng để giao tiếp giữa máy khách (client) và máy chủ (server). Giao tiếp được bắt đầu bằng việc máy khách gửi HTTP request, máy chủ nhận yêu cầu và phản hồi HTTP response.

### Tag

Thẻ HTML (HTML tag) là các từ khóa (keyword), được sử dụng để báo cho trình duyệt biết cách định dạng và hiển thị các nội dung. Trình duyệt sẽ không hiển thị các từ khóa này ra ngoài giao diện. Thẻ HTML thường gồm hai phần: phần mở thẻ gọi là thẻ mở (opening tag, start tag) và phần đóng thẻ gọi là thẻ đóng (closing tag, end tag).

### Empty tag

Có một số ít các phần tử không có phần nội dung, mà bản thân phần tử đó sẽ là một chỉ dẫn để thực hiện một hành động nào đó, như xuống dòng, tạo một đường kẻ ngang, hiển thị một hình ảnh, các phần tử này được gọi là phần tử rỗng. Ví dụ, phần tử <img> để tải lên 1 hình ảnh, phần tử <br> để xuống dòng, phần tử <hr> dùng để tạo 1 đường kẻ ngang,…

### Element

1 phần tử được xác định là tập hợp tag và nội dung bên trong nó. Một phần tử HTML sẽ bắt đầu là thẻ mở, tới phần nội dung, và cuối cùng là thẻ đóng.

### Attribute

Thuộc tính là thành phần bổ sung của mỗi phần tử HTML. Thuộc tính được dùng để thay đổi chức năng mặc định của phần tử hoặc là một thành tố bắt buộc phải có, để một phần tử có thể thực hiện đúng chức năng.

Với các phần tử gồm cả thẻ mở và thẻ đóng thì thuộc tính luôn được đặt trong thẻ mở. Mỗi phần tử có thể chứa nhiều thuộc tính, mỗi thuộc tính cách nhau bởi một khoảng trắng, tính thứ tự trước sau của các thuộc tính là không quan trọng.

Hầu hết các thuộc tính đều có giá trị đi kèm, giá trị của thuộc tính nằm sau dấu “=”. Tuy nhiên, có một số trường hợp thuộc tính được viết rút gọn chỉ gồm một từ mô tả, ví dụ thuộc tính checked.

Giá trị của thuộc tính có thể là: văn bản, số, kí tự, URL, đơn vị đo.

Giá trị của thuộc tính thường được bao lại bằng dấu nháy kép ("") hoặc nháy đơn ('). Mọi người hay sử dụng dấu nháy kép. Tuy nhiên, cũng có một số giá trị không cần phải bao lại bằng dấu nháy. Lưu ý, đừng nhầm dấu ("") với dấu (“”) hoặc dấu (') với dấu (‘), việc chép mã HTML từ tài liệu Word để chạy thử hay bị nhầm các dấu này, và gây ra lỗi.

Có 4 loại thuộc tính cơ bản:

|  |  |
| --- | --- |
| **Loại thuộc tính** | **Mô tả** |
| Thuộc tính bắt buộc (required attributes) | Là thuộc tính bắt buộc phải có để phần tử thực hiện đúng chức năng của nó. |
| Thuộc tính tùy chọn (optional attributes) | Là thuộc tính được sử dụng để thay đổi chức năng mặc định của một phần tử. |
| Thuộc tính chuẩn (standard attributes) hay thuộc tính phổ dụng (global attributes) | Là thuộc tính có thể sử dụng được cho nhiều loại phần tử. |
| Thuộc tính sự kiện (event attributes) | Là thuộc tính được sử dụng để kích hoạt một hành động (script) trong các trường hợp cụ thể |

### Doctype

Document Type xác định phiên bản của văn bản HTML đang được dùng. Chuẩn hiện tại là HTML5.

### Inline element

Một phần tử có kiểu hiển thị là inline (inline-level) sẽ không yêu cầu một hàng mới khi hiển thị và nó sẽ chiếm độ rộng vừa đủ để chứa hết phần nội dung của nó. Một số phần tử hiển thị kiểu inline như: <a>, <button>, <span> ...

### Block element

Một phần tử có kiểu hiển thị là block (block-level) sẽ luôn được hiển thị ở một hàng mới và nó sẽ chiếm toàn bộ chiều rộng khả dụng (nghĩa là chiếm toàn bộ chiều rộng của phần tử chứa nó). Một số phần tử hiển thị kiểu block như: <h1>, <p>, <div>…

### URI – Uniform Resource Identifier

URI (Uniform Resource Identifier) – tạm dịch là “định dạng tài nguyên thống nhất” hay “nhận dạng tài nguyên thống nhất”: là một chuỗi ký tự, được sử dụng để xác định, nhận dạng một tên hoặc một tài nguyên. Gọi nôm na là “chuỗi nhận dạng tài nguyên thống nhất”, gọi tắt là chuỗi nhận dạng.

URI gồm hai loại là URL và URN, trong đó:

– URL là nhận dạng tài nguyên bằng địa chỉ, nghĩa là cho biết tài nguyên nằm ở đâu và dùng giao thức gì để lấy được nó

– URN là nhận dạng tài nguyên bằng tên, nghĩa là cho biết tên của tài nguyên mà không cho biết tài nguyên này nằm ở đâu và làm sao để lấy được tài nguyên này

Một URI cũng có thể được tạo ra bằng cách kết hợp URL và URN.

### URN – Uniform Resource Name

URN (Uniform Resource Name – tạm dịch là nhận dạng tài nguyên bằng tên), vì tên thì rất dễ bị trùng nên phải thêm các thông tin đi kèm, ví dụ phải được đặt tên theo một hệ thống thống nhất (namespace). ISBN là một hệ thống như vậy, ví dụ: urn:isbn:0451450523. Hoặc hệ thống tên của ietf sẽ là urn:ietf:rfc:2648.

### URL – Uniform Resource Locator

URL[[5]](file:///D:\\Liv\\TuHoc\\Hoc%20lam%20web%201_HTML.docx" \l "_ftn5" \o ") là viết tắt của Uniform Resource Locator, dịch sát nghĩa là “bộ định vị tài nguyên thống nhất”, hiểu nôm na là “nhận dạng tài nguyên bằng địa chỉ”, hay địa chỉ tài nguyên trên web, gọi tắt là địa chỉ web (web address).

URL là một tham chiếu tới tài nguyên web, cho biết vị trí của tài nguyên web trên hệ thống mạng và cho biết sử dụng giao thức gì để lấy được tài nguyên đó.

Phân tích cấu trúc URL:

Định dạng chung:



Ví dụ 1 số URL:

*https://en.wikipedia.org/wiki/Uniform\_Resource\_Locator*

*http://daotao.congty.vn/e-learning/ViewCourses.php?cSval=3151&SjTemp=158#noi-dung*

[*file:///C:/Users/Win%208.1/Downloads/openemr/trangweb.html*](file:///C:/Users/Win%208.1/Downloads/openemr/trangweb.html)

Ý nghĩa các thành phần:

– Scheme là giao thức được sử dụng để lấy tài nguyên hay để giao tiếp giữa web client và web server. Ví dụ: http, https, ftp, file.

– Domain là tên miền của web server, nơi chứa (các) trang web. Ví dụ: en.wikipedia.org, daotao.congty.vn.

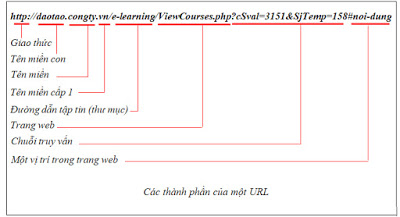
– Port là cổng được sử dụng để giao tiếp giữa web client và web server, nếu không chỉ định rõ, thì giá trị của port sẽ được ngầm hiểu là 80.

– Path là đường dẫn (thư mục) của trang web hoặc nơi chứa các tài nguyên của một website tại web server.

– Ngoài ra, URL cũng có thể chứa chuỗi truy vấn (query string), ví dụ, *cSval=3151&SjTemp=158;* hoặc một định danh trên trang HTML (fragment\_id), ví dụ #noi-dung.

Mặc dù URL là tập con của URI, và cộng đồng đang có xu hướng chuyển qua sử dụng URI thay cho URL vì URI chính xác và tổng quát hơn. Tuy nhiên, đôi khi người ta vẫn thích sử dụng khái niệm URL hơn. Trong thực tế, URL và URI có thể sử dụng thay thế cho nhau.

Hình minh họa 1 URL:



### Path

Absolute: URL tuyệt đối (absolute URL): cung cấp đường dẫn đầy đủ để có thể lấy được tài nguyên, bao gồm giao thức (protocol), tên miền (domain name), đường dẫn tập tin (pathname). URL tuyệt đối thường được sử dụng khi cần liên kết đến một tài nguyên không nằm trên cùng server. Ví dụ: href=“http://www.google.com/”

Relative: URL tương đối (relative URL): cung cấp đường dẫn tương đối để có thể lấy được tài nguyên, URL tương đối thường được sử dụng khi cần liên kết đến một tài nguyên nằm trên cùng server. Khi đó, URL sẽ không cần có giao thức và tên miền, mà chỉ cần đường dẫn tập tin (pathname). Ví dụ: href=“images/logo.gif”

### Web client

Là một chương trình có thể giao tiếp với Web Server, gửi yêu cầu và nhận thông tin từ Web Server, sau đó xử lý các thông tin nhận được để hiển thị hoặc phục vụ các mục đích khác. Trình duyệt chính là một Web Client.

## CSS

### Link CSS

**Sử dụng phần tử link**

Trong tài liệu HTML, để tạo liên kết tới tập tin *.css*, sử dụng phần tử *link* trong *head*. Đây là phương pháp được sử dụng phổ biến nhất.

 Ví dụ,

<head>

<title>Titles are required</title>

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

</head>

Ý nghĩa các thuộc tính của phần tử *link*,

– *rel="stylesheet"*: cho biết mối liên hệ giữa tài liệu hiện tại (HTML) và tài liệu được liên kết đến (CSS), *rel* là viết tắt của relationship, mối liên hệ ở đây là “mã định dạng”

– *type="text/css"*: cho biết loại tài liệu được liên kết đến, ở đây là một tài liệu CSS

– *href="/path/stylesheet.css"*: đường dẫn tới tập tin *.css*, đường dẫn được bắt đầu từ thư mục gốc của website (“/”).

Trong một tài liệu HTML, có thể sử dụng nhiều phần tử *link* để liên kết tới nhiều tập tin .*css* khác nhau, khi đó tất cả các luật trong các tập tin *.css* đều được áp dụng, nếu có xung đột, luật nào nằm trong tập tin *.css* được liên kết sau sẽ có quyền ưu tiên cao hơn.

**Sử dụng luật import**

Đây là cách thứ hai để tạo sự kiên kết giữa tài liệu HTML và CSS. Vì *import* là một luật của CSS nên nó phải được đặt trong phần tử *style*, hoặc đặt trong một tập tin *.css*.

Ví dụ, đặt trong phần tử style,

<head>

<style>

@import url("/path/stylesheet.css");

p { font-face: Verdana;}

</style>

<title>Titles are required.</title>

</head>

Lưu ý: luật *import* phải nằm trước mọi bộ chọn CSS, có thể sử dụng nhiều luật *import* để liên kết tới nhiều tập tin *.css*, tập tin *.css* được *import* sau sẽ có độ ưu tiêu cao hơn.

### Cascading

Từ cascade có nghĩa là “xếp lớp” hay “dòng thác”, hiểu theo nghĩa là khi có nhiều tài liệu CSS cùng muốn áp dụng trên một phần tử, thì làm ơn hãy xếp hàng, anh nào có độ ưu tiên thấp thì đứng trước, anh nào có độ ưu tiên cao hơn thì đứng sau. Khi có hiệu lệnh, thì từng anh một thực hiện việc định dạng của mình, anh có độ ưu tiên thấp hơn định dạng trước, anh có độ ưu tiên cao hơn định dạng sau và sẽ có quyền ghi đè anh có độ ưu tiên thấp hơn.

Như đã biết, mã CSS do lập trình viên tạo ra có ba loại, ứng với ba nguồn chứa khác nhau, gồm: đặt trong phần tử HTML, đặt trong tài liệu HTML và đặt ngoài tài liệu HTML. Nói chung, tài liệu CSS nào nằm càng gần phần tử HTML hơn thì có độ ưu tiên cao hơn, vậy độ ưu tiên cao nhất sẽ là CSS đặt trong phần tử HTML, sau đó đến CSS đặt trong tài liệu HTML và cuối cùng là CSS đặt ngoài tài liệu HTML.

Bảng sau là thứ tự ưu tiên của các tài liệu CSS, dựa vào nguồn của nó, thứ tự ưu tiên sẽ tăng dần từ trên xuống,

|  |
| --- |
| CSS mặc định của trình duyệt |
| CSS do người duyệt web định nghĩa |
| CSS để bên ngoài tài liệu HTML, được liên kết bằng phần tử *link* |
| CSS để bên ngoài tài liệu HTML, được liên kết bằng chức năng *@import* |
| CSS đặt ngay trong tài liệu HTML, được chứa trong phần tử *style* |
| CSS đặt ngay trong phần tử HTML, được chứa trong thuộc tính *style* |
| Các luật do lập trình viên đánh dấu là “!important” |
| Các luật do người duyệt web đánh dấu là “!important” |

Để ngăn việc ghi đè cho một luật CSS bất kì, sử dụng từ khóa “!important” ngay trước dấu “;” của mỗi luật. Ví dụ,

### Specificity

Khi đã chọn được tài liệu CSS để áp dụng cho tài liệu HTML rồi, thì vẫn có thể xảy ra tình trạng xung đột. Do vậy, lại cần phải sử dụng đến tính chất “cascading” ở mức luật (rule level). Nghĩa là, khi có hai luật trong một tài liệu CSS có xung đột thì sẽ dựa vào loại của bộ chọn (type of selector) để xác định độ ưu tiên, bộ chọn nào càng rõ ràng (specificity) thì có độ ưu tiên cao hơn.

Như đã biết, khi có hai luật CSS bị xung đột thì luật nào càng rõ ràng, càng cụ thể thì càng có độ ưu tiên cao hơn. Thế nào là rõ ràng, cụ thể? Phương pháp xác định tính rõ ràng, cụ thể của CSS khá phức tạp, tuy nhiên, dưới đây sẽ liệt kê các loại bộ chọn có tính rõ ràng, cụ thể từ cao xuống thấp:

– ID

– Class

– Bộ chọn theo ngữ cảnh (contextual selectors)

– Bộ chọn là tên phần tử

Có thể hiểu “tính rõ ràng, cụ thể” của một bộ chọn chính là khả năng bao phủ (giới hạn) các phần tử trong tài liệu HTML của bộ chọn đó, bộ chọn nào càng bao phủ (giới hạn) được ít phần tử thì bộ chọn đó càng rõ ràng, càng cụ thể.

Ví dụ, với hai luật CSS sau, cái thứ hai sẽ có độ ưu tiên cao hơn, do nó là một bộ chọn theo ngữ cảnh (chọn dựa vào các hậu duệ), và rõ ràng độ bao phủ của nó sẽ nhỏ hơn so với cái thứ nhất,

strong { color: red;}

h1 strong { color: blue; }

Kinh nghiệm trong thực tế là nên viết luật CSS cho bộ chọn là tên một phần tử HTML trước, sau đó sẽ ghi đè luật CSS này bằng các luật CSS cụ thể hơn. Ví dụ,

p { line-height: 1.2em; }

blockquote p { line-height: 1em; }

p.intro { line-height: 2em; }

### Order

Cuối cùng, với hai luật có cùng độ ưu tiên vẫn có thể xảy ra xung đột, khi đó cũng áp dụng quy tắc “cái nào gần nội dung cần định dạng hơn sẽ có độ ưu tiên cao hơn (cái cuối cùng sẽ có độ ưu tiên cao nhất)”.

### Rule

Để định dạng phần tử HTML ta sử dụng các luật (rules hoặc rule sets), đây là các chỉ dẫn để mô tả cách hiển thị một phần tử hoặc nhóm phần tử.

Một luật bao gồm hai phần:  phần “lựa chọn” phần tử và phần “quy định” cách thức hiển thị của phần tử tương ứng.

Ví dụ sau là một đoạn mã CSS gồm hai luật, luật đầu quy định tất cả các phần tử *h1* của tài liệu sẽ có chữ màu xanh, luật sau quy định tất cả các phần tử *p* của tài liệu sẽ có kiểu chữ là sans-serif và chữ nhỏ.

h1 { color: green; }

p { font-size: small; font-family: sans-serif; }

### Selector

Phần lựa chọn phần tử được gọi là selector (bộ chọn)

#### Descendant selector

Như đã biết, phần tử hậu duệ là các phần tử chứa bên trong một phần tử khác (gồm: con, cháu, chắt…). Chọn theo hậu duệ là một trường hợp của phương pháp lựa chọn các phần tử theo ngữ cảnh (contextual selector).

Lựa chọn theo ngữ cảnh nghĩa là lựa chọn dựa vào mối quan hệ giữa các phần tử.

Để chọn theo hậu duệ, sử dụng một dãy các bộ chọn, mỗi bộ chọn ngăn cách nhau bằng một khoảng trắng. Khoảng trắng này có nghĩa là phần tử phía sau sẽ là hậu duệ của phần tử phía trước. Ví dụ:

li em { color: red }

Bộ chọn trên có nghĩa là: chỉ có các phần tử em là hậu duệ của phần tử li mới có chữ màu đỏ, trong khi các phần tử em còn lại sẽ không bị tác động bởi luật CSS này.

#### Contextual selector

Chọn theo các hậu duệ là một trong bốn loại bộ chọn theo ngữ cảnh, CSS3 gọi các bộ chọn này là bộ chọn tổ hợp (combinators), các bộ chọn theo ngữ cảnh khác gồm: chọn các phần tử con (child selectors), chọn một em liền sau (adjacent sibling selectors) và chọn nhiều em (general sibling selectors).

– Chọn các phần tử con

Cách chọn này cũng tương tự như chọn theo các hậu duệ, tuy nhiên, chỉ giới hạn cho các hậu duệ là con trực tiếp. Con trực tiếp của một phần tử được hiểu là các phần tử nằm ở cấp liền kề trong tài liệu HTML (không có cấp nào xen vào giữa). Sử dụng dấu lớn hơn (>) để thể hiện bộ chọn này.

Ví dụ, luật CSS sau sẽ làm cho chữ trong phần tử em có màu đỏ, tuy nhiên, chỉ có phần tử em là con trực tiếp của p mới chịu tác động, trong khi các phần tử em khác thì không chịu tác động.

p > em { color: red; }

– Chọn một em liền kề

Các phần tử có cùng cha thì được gọi là anh em; trong tài liệu HTML, phần tử anh sẽ nằm trước, phần tử em nằm sau; trong cấu trúc cây (cây DOM), phần tử anh nằm ở phía trái, phần tử em nằm ở phía phải.

Chọn một em liền kề của một phần tử là chọn duy nhất một phần tử là em liền kề của nó. Sử dụng dấu cộng (+) để thể hiện bộ chọn này.

Ví dụ, luật CSS sau sẽ làm cho chữ trong phần tử h2 có màu đỏ, tuy nhiên, chỉ có một phần tử h2 nằm ngay sau ul mới chịu tác động, trong khi các phần tử h2 khác thì không chịu tác động.

ul + h2 { color: red; }

– Chọn nhiều em

Bộ chọn theo kiểu nhiều em của một phần tử sẽ chọn tất cả các phần tử là em (có tên được chỉ định) của nó. Sử dụng dấu ngã (~) để thể hiện bộ chọn này.

Ví dụ, luật CSS sau sẽ làm cho chữ trong phần tử h2 có màu đỏ, tuy nhiên, chỉ những phần tử h2 nằm sau ul (cùng cha body) mới chịu tác động, trong khi các phần tử h2 nằm phía trước và các vị trí khác thì không chịu tác động.

ul ~ h2 { color: red; }

#### ID selector

Như đã biết, thuộc tính id được sử dụng để gán định danh (identifier) cho một phần tử bất kì, định danh này là duy nhất trong tài liệu HTML. Trong CSS, có thể sử dụng id làm bộ chọn để thực hiện định dạng cho phần tử, trước id sẽ có thêm dấu thăng (#) (octothorpe hoặc hash symbol).

Ví dụ, có đoạn HTML sau:

<li id="catalog1234">Happy Face T-shirt</li>

Luật CSS sau sẽ định dạng cho duy nhất một phần tử li dựa vào bộ chọn id,

li#id { color: red; }

Tuy nhiên, vì id là duy nhất trong tài liệu HTML, nên có thể bỏ qua tên của phần tử, luật CSS được viết lại như sau,

#id { color: red; }

#### Class selector

Ngoài id, còn có một thuộc tính khác để gán định danh cho các phần tử là class. Thuộc tính class sẽ thực hiện phân lớp các phần tử vào các nhóm một cách logic, tất cả các phần tử có class giống nhau sẽ cùng thuộc về một nhóm. Khác với id, nhiều phần tử có thể có class giống nhau, và một phần tử có thể thuộc về nhiều class.

Trong CSS, có thể sử dụng class để tạo ra bộ chọn gồm các phần tử có giá trị class giống nhau, trước class sẽ có thêm dấu chấm (.).

Ví dụ, bộ chọn sau sẽ chọn tất cả các phần tử p có thuộc tính class là special,

p.special { color: red; }

Nếu muốn áp dụng luật CSS trên cho mọi phần tử có thuộc tính class là special thì bỏ đi tên phần tử p, chỉ giữ lại tên lớp như sau (nhớ có dấu . ở phía trước),

.special { color: red; }

#### Universal selector

Bộ chọn “mọi phần tử” (universal selector) được sử dụng để chọn mọi phần tử, sử dụng dấu \* để tạo ra bộ chọn này. Ví dụ, luật CSS sau sẽ thiết lập “màu mặt trước” là màu xám cho mọi phần tử trong tài liệu HTML.

\* { color: gray; }

Bộ chọn “mọi phần tử” khá hữu ích trong việc tạo một bộ chọn theo ngữ cảnh (chọn theo các hậu duệ), ví dụ để chọn tất cả các phần tử trong vùng “intro” sử dụng luật CSS sau,

#intro \* { color: gray; }

#### Attribute selector

Có thể tạo bộ chọn dựa vào tên và giá trị thuộc tính của phần tử. Cách làm này giúp tạo ra các bộ chọn rất linh hoạt mà không cần phải thêm thuộc tính *class* hoặc *id* vào các phần tử.

**element[attribute]**

Đây là bộ chọn đơn giản, nó sẽ chọn ra các phần tử với tên thuộc tính cụ thể, không quan tâm tới giá trị của thuộc tính.

Ví dụ, bộ chọn sau sẽ chọn ra tất cả các phần tử *img*, có thuộc tính *title*

img[title] {border: 3px solid;}

**element[attribute=“giá trị”]**

Bộ chọn này sẽ chọn ra các phần tử với tên và giá trị thuộc tính cụ thể. Vì giá trị có phân biệt chữ hoa/chữ thường nên cần xác định giá trị chính xác.

Ví dụ, bộ chọn sau sẽ chọn ra tất cả các phần tử *img*, có tên thuộc tính *title* và giá trị là *“con vịt”*,

img[title="con vịt"] {border: 3px solid;}

**element[attribute~=“giá trị”]**

Bộ chọn này sẽ chọn ra các phần tử với tên và một phần giá trị của thuộc tính.

Ví dụ, bộ chọn sau sẽ chọn ra tất cả các phần tử *img*, mà trong giá trị thuộc tính *title* có chứa từ *“con”*, như vậy, các phần tử *img* với *title* có giá trị là *“con gà”* hay *“con vịt”* đều được chọn.

img[title~="con"] {border: 3px solid;}

**element[attribute|=“giá trị”]**

Bộ chọn này sẽ chọn ra các phần tử với tên thuộc tính và giá trị thuộc tính sẽ bắt đầu bằng một từ cụ thể, sau đó là dấu gạch nối (hyphen) (-).

Ví dụ, bộ chọn sau sẽ chọn ra tất cả các phần tử *img*, mà trong giá trị thuộc tính *title* bắt đầu là từ *“con”*, sau đó là dấu (-), như vậy các phần tử *img* với *title* có giá trị là *“con-gà”* hay *“con-vịt”* đều được chọn.

img[title|="con"] {border: 3px solid;}

**element[attribute^=“phần đầu của giá trị”]**

Bộ chọn này sẽ chọn ra các phần tử, với tên thuộc tính và phần đầu của giá trị được xác định trước.

Ví dụ, bộ chọn sau sẽ chọn ra tất cả các phần tử *img*, mà trong giá trị thuộc tính *src*bắt đầu là chuỗi *“/images/icons”*.

img[src^="/images/icons"] {border: 3px solid;}

**element[attribute$=“phần cuối của giá trị”]**

Bộ chọn này sẽ chọn ra các phần tử, với tên thuộc tính và phần cuối của giá trị được xác định trước.

Ví dụ, bộ chọn sau sẽ chọn ra tất cả các phần tử *img*, mà trong giá trị thuộc tính *src*kết thúc là chuỗi *“.jpg”*.

img[src^=".jpg"] {border: 3px solid;}

**element[attribute\*=“một phần giá trị bất kì”]**

Bộ chọn này sẽ chọn ra các phần tử, với tên thuộc tính và một phần bất kì trong giá trị thuộc tính.

Ví dụ, bộ chọn sau sẽ chọn ra tất cả các phần tử *img*, mà trong giá trị thuộc tính *title*có chứa chuỗi *“on”.* Như vậy các phần tử *img* với *title* có giá trị là *“con-gà”* hay *“con-vịt”* hay *“anh-chon”* đều được chọn.

img[title\*="on"] {border: 3px solid;}

### Declaration

Phần quy định cách thức hiển thị được gọi là declaration (khai báo)

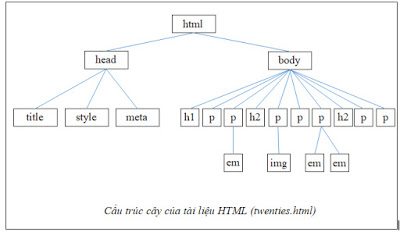
– Phần khai báo bao gồm các cặp *“thuộc tính: giá trị”*(property: value), được bao lại bằng dấu ngoặc nhọn ({})

– Giữa thuộc tính và giá trị có một dấu hai chấm (:) và một khoảng trắng

– Cuối mỗi cặp *“thuộc tính: giá trị”*là dấu chấm phẩy (;)

### Inhertitance

Mặc định, *một số* định dạng CSS đang được áp dụng cho một phần tử cũng sẽ được áp dụng trên các phần tử nằm trong nó.



Để dễ hiểu, có thể xem mối quan hệ giữa các nút trong “cây HTML” như là mối quan hệ trong một dòng tộc, với mỗi phần tử HTML là một thành viên trong dòng tộc.

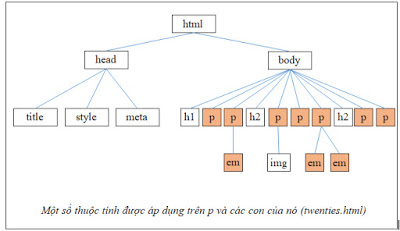
– Mọi phần tử nằm trong một phần tử thì được gọi hậu duệ (descendants) của nó. Ví dụ, các phần tử *h1, h2, p, em, img* được gọi là hậu duệ của phần tử *body*

– Phần tử chứa trực tiếp phần tử khác được gọi là cha (parent), các phần tử được chứa gọi là con (child). Ví dụ, phần tử *p* chứa *em*, nên *p* là cha, *em* là con

– Với một phần tử bất kì, mọi phần tử mở mức cao hơn mà có mối liên hệ nội tộc (…ông-cha-con-cháu…) thì được gọi là tổ tiên (ancestors) của nó. Ví dụ, tổ tiên của *img* là *p*, *body*, *html*

– Hai phần tử có cùng cha thì được gọi là anh em (siblings), Ví dụ, *h1, h2, p* là anh em, vì đều có cha là *body*

Như ở ví dụ đang xét, khi thiết lập định dạng font cho phần tử *p*, thì định dạng đó sẽ tác động trên mọi phần tử *p* của tài liệu HTML và các phần tử văn bản kiểu *inline* (*em*) nằm trong nó. Có nghĩa là phần tử *em* đã kế thừa thuộc tính định dạng font từ phần tử *p*. Tuy nhiên, phần tử *img* cũng nằm trong *p* nhưng sẽ không kế thừa thuộc tính định dạng font, vì hình ảnh thì không có thuộc tính font. Vậy là sẽ có một số thuộc tính được kế thừa và có một số thuộc tính thì không.



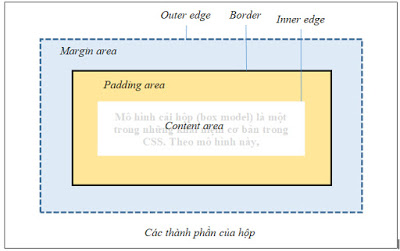
Nói chung, các thuộc tính liên quan đến định dạng văn bản (ví dụ: font-size, color, style) thì sẽ được kế thừa, các thuộc tính liên quan đến borders, backgrounds, margins sẽ không được kế thừa.

### Box model

Mô hình cái hộp là cách mà trình duyệt đã dùng để hiển thị nội dung, nó dùng hình chữ nhật để chứa phần tử HTML, mỗi phần tử HTML được chứa trong một hộp riêng, có bao nhiêu phần tử HTML thì có bấy nhiêu cái hộp (không phân biệt là phần tử kiểu block hay inline).

Với mỗi cái hộp sẽ có các thuộc tính đi kèm, ví dụ: đường biên (border), căn lề ngoài (margin), căn lề trong (padding), nền (background), vị trí (position); lập trình viên sẽ học cách để thiết lập giá trị hợp lý cho các thuộc tính này.

Mặc định, các hộp của phần tử kiểu block có chiều rộng chiếm hết màn hình, trong khi các hộp của phần tử kiểu inline có kích thước vừa đủ để chứa nội dung bên trong.



– Vùng nội dung (content area): là vùng trong cùng của hộp, chứa nội dung của phần tử. Vùng màu trắng chứa văn bản ở hình trên

– Cạnh trong (inner edge): là cạnh của Vùng nội dung, nó là đường phân cách giữa Vùng nội dung và Vùng đệm trong (padding area). Đường này không được hiển thị trên trang web

– Vùng đệm trong (padding area): là khoảng không nằm giữa Vùng nội dung và Đường viền. Vùng đệm trong là tùy chọn (có thể có hoặc không). Ở hình trên, Vùng đệm trong có màu cam vàng

– Đường viền (border): là đường bao quanh phần “nhìn thấy trên giao diện” của phần tử và Vùng đệm trong (nếu có thiết lập). Đường viền là tùy chọn. Đường màu đen ở hình trên

– Vùng đệm ngoài (margin area): là khoảng không thêm vào phía ngoài của Đường viền. Ở hình trên, Vùng đệm ngoài có màu xanh lợt. Tuy nhiên, thực tế, Vùng đệm ngoài luôn trong suốt, do vậy phần nhìn thấy sẽ là màu nền của phần tử cha

– Cạnh ngoài (outer edge): là cạnh của Vùng đệm ngoài, đây chính là cạnh giới hạn toàn bộ không gian của một phần tử trên trang web. Ở hình trên, Cạnh ngoài là đường nét đứt, thực tế, đường này không được hiển thị trên trang web. Như vậy, kích thước một phần tử sẽ bao gồm kích thước của Vùng nội dung, Vùng đệm trong, Đường viền, và Vùng đệm ngoài.

### Styling text

Trong CSS để định dạng phông chữ (font) sử dụng một số thuộc tính phổ biến sau:

– font-family: định phông chữ được dùng

– font-size: định kích thước chữ

– font-style: định kiểu dáng hiển thị

– font-weigth: định độ rộng của nét chữ

– font-variant: biến đổi chữ về dạng in hoa nhưng với kích thước của chữ ở dạng thường

­– font: cách viết gọn

### Styling background

Sử dụng thuộc tính *background-color* để thiết lập màu nền cho các phần tử HTML.

Dưới đây là một số đặc tính của *background-color*,

– Giá trị: *giá trị màu (tên hoặc số) | transparent | inherit*

– Mặc định: *transparent (trong suốt)*

– Áp dụng: *cho mọi phần tử*

– Kế thừa: *không*

Khung nền (canvas) bao gồm: vùng chứa nội dung, khoảng đệm phía trong đường viền (padding), nền chứa đường viền, không bao gồm vùng đệm phía ngoài đường viền (margin).

Thuộc tính *background-color* sẽ đổ màu cho toàn bộ phần khung nền, hay được gọi tắt là nền.

### Typeface, font-family, @font-face

Để chọn kiểu chữ (hay phông chữ), sử dụng thuộc tính *font-family*, với một số đặc tính sau:

– Giá trị (values): tên của phông chữ hoặc họ phông, ngăn cách nhau bằng dấu phẩy | inherit (kế thừa). Đây là một dãy gồm nhiều phông, trình duyệt sẽ tìm các phông này để định dạng cho nội dung, ưu tiên từ trái sang phải, nếu không có, nó sẽ sử dụng phông mặc định.

– Phông mặc định (default): tùy theo trình duyệt

– Áp dụng: cho mọi phần tử (vì có một số thuộc tính chỉ áp dụng cho một số phần tử HTML cụ thể)

– Kế thừa: có (có kế thừa cho các phần tử chứa trong nó)

Ví dụ:

body { font-family: Arial; }

tt { font-family: Courier, monospace; }

p { font-family: “Duru Sans”, Verdana, sans-serif; }

Lưu ý:

– Tên phông chữ phải viết hoa kí tự đầu tiên, ví dụ Arial; họ phông viết thường, ví dụ sans-serif

– Nếu phần giá trị gồm nhiều tên phông, họ phông, thì sử dụng dấu phẩy (,) để ngăn cách

– Tên phông có khoảng trắng thì phải bao lại bằng dấu nháy kép (“”), ví dụ “Duru Sans”

Thông thường, số lượng phông có thể sử dụng là hữu hạn, nó tùy thuộc vào số phông đã được cài đặt trên đĩa cứng của người dùng. Tuy nhiên, lập trình viên cũng có thể sử dụng các phông không có trên đĩa cứng của người dùng bằng việc sử dụng luật *@font-face* của CSS (tìm hiểu thêm về web font).

Để sử dụng web font:

– Lấy web font trực tiếp trên mạng

<link href="https://fonts.googleapis.com/css?family=Open+Sans" rel="stylesheet">

– Sử dụng font cho phần tử HTML

body {

    font-family: "Open Sans", sans-serif;

}

Hoặc,

– Tải web font trên mạng về, để sẵn trong website

– Sử dụng luật @font-face để khai báo và chỉ ra đường dẫn của web font

@font-face {

  font-family: 'Open Sans';

  font-style: normal;

  font-weight: 300;

  src: local('Open Sans Light'), local('OpenSans-Light'),

       url('../vendor/fonts/open-sans-v15-vietnamese\_latin/open-sans-v15-vietnamese\_latin-300.woff2') format('woff2'), /\* Chrome 26+, Opera 23+, Firefox 39+ \*/

       url('../vendor/fonts/open-sans-v15-vietnamese\_latin/open-sans-v15-vietnamese\_latin-300.woff') format('woff'); /\* Chrome 6+, Firefox 3.6+, IE 9+, Safari 5.1+ \*/

}

Styling color

Để thiết lập màu cho văn bản, sử dụng thuộc tính *color*. Dưới đây là một số đặc tính của *color*

– Giá trị: *giá trị màu (tên màu hoặc giá trị số) | inherit*

– Mặc định: *tùy thuộc vào trình duyệt và thiết lập của người dùng*

– Áp dụng: *cho mọi phần tử*

– Kế thừa: *có*

Giá trị của thuộc tính *color* có thể là tên màu, CSS2 cung cấp sẵn 17 tên màu, CSS3 cung cấp sẵn 140 tên màu. Ví dụ một số tên màu,

*Black, white, purple, lime, navy, aqua, silver, maroon, fuchsia, olive, blue, orange, gray, red, green, yellow, teal.*

Giá trị của thuộc tính *color* cũng có thể là giá trị số, bắt đầu là dấu *#*, hoặc *rgb* sau đó là bộ ba giá trị của ba màu cơ bản là Red, Green và Blue (RGB), Ví dụ, màu xám sẽ có giá trị là #666 hoặc #666666.

Để ý là thuộc tính *color* không phải chỉ để thiết lập màu cho chữ, mà theo mô tả của CSS thì nó được sử dụng để thay đổi “màu mặt trước” (foreground color đối lập với màu nền – background) của một phần tử. Mặt trước bao gồm văn bản và đường biên. Do vậy, khi thiết lập màu cho phần tử (bao gồm phần tử *img*) bằng thuộc tính *color* cũng đồng thời thiết lập màu cho đường biên.

### Pseudo-class

Trong CSS, có thể định dạng phần tử dựa trên các trạng thái của 1 phần tử, bằng cách sử dụng bộ chọn *pseudo-class*, gọi nôm na là “lớp giả”.

Sao lại gọi là “lớp giả”? Như đã biết, thuộc tính *class* được sử dụng để gán định danh cho phần tử, các phần tử có cùng giá trị *class* (cùng định danh) sẽ được xem là cùng một lớp. Tuy nhiên, trạng thái của phần tử không được thể hiện tường minh trong mã nguồn, nghĩa là không có định danh, mà nó được xác định ngầm (dựa vào vị trí của phần tử trong mã nguồn hoặc tương tác của người dùng), các phần tử có cùng trạng thái cũng được xem là cùng một lớp, nhưng lớp này không tường minh, nên gọi là “lớp giả”.

Cú pháp của *pseudo-class* thường được bắt đầu bằng tên một phần tử, tiếp theo là dấu hai chấm (:), cuối cùng là tên của trạng thái, ví dụ: *li:first-child*.

Bộ chọn *pseudo-class* có hai loại: bộ chọn *pseudo-class* dựa trên tương tác của người dùng gọi là bộ chọn động, bộ chọn *pseudo-class* dựa trên vị trí của phần tử trong mã HTML được gọi là bộ chọn tĩnh.

-Pseudo-class động:

Bộ chọn pseudo-class cho liên kết:

*:link* dùng để định dạng cho liên kết chưa được người dùng bấm vào

*:visited* dùng để định dạng cho liên kết đã được người dùng bấm vào

Bộ chọn pseudo-class dựa trên thao tác người dùng:

*:focus*  áp dụng khi người dùng chọn phần tử để nhập nội dung, phần tử được chọn sẽ nổi bật hơn so với các phần tử còn lại, bộ chọn này hay áp dụng cho các ô nhập liệu của form

*:hover* áp dụng khi rê con trỏ chuột trên một phần tử, mặc dù có thể áp dụng trên mọi phần tử HTML, tuy nhiên, bộ chọn này hay được áp dụng trên các liên kết, nhằm gây chú ý cho người sử dụng.

*:active* áp dụng khi một phần tử (nút hoặc liên kết) đang được bấm chuột.

-Pseudo-class tĩnh:

Bộ chọn *pseudo-class* tĩnh sẽ chọn ra các phần tử dựa vào vị trí của nó trong cấu trúc tài liệu HTML. Dưới đây là ví dụ một số bộ chọn:

*:root, :empty, :first-child, :last-child, :only-child*

*:first-of-type, :last-of-type, :only-of-type, :nth-child()*

*:nth-last-child(), :nth-of-type(), :nth-last-of-type()*

### Pseudo-element

Không chỉ có bộ chọn *pseudo-class* (“lớp giả”), CSS còn có các bộ chọn *pseudo-element* (“phần tử giả”). Bộ chọn *pseudo-element* không chọn các phần tử trong tài liệu HTML, cũng không dựa vào trạng thái của phần tử, mà nó sẽ chọn các đối tượng “giả”, dựa vào vị trí hiển thị của các đối tượng trên cửa sổ trình duyệt. Có bốn bộ chọn *pseudo-element* gồm, *: :first-line, : :first-letter, ::before* và *::after*.

Lưu ý, trong CSS3, để phân biệt giữa *pseudo-class*và*pseudo-element,*khi viết *pseudo-class* sẽ sử dụng một dấu hai chấm (:), trong khi *pseudo-element* sử dụng hai dấu hai chấm (::). Trong CSS2 và CSS1, cả hai bộ trọn này đều sử dụng một dấu hai chấm (:).

**Bộ chọn ::first-line**

Bộ chọn *::first-line* được sử dụng để chọn hàng đầu tiên của một phần tử văn bản (khi văn bản hiển thị trên trình duyệt). Tuy nhiên, với bộ chọn này, chỉ có thể áp dụng được các thuộc tính sau,

*color, font, background*

*word-spacing,  letter-spacing, text-decoration*

*vertical-align, text-transform, line-height*

**Bộ chọn ::first-letter**

Bộ chọn *::first-letter* được sử dụng để chọn kí tự đầu tiên của một phần tử văn bản (khi văn bản hiển thị trên trình duyệt). Tuy nhiên, với bộ chọn này, chỉ có thể áp dụng được các thuộc tính sau,

*color, font, text-decoration*

*text-transform, vertical-align, padding*

*background, margin, line-height*

*border, float, letter-spacing, word-spacing*

**Bộ chọn ::before**

Bộ chọn *::before* được sử dụng để chèn thêm nội dung vào trước một phần tử mà không cần thêm nội dung đó vào mã nguồn (cái này được gọi là thêm nội dung bằng CSS).

**Bộ chọn :after**

Bộ chọn *::after* được sử dụng để chèn thêm nội dung vào sau một phần tử mà không cần thêm nội dung đó vào mã nguồn (cái này được gọi là thêm nội dung bằng CSS).

### Normal flow

Thông thường, các phần tử sẽ được hiển thị theo cách sau:

– Các phần tử dạng văn bản được hiển thị từ trên xuống dưới, từ trái qua phải (không tính các ngôn ngữ hiển thị theo hướng từ phải sang trái) theo đúng thứ tự xuất hiện trong mã nguồn

– Các phần tử kiểu block sẽ được xếp chồng từng khối, theo thứ tự xuất hiện trong mã nguồn, với chiều rộng bằng với cửa sổ trình duyệt hoặc bằng với phần tử chứa nó

– Các phần tử kiểu inline và các phần tử dạng văn bản được xếp tuần tự theo hàng, nằm trong phần tử block chứa nó

– Khi cửa sổ trình duyệt hoặc phần tử block có thay đổi về kích thước thì kích thước các phần tử block bên trong cũng thay đổi theo, và các phần tử inline bên trong cũng sẽ tự sắp xếp lại cho vừa với phần tử block chứa nó

### Float

Thuộc tính float (trôi dạt) sẽ dịch chuyển một phần tử về phía trái hoặc phía phải xa nhất có thể và cho phép các nội dung phía sau bao phủ xung quanh nó.

Float là kĩ thuật chính được sử dụng để tạo ra bố cục nhiều cột, thanh điều hướng, biểu diễn dữ liệu giống dạng bảng.

Dưới đây là một số đặc tính khi một phần tử được thiết lập float:

– Một phần tử khi được thiết lập float sẽ giống với một hòn đảo giữa dòng nước, nghĩa là nó không thuộc về dòng nước, nhưng nó nằm giữa dòng nước, dòng nước (nội dung) sẽ nhường cho nó một chỗ đứng, và chảy xung quanh nó. Đây là đặc điểm duy nhất chỉ có ở các phần tử được thiết lập float

– Một phần tử chỉ trôi dạt (phải/trái) trong vùng nội dung của phần tử chứa nó, nghĩa là không được trôi vào vùng padding

– Khi một phần tử trôi dạt, margin của nó vẫn được bảo toàn, nghĩa là toàn bộ không gian của phần tử vẫn được bảo toàn

### Clear

Để ngắt ảnh hưởng của chế độ float, nhằm đưa văn bản bao quanh về chế độ hiển thị thông thường, sử dụng thuộc tính *clear*.

Khi gắn thuộc tính *clear* vào phần tử nào thì phần tử đó sẽ không còn bị tác động bởi thuộc tính float đã được thiết lập ở phía trên, nghĩa là, nó không nằm cạnh phần tử float, mà sẽ được tách xuống một vùng mới, nằm dưới phần tử float.

Lưu ý, thuộc tính *clear* được gắn vào phần tử bên dưới, chứ không gắn vào chính phần tử float. Phần tử nào có giá trị clear là *left* sẽ giúp nó không còn bị ảnh hưởng bởi chế độ *float: left* được thiết lập ở phía trên nó. Phần tử nào có giá trị clear là *right* sẽ giúp nó không còn bị ảnh hưởng bởi chế độ *float: right* được thiết lập ở phía trên nó. Nếu phía trên có nhiều phần tử được float (có thể cả left và right), thì thiết lập giá trị cho clear là *both* để đảm bảo chắc chắn không còn bị ảnh hưởng bởi chế độ float nữa.

### Positioning

Để chọn kiểu định vị cho một phần tử, sử dụng thuộc tính *position.*

Giá trị của thuộc tính *position* là các từ khóa, mỗi từ khóa ứng với một kiểu định vị, các phần sau sẽ tìm hiểu chi tiết về mỗi trường hợp, ở đây chỉ mô tả tóm tắt từng kiểu.

– *static*: đây là kiểu định vị thông thường, kiểu mặc định, trong đó, các phần tử sẽ được hiển thị theo đúng thứ tự của chúng trong mã nguồn. Từ *static* có nghĩa là tĩnh tại, nằm tại một vị trí, không di chuyển hay thay đổi vị trí.

– *relative*: phần tử được định vị dựa vào vị trí gốc (vị trí theo thứ tự trong mã nguồn) của nó. Vị trí gốc của phần tử trên giao diện sẽ vẫn được dành riêng cho nó, là một vùng không gian trống. Vị trí mới của phần tử được xác định dựa vào khoảng dời (offset distance) so với vị trí gốc. Từ *relative* có nghĩa là tương đối, nghĩa là vị trí mới sẽ là dịch chuyển một khoảng tương đối so với vị trí gốc.

– *absolute*: phần tử sẽ được tách khỏi luồng hiển thị thông thường, và vị mới của phần tử được xác định dựa theo cửa sổ trình duyệt hoặc phần tử chứa nó. Khác với kiểu *relative*, ở kiểu này, vùng không gian gốc của phần tử sẽ bị thu hồi, và nó thực sự không ảnh hưởng tới bố cục chung của các phần tử xung quanh.

– *fixed*: phần tử sẽ được tách khỏi luồng hiển thị thông thường, và vị trí mới của phần tử được xác định dựa theo cửa sổ trình duyệt chứ không dựa theo phần tử khác trong tài liệu. Khi người dùng cuộn nội dung tài liệu, thì phần tử định vị theo kiểu fixed vẫn nằm nguyên tại vị trí của nó trên cửa sổ trình duyệt.

Mỗi kiểu định vị có mục đích sử dụng riêng, tuy nhiên, kiểu absolute đa năng hơn cả. Với kiểu absolute, người thiết kế có thể đặt một đối tượng ở vị trí bất kì trong cửa sổ, hoặc trong một phần tử khác, có thể tạo bố cục nhiều cột, hoặc tạo các định dạng hiển thị đa dạng khác.

### Layout

Vì trang web sẽ được hiển thị trên các màn hình có kích thước khác nhau, từ màn hình rất nhỏ của điện thoại thông minh, đến các màn hình máy tính, thậm chí là màn chiếu rất lớn. Ngoài ra, việc thay đổi kích thước của phông chữ cũng làm ảnh hưởng rất nhiều đến bố cục chung của trang web. Hiện nay, có rất nhiều loại bố cục cho một trang web. Mỗi loại bố cục có những ưu và nhược điểm riêng.

Dưới đây là một số loại bố cục phổ biến,

– Fixed layout: bố cục có kích thước cố định, cho dù kích thước màn hình hoặc kích thước văn bản có thay đổi

– Fluid (liquid) layout: bố cục thay đổi theo kích thước của màn hình

– Elastic layout: bố cục thay đổi theo kích thước của văn bản

– Hybrid layout: gồm những vùng thay đổi và những vùng cố định

#### Fixed layout

Bố cục kiểu cố định (fixed layout), như tên gọi của nó, là bố cục mà các thành phần có chiều rộng được người thiết kế xác định cụ thể, luôn cố định, đơn vị tính là px. Với kiểu bố cục này, người thiết kế sẽ áp đặt luôn: mối liên hệ giữa các thành phần của trang, việc căn lề, độ dài của một hàng (line length).

Kiểu bố cục cố định đã từng được sử dụng rộng rãi, khi kích thước màn hình tương đối đồng nhất, khi mà mọi người đều sử dụng máy tính (desktop) để duyệt web. Tuy nhiên, hiện nay, phương tiện dùng để duyệt web rất đa dạng, với nhiều kích thước màn hình khác nhau, vì vậy, kiểu bố cục cố định cũng không còn được sử dụng nhiều.

Để tạo bố cục cố định, việc đầu tiên cần làm là xác định độ rộng của trang web. Độ rộng của trang web được xác định dựa trên độ phân giải của màn hình. Ví dụ, độ rộng phổ biến của trang web thường được chọn là 960px, hoặc có thể tăng hoặc giảm một chút. Việc thứ hai cần làm là trang web sẽ căn trái hay căn giữa so với màn hình? Nếu căn trái, phần không gian trống (extra space) sẽ nằm ở phía phải, nếu căn giữa, phần không gian trống sẽ được chia đều sang hai bên.

#### Fluid layout

Kiểu fluid còn có tên khác là liquid, chữ fluid hay liquid đều có nghĩa là chất lỏng. Vì là chất lỏng nên nó dễ dàng thay đổi hình dạng theo “bình chứa”.

Trong bố cục kiểu fluid, vùng nội dung và các cột bên trong có thể thay đổi kích thước (rộng hơn hoặc hẹp hơn) để chiếm hết không gian hữu dụng (available) của màn hình. Nội dung được hiển thị một cách tự nhiên, không quy định “cứng” độ rộng hoặc ngắt hàng. Khi kích thước của phần tử thay đổi, văn bản bên trong sẽ được động sắp xếp lại tùy theo kích thước của đối tượng chứa nó. Ví dụ, trang web của W3C (www.w3.org) có bố cục kiểu fluid; thử thay đổi kích thước của cửa sổ trình duyệt để quan sát sự thay đổi của vùng nội dung và các cột giao diện.

Kiểu bố cục fluid là nền tảng của kĩ thuật thiết kế giao diện tùy biến theo kích thước màn hình (responsive web design). Hiện nay, người dùng sử dụng rất nhiều thiết bị với kích thước màn hình đa dạng để duyệt web. Vì vậy, mọi người đang hướng đến các thiết kế có độ tùy chỉnh cao, và kiểu bố cục fluid đang được quan tâm nhiều.

Để tạo bố cục kiểu fluid, sử dụng đơn vị % để xác định độ rộng (thuộc tính *width*) của các thành phần. Trong một số trường hợp có thể không cần thiết lập giá trị cho thuộc tính *width*, khi đó giá trị mặc định sẽ được sử dụng là *auto*, và trình duyệt sẽ tự động hiển thị các phần tử sao cho hợp lý trong cửa sổ trình duyệt hoặc trong phần tử chứa.

#### Elastic layout

Từ elastic có nghĩa là đàn hồi, co giãn. Trong bố cục này, cỡ chữ và phần tử chứa nó sẽ được co giãn cùng nhau. Nếu chữ lớn hơn, phần tử chứa nó sẽ rộng ra; nếu chữ nhỏ hơn, phần tử chứa nó sẽ co lại, kết quả là số chữ trên một hàng (line length) sẽ không bị thay đổi.

Kiểu bố cục elastic khắc phục được hai vấn đề, một là tình trạng hàng quá dài (nhiều chữ), khó đọc của kiểu bố cục fluid; hai là tình trạng quá ít chữ trên một hàng (khi phóng to chữ) của kiểu bố cục cố định (fixed).

Để tạo bố cục kiểu elastic, cái quan trọng nhất chính là đơn vị đo. Bố cục elastic sử dụng đơn vị đo là *em*, đây là đơn vị đo tính theo kích thước phông chữ. Ví dụ, phần tử có kích thước phông là 16px thì 1em tương đương với 16px, vậy 2em sẽ là 32px, 1.5em sẽ là 24px. Như hình phía trên, cả hai trang web (bên trái, bên phải) đều có kích thước là 48em, tuy nhiên độ lớn của giao diện lại khác nhau, lý do là phông chữ của hai trang có kích thước khác nhau.

#### Hybrid layout

Tới phần này, nếu để ý sẽ thấy ba kiểu tạo bố cục thực ra là ba kiểu dùng đơn vị đo cho các phần tử. Kiểu fixed dùng đơn vị đo px, kiểu fluid dùng đơn vị đo % và kiểu elastic dùng đơn vị đo là *em*.

Vậy nếu kiểu bố cục nào mà kết hợp các kiểu đơn vị đo trên thì được gọi là bố cục kiểu hybrid. Chữ hybrid có nghĩa là hỗn hợp, kết hợp.

Trong nhiều trường hợp, việc kết hợp những vùng nội dung cố định và những vùng nội dung co dãn trong một trang web là rất quan trọng.

### Flexbox

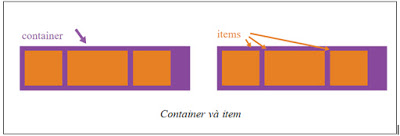
Flexbox là viết rút gọn của flexible box, từ flexible nghĩa là linh hoạt. Flexbox thường được sử dụng để tạo bố cục cho các thành phần của một ứng dụng web hoặc các thành phần có kích thước nhỏ của trang web.

Ý tưởng của flexbox là sắp xếp các mục (item) trong một thùng chứa (container), các item có thể tự động thay đổi kích thước cho vừa với container. Nghĩa là không cần thiết lập kích thước cho item, không cần float, các item sẽ tự điều chỉnh việc hiển thị sao cho vừa gọn trong container.

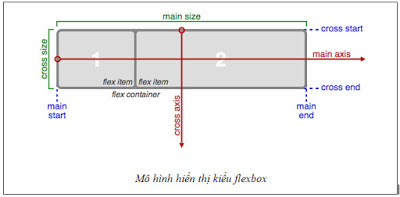
Điểm khác biệt quan trọng nhất của flexbox so với các kiểu bố cục truyền thống, là tính linh hoạt của hướng hiển thị (direction-agnostic). Ở kiểu bố cục truyền thống, một phần tử kiểu block sẽ hiển thị theo hướng từ trên xuống dưới (vertical), một phần tử kiểu inline sẽ hiển thị theo hướng ngang (horizontal). Với flexbox, người lập trình có thể thay đổi hướng hiển thị theo chiều ngang hoặc chiều dọc. Lưu ý: flexbox chỉ sắp xếp các phần tử theo một chiều (one-dimension), hoặc là chiều ngang hoặc là chiều dọc.

Flexbox không phải là một thuộc tính đơn lẻ, mà nó là cả một mô-đun, với nhiều thành phần và thuộc tính đi kèm.

Hai thành phần cơ bản của flexbox là container (còn gọi là parent element) và các mục con bên trong gọi là item (hay children).



Nếu bố cục truyền thống hoạt động dựa trên các đặc tính hiển thị của phần tử kiểu block và inline thì flexbox hiển thị dựa trên một mô hình có tên là “flex-flow directions”.



Trong đó:

– *main-axis*: hướng ngang của container, mặc định, các item sẽ được xếp theo chiều ngang, tuy nhiên, có thể thay đổi hướng bằng thuộc tính flex-direction

– *main-start, main-end*: các item sẽ được đặt trong container, theo hàng, hướng từ *main-start* tới *main-end*

– *main-size*: kích thước (chiều rộng hoặc chiều cao) của item tính theo *main axis*

– *cross-size*: kích thước (chiều rộng hoặc chiều cao) của item tính theo *cross axis*

– *cross-start, cross-end*: các hàng sẽ được đặt theo thứ tự từ *cross-start* tới *cross-end*

Để tạo ra một container, sử dụng thuộc tính *display: flex;*

.container {

            display: flex;

}

Khi đó, các phần tử bên trong container sẽ trở thành item, và hiển thị theo mô hình flexbox. Từ đây có thể sử dụng rất nhiều các thuộc tính để định dạng container và item.

Dưới đây là một số ví dụ về cách sử dụng các thuộc tính,

– Mặc định, các item sẽ xuất hiện theo chiều ngang, từ trái sang phải; để đổi chiều từ phải sang trái, sử dụng thuộc tính *flex-direction: row-reverse* cho container

.container {

            display: flex;

            flex-direction: row-reverse;

}

– Để các item xuất hiện theo chiều dọc, sử dụng thuộc tính *flex-direction: column*

.container {

            display: flex;

            flex-direction: column;

}

– Để thay đổi thứ tự xuất hiện của các item, sử dụng thuộc tính *order: value*, với value là giá trị để xác định vị trí của item trong nhóm. Giá trị value mặc định của các item đều là 0, khi đó, thứ tự hiển thị sẽ dựa theo mã nguồn (HTML). Khi có thiết lập value, item nào có value lớn sẽ đứng sau. Trong trường hợp item cùng giá trị value, thì thứ tự xuất hiện sẽ dựa vào thứ tự trong mã nguồn.

Ví dụ, với hai lệnh CSS dưới đây, thứ tự xuất hiện của các item sẽ là: Mục 1 > Mục 4 > Mục 2 > Mục 3

.item2 {

            order: 1;

}

.item3 {

            order: 2;

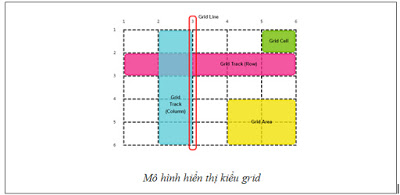
}

Để làm việc với flexbox còn có rất nhiều các thuộc tính khác. Ví dụ các thuộc tính áp dụng cho container: display, flex-direction, flex-wrap, flex-flow, justify-content, align-item, align-content. Ví dụ các thuộc tính áp dụng cho item: order, flex-grow, flex-shrink, flex-basis, flex, align-self.

### Grid system

Khác với flexbox, là kĩ thuật tạo bố cục một chiều và thường được sử dụng để tạo bố cục cho các vùng nhỏ; grid là kĩ thuật tạo bố cục hai chiều (two-dimension), có thể thao tác theo cả hàng và cột. Grid thường được sử dụng để tạo bố cục cho toàn trang hoặc cho các thành phần nhỏ trên trang web.

Cũng giống với kĩ thuật flexbox, hệ thống grid cũng gồm hai thành phần và các thuộc tính. Hai thành phần gồm phần tử cha (parent element), phần tử chứa, được gọi là container và phần tử con, là các phần tử nằm bên trong, được gọi là item.



Grid định nghĩa một hệ thống lưới gồm các đường (line) ngang và dọc. Khoảng không gian được giới hạn bởi các đường ngang/dọc được gọi là hàng/cột, hay được gọi chung là track.

Phần giao nhau của một hàng và cột gọi là ô (cell), tập hợp nhiều ô gọi là một vùng (area).

Độ rộng của track có thể được thiết lập cố định bằng px, cũng có thể thiết lập tương đối bằng %, hoặc bằng đơn vị fr. Fr là viết tắt của fraction, nghĩa là một phần không gian của container.

Có thể đặt các item vào vị trí chính xác trên grid bằng tên hàng, số hàng hoặc địa chỉ của một vùng.

### Reset CSS

Như đã biết, mỗi trình duyệt luôn có mã CSS riêng, dùng để định dạng các phần tử HTML, gọi là user agent style sheets. Ví dụ, với phần tử *h1*, mặc dù lập trình viên không thiết lập CSS, thì trình duyệt vẫn định dạng *h1* với phông chữ lớn, đậm, và khoảng cách dòng lớn hơn bình thường. Tuy nhiên, tùy theo mỗi trình duyệt, *h1* sẽ có độ lớn, độ đậm và khoảng cách dòng khác nhau, kết quả là chúng ta sẽ có một giao diện không ổn định. Ngoài ra, ngay cả khi người lập trình có thiết lập CSS, thì các phần tử HTML vẫn kế thừa một số thuộc tính từ CSS mặc định, điều này gây ra các kết quả ngoài dự tính.

Vì các lý do trên, các lập trình viên thường phải thực hiện reset CSS của trình duyệt, mục đích là để ghi đè các thiết lập CSS mặc định, để mọi thứ trở về với trạng thái ban đầu. Khi đó, người lập trình phải thiết lập một cách tường minh (explicitly) các thuộc tính của phần tử, như: font, text, margin, padding. Đổi lại, người lập trình sẽ đảm bảo luôn kiểm soát được các tác động của CSS lên giao diện.

Việc reset CSS không phải là bắt buộc, vì nếu muốn dựa vào các định dạng CSS có sẵn của trình duyệt, để không phải mất công viết lại từ đầu, thì không nhất thiết phải reset CSS. Chỉ khi nào muốn tất cả mã CSS đều được kiểm soát thì nên reset CSS.

### Sprite

Nghĩa thông thường của sprite là tiên nữ. Trong đồ họa máy tính, sprite là một hình ảnh hai chiều hoặc hình ảnh động được tích hợp vào một cảnh lớn hơn.

Trong kĩ thuật CSS, sprites là một cách để cải thiện hiệu suất tải và hiển thị trang web, bằng cách kết hợp nhiều hình ảnh nhỏ hoặc biểu tượng thành một hình ảnh lớn hơn gọi là dải sprite hoặc bộ tile, và sử dụng CSS để chọn tường ảnh nhỏ/biểu tượng, hiển thị trên trang web. Mỗi ảnh nhỏ/biểu tượng gọi là một sprite (một cô tiên).

Có rất nhiều các công cụ giúp tạo sprites tự động, vào google, gõ từ khóa “sprite tool” hoặc “CSS sprite generator” để tìm hiểu và học thêm về các công cụ này.

## JavaScript