[Bài đọc] Tổng quan về Git

Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Khái niệm về Git
* Các khái niệm quan trọng của Git như:
  + Repository
  + Local Repository
  + Remote Repository
  + Version (Phiên bản
  + Quản lý Phiên bản mã nguồn
  + Đồng bộ giữa các Repository

Nội dung

Git là gì?

Git là một cách thức (mô hình, giao thức, công cụ, qui tắc) để lưu trữ mã nguồn và hỗ trợ cộng tác trong quá trình phát triển ứng dụng.

Hãy tưởng tượng, Bob và James cùng tham gia phát triển một dự án. Họ lưu trữ mã nguồn của dự án ở đâu? Làm thế nào để họ cộng tác với nhau? Làm thế nào để họ chia sẻ chung một bộ mã nguồn? Git có thể giúp Bob và James trong trường hợp này.

Các khái niệm quan trọng trong Git

**Repository**: Là nơi chứa mã nguồn (repository có nghĩa là "kho chứa"), tức là nơi mà chúng ta lưu trữ các file, các đoạn mã được viết ra trong suốt quá trình phát triển dự án. Có 2 loại Repository là Local Repository và Remote Repository.

**Local Repository**: Là nơi lưu trữ mã nguồn trên máy tính của lập trình viên. Chẳng hạn Bob và James mỗi người đều có các Local Repository trên máy của riêng mình. Các đoạn mã do Bob hoặc James viết ra đều được lưu trữ trên các Local Repository đó.

**Remote Repository**: Là nơi lưu trữ mã nguồn ở trên mạng (chẳng hạn là mạng Internet hay là mạng nội bộ của một công ty). Để có Remote Repository thì chúng ta cần có một máy chủ và cài Git Server lên đó. Hoặc chúng ta có thể sử dụng các dịch vụ miễn phí (chẳng hạn như GitHub.com, GitLab.com...). Remote Repository (có nghĩa là "kho chứa từ xa") là nơi mà cả Bob và James đều có thể truy cập được để tải mã nguồn về hoặc đưa mã nguồn lên đó.

**Version (Phiên bản)**: Là trạng thái mã nguồn ở một thời điểm nhất định nào đó. Chẳng hạn, sáng hôm nay Bob hoàn thành xong tính năng Đăng nhập của website, Bob quyết định đánh dấu rằng đây là một phiên bản của mã nguồn. Đến buổi trưa, James hoàn thành tính năng Cài đặt lại mật khẩu, James lại quyết định đánh dấu rằng đây là một phiên bản nữa của mã nguồn. Như vậy, có thể thấy rằng trong quá trình phát triển một dự án thì mã nguồn sẽ có rất nhiều phiên bản được tạo ra liên tiếp nhau. Đến buổi chiều, chẳng may Bob lỡ tay xoá mất mấy đoạn mã quan trọng. Nhưng Bob không hề lo lắng, bởi vì Bob có thể "quay trở về" phiên bản trước đó của mã nguồn một cách dễ dàng. Đó là một trong rất nhiều những lợi ích của việc quản lý phiên bản mã nguồn.

**Quản lý Phiên bản mã nguồn**: Git không chỉ giúp chúng ta quản lý mã nguồn mà còn quản lý phiên bản mã nguồn. Tức là giúp chúng ta làm chủ được từng thay đổi của mã nguồn theo thời gian.

**Đồng bộ giữa các Repository**: Ngoài Repository chung (chính là Remote Repository) thì Bob và James đều có các Repository riêng của mình (chính là Local Repository), do đó phải có cơ chế để đồng bộ giữa các Repository này với nhau, tức là giữ cho mã nguồn ở các Repository này giống hệt nhau. Hay nói cách khác, tất cả các Repository này là những bản sao của nhau.

Để làm được việc này, chúng ta cần biết được cách vận hành của Git.

Kết luận

Sau bài đọc, học viên đã được cung cấp những kiến thức sau:

* Khái niệm về Git
* Các khái niệm quan trọng của Git như:
  + Repository
  + Local Repository
  + Remote Repository
  + Version (Phiên bản
  + Quản lý Phiên bản mã nguồn
  + Đồng bộ giữa các Repository

## [Bài đọc] Các bước làm việc cơ bản với Git

## Mục tiêu

Bài đọc mô phỏng lại hai đối tượng là Bob và James sử dụng Git để quản lý mã nguồn và cộng tác với nhau.

## Nội dung

Bob và James bắt đầu viết mã cho dự án của mình, họ quyết định sử dụng dịch vụ miễn phí của GitHub.com. Các bước đơn giản nhất mà họ làm là như sau:

1. Tạo tài khoản ở trang [http://github.com](http://github.com/). Tài khoản được tạo qua gmail.

2. Tạo Repository cho dự án trên trang http://github.com. Đây chính là **Remote Repository**.

3. Bob nhân bản (tiếng Anh gọi là ***clone***) cái Repository vừa tạo trên GitHub về máy tính của mình. Đây chính là **Local Repository** của Bob.

4. James sao chép Repository trên GitHub về máy tính của mình. Đây chính là **Local Repository** của James.

5. Bob viết các đoạn mã trên máy của mình, rồi lưu lại trong Local Repository. Trong lúc này James cũng có thể ngồi viết mã trên máy của mình rồi lưu lại trong Local Repository của mình mà không ảnh hưởng gì tới Bob.

6. Bob đẩy (tiếng Anh gọi là ***push***) các đoạn mã từ Local Repository lên trên Remote Repository để chia sẻ với James. Khi hoàn thành bước này thì Local Repository trên máy của Bob và Remote Repository là **giống hệt nhau**.

7. James kéo (tiếng Anh gọi là ***pull***) các đoạn mã mà Bob vừa viết từ Remote Repository về máy của mình. Giờ đây trên máy của James có cả những đoạn mã do chính anh viết và những đoạn mã do Bob viết.

8. Lúc này James có thể đẩy các thay đổi từ Local Repository của mình lên Remote Repository để chia sẻ cùng Bob. Khi hoàn thành bước này thì Local Repository trên máy của James và Remote Repository là giống hệt nhau.

9. Bob kéo các thay đổi từ Remote Repository về máy của mình. Khi hoàn thành bước này thì **cả 3 Repository đều giống hệt nhau**.

Sau đó, Bob và James cứ lặp đi lặp lại công việc của mình như được mô tả từ bước 5 đến bước 9.

Trật tự và chi tiết một số bước có thể khác, nhưng về cơ bản thì luồng công việc của Bob và James cứ diễn ra như thế.

## Kết luận

Sau bài đọc, học viên đã hình dung được việc quản lý mã nguồn và cộng tác trên Git như thế nào trong một nhóm có ít nhất là 2 người.

## [Bài đọc] Một trang web cơ bản và các thẻ HTML

## Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Ý nghĩa của các thẻ HTML cơ bản
* Giới thiệu về các trình duyệt
* Cấu trúc của một trang HTML

## Nội dung

#### **Một trang web đơn giản**

Một trang web đơn giản có mã nguồn như sau:

Trong đoạn mã trên:

* Khai báo <!DOCTYPE html> quy định rằng đây là một tài liệu HTML5
* Phần tử <html> là phần tử gốc của trang HTML
* Phần tử <head> chứa thông tin mô tả về tài liệu hiện tại
* Phần tử <title> chứa tiêu đề của tài liệu
* Phần tử <body> chứa toàn bộ nội dung của trang web (những gì mà chúng ta nhìn thấy được trên trình duyệt)
* Phần tử <h1> định nghĩa một tiêu đề lớn
* Phần tử <p> định nghĩa một đoạn văn.

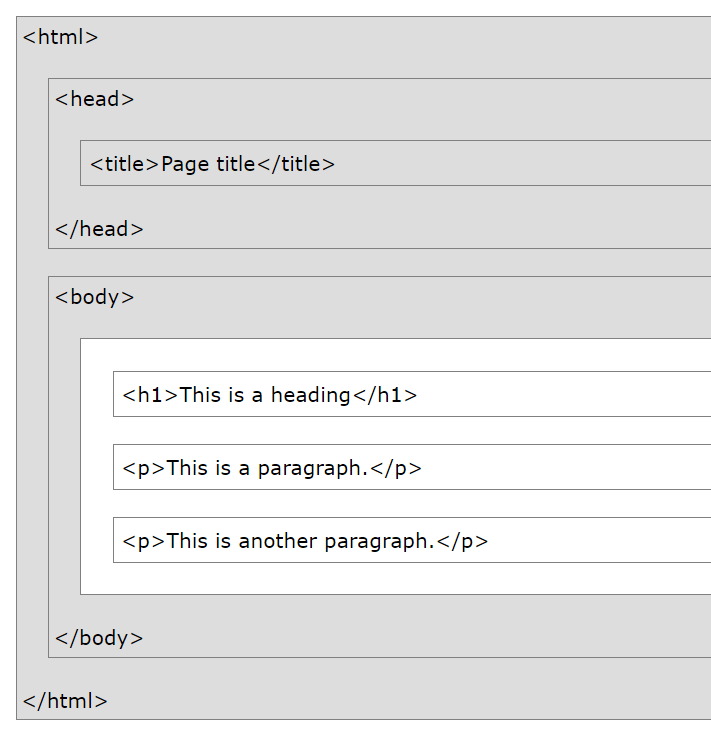
#### **Trình duyệt**

Trình duyệt (chẳng hạn như Chrome, IE, Firefox, Safari...) là những phần mềm được sử dụng để đọc các tài liệu HTML và hiển thị chúng dưới dạng giao diện người dùng.

Trình duyệt không hiển thị các thẻ HTML ra giao diện, thay vào đó chúng dựa vào các thẻ HTML để biết phải hiển thị những thành phần tương ứng ra giao diện. Chẳng hạn ở ví dụ trên, khi trình duyệt đọc được một thẻ <h1> thì chúng sẽ hiển thị ra một tiêu đề lớn, còn khi đọc được một thẻ <p> thì lại hiển thị một đoạn văn bản.

#### **Cấu trúc của một trang HTML**

Cấu trúc của một trang HTML được mô tả như sau:



Lưu ý: Chỉ có phần nội dung ở bên trong thẻ <body> mới được hiển thị trên trình duyệt.

Các thẻ <html>, <head>, <title>, <body>  là bắt buộc với tất cả các tài liệu HTML. Ngoài ra chúng ta có thể thêm nhiều thẻ khác để tạo nội dung cho trang web.

## Kết luận

Sau bài đọc, học viên đã được cung cấp những kiến thức sau:

* Ý nghĩa của các thẻ HTML cơ bản
* Giới thiệu về các trình duyệt
* Cấu trúc của một trang HTML

## [Bài đọc] Thẻ HTML và thuộc tính

## Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Cấu trúc của một thẻ HTML
* Một số thuộc tính nổi bật của một vài thẻ HTML cơ bản
* Lưu ý khi viết tên các thẻ HTML
* Phân biệt dấu nháy đơn và dấu nháy kép

## Nội dung

#### **Thẻ HTML**

Thẻ HTML là tên gọi của các phần tử, được sử dụng để tạo nên một tài liệu HTML. Hay nói cách khác, một trang web được tạo nên từ các thẻ HTML.

Thẻ HTML bao gồm tên thẻ được đặt trong 2 dấu ngoặc nhọn:

**<tên\_thẻ>Nội dung của thẻ</tên\_thẻ>**

**Ghi nhớ:**

* Các thẻ HTML thường bao gồm một cặp, chẳng hạn như <p> và  </p>
* Thẻ đầu tiên được gọi là **thẻ mở**, thẻ thứ hai được gọi là **thẻ đóng**.
* Thẻ đóng được viết như thẻ mở, nhưng có thêm một dấu xuyệc trái (/) đặt ở trước tên.

Một số thẻ cơ bản:

|  |  |  |
| --- | --- | --- |
| **Thẻ bắt đầu** | **Nội dung của thẻ** | **Thẻ kết thúc** |
| <h1> | My First Heading | </h1> |
| <p> | My first paragraph. | </p> |
| <br> |  |  |

**Thuộc tính của thẻ**

Thuộc tính cung cấp các thông tin bổ sung cho các thẻ HTML

* Các thẻ HTML có thể khai báo thêm các thuộc tính
* Các thuộc tính cung cấp thêm các thông tin cần thiết cho thẻ
* Thuộc tính luôn được viết trong thẻ mở
* Thuộc tính đi thành cặp thuộc\_tính/giá\_trị dạng: tên\_thuộc\_tính="giá\_trị"

##### ***Thuộc tính title***

Thuộc tính **title** dùng để định nghĩa tiêu đề của một thành phần HTML. Khi duyệt web, bạn di chuột tới thành phần nào của trang, một *tooltip* sẽ xuất hiện nếu bạn có khai báo thuộc tính **title**, nội dung chính là giá trị khi khai báo.

Bạn hãy thử di chuột lên tiêu đề **"Thuộc tính title"** bên trên.

Ví dụ:

<p title="I'm a tooltip">  
 This is a paragraph.  
</p>

##### ***Thuộc tính href***

Thẻ**<a>** dùng để khai báo một liên kết trên trang web. Muốn khai báo địa chỉ đích của liên kết ta viết thêm thuộc tính **href** cho nó.

Ví dụ:

<a href="https://www.w3schools.com">This is a link</a>

##### ***Thuộc tính về kích thước***

Thuộc tính **width** và **height** dùng để khai báo kích thước của đối tượng, như của hình ảnh chẳng hạn.

* width: chiều rộng
* height: chiều cao

Ví dụ:

<img src="w3schools.jpg" width="104" height="142">

Trong ví dụ trên hình ảnh có độ rộng là 104px và cao là 142px (px:pixel - điểm ảnh, một đơn vị đo phổ biến khi thiết kế web).

##### ***Thuộc tính alt***

Thuộc tính **alt** dùng để khai báo một đoạn văn bản thay thể (alternative) khi vì lí do nào đó, một thành phần của HTML không được hiển thị lên trang web.

Thuộc tính này cũng được đọc bởi các chương trình đọc màn hình (screen reader), dành cho những người khiếm thị (chỉ nghe được).

<img src="w3schools.jpg" alt="W3Schools.com" width="104" height="142">

**Lưu ý:**

*Luôn viết thuộc tính bằng chữ in thường*

* HTML không bắt buộc bạn phải viết tên thuộc tính in thương hay in hoa. Bạn có thể viết width hoặc WIDTH, thậm chí WiDth đều được.
* Tuy nhiên, viết mã HTML bằng chữ in thường là tốt nhất.

*Luôn đặt giá trị của thuộc tính trong cặp dấu nháy*

* HTML5 không bắt buộc bạn phải dùng cặp dấu nháy **" "** để bao đóng giá trị của thuộc tính. Nhưng bạn nên làm như vậy nếu không muốn gặp một số điều phiền phức sau này, ví dụ như giá trị của thuộc tính có chứa dấu cách.
* Ví dụ hợp lệ (nhưng không khuyên dùng):

<a href= https://www.w3schools.com>

           Nhưng ví dụ sau đây sai cú pháp:

<p title= About W3Schools>

*Dùng dấu nháy đơn hay nháy kép?*

* Cả dấu nháy đơn **' '** và dấu nháy kép **" "** đều được chấp nhận khi bạn viết giá trị của thuộc tính.
* Trong một số trường hợp, như muốn có dấu nháy trong thuộc tính, bạn có thể sử dụng các cặp dấu nháy đơn và nháy kép bao nhau.
* Ví dụ:

<p title=' John "ShotGun" Nelson'>

           hoặc

<p title "John 'ShotGun' Nelson">

           đều được chấp nhận.

## Kết luận

Sau bài đọc, học viên đã được cung cấp những kiến thức sau:

* Cấu trúc của một thẻ HTML
* Một số thuộc tính nổi bật của một vài thẻ HTML cơ bản
* Lưu ý khi viết tên các thẻ HTML
* Phân biệt dấu nháy đơn và dấu nháy kép

## [Bài đọc] Các thẻ tiêu đề h1, h2, h3, h4, h5, h6

## Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Các thẻ tiêu đề trong HTML (h1-h6)
* Thẻ tạo đường ngang trong HTML (hr)
* Thực hành sử dụng các thẻ tiêu đề và thẻ hr

## Nội dung

Tiêu đề là những thành phần rất quan trọng trong các tài liệu HTML.

Chúng ta sử dụng các thẻ <h1>, <h2>, <h3>, <h4>, <h5>, <h6> để định nghĩa các tiêu đề.

Thẻ  <h1> định nghĩ tiêu đề quan trọng nhất (nổi bật nhất, lớn nhất), các thẻ khác sẽ lần lượt được sử dụng các tiêu đề ít quan trọng hơn. Thẻ <h6> định nghĩa tiêu đề nhỏ nhất.

Các công cụ tìm kiếm (chẳng hạn như Google, Bing...) sử dụng tiêu đề để đánh chỉ số và nội dung của trang web. Do đó, chúng ta nên sử dụng đúng tiêu đề để giúp cho người dùng dễ tìm thấy trang web của mình hơn.

Ngoài ra, người dùng cũng thường "lướt" qua nội dung trang web khá nhanh, do đó việc sử dụng tiêu đề sẽ giúp chúng ta nhanh chóng thể hiện được ghi được dấu ấn.

#### **Tạo đường kẻ ngang**

Nếu muốn phân tách các nội dung của trang web theo chiều ngang, chúng ta có thể sử dụng thẻ  <hr>. Thẻ này sẽ kẻ một đường nằm ngang tài liệu HTML.

## Kết luận

Sau bài đọc, học viên đã được cung cấp những kiến thức sau:

* Các thẻ tiêu đề trong HTML (h1-h6)
* Thẻ tạo đường ngang trong HTML (hr)
* Thực hành sử dụng các thẻ tiêu đề và thẻ hr

## [Bài đọc] Thẻ tạo đoạn văn bản: p

## Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Thẻ p và ý nghĩa
* Quy tắc hiển thị HTML trên trình duyệt
* Ý nghĩa của việc dùng thẻ đóng trong HTML
* Làm quen với thẻ xuống dòng (<br>)
* Hiển thị bài thơ trên trang web với thẻ <pre>

## Nội dung

Ta dùng thẻ **<p>**  để định nghĩa một **đoạn văn**.

##### **Ví dụ:**

**Chú ý:** Các trình duyệt tự động thêm vài khoảng trắng trước và sau một đoạn văn.

### Hiển thị HTML

Bạn không thể chắc chắn về cách mã HTML được hiển thị trên trình duyệt.

Với các kích thước màn hình khác nhau, mã HTML sẽ được hiển thị khác nhau.

Bạn không thể thay đổi cách hiển thị HTML bằng cách thêm các khoảng trắng hay xuống dòng trong mã HTML.

Trình duyệt sẽ bỏ tất cả các dòng và khoảng trắng thừa khi hiển thị trang web:

### Đừng quên thẻ đóng

Hầu hết các trình duyệt sẽ hiển thị chính xác mã HTML dù cho bạn quên thẻ đóng:

Ví dụ trên sẽ hiển thị tốt trên hầu hết trình duyệt, nhưng không nên phụ thuộc vào trình duyệt.

**Chú ý:** Không dùng thẻ đóng có thể tạo ra những kết quả không mong muốn hoặc gây ra lỗi.

### Thẻ xuống dòng

Thẻ **<br>** định nghĩa **xuống dòng**.

Dùng <br> nếu bạn muốn xuống dòng mà không tạo ra một đoạn văn mới.

 Thẻ <br> là thẻ rỗng, vì vậy nó không cần thẻ đóng.

### Hiển thị một bài thơ

Nếu dùng thẻ <p>, bài thơ sẽ chỉ hiển thị trên một dòng:

### Dùng thẻ <pre>

Để hiển thị một bài thơ ta có thể dùng thẻ **<pre>.**

Thẻ <pre> định nghĩa một đoạn văn bản đã được định dạng trước.

Đoạn văn bản bên trong thẻ <pre> </pre> sẽ được hiển thị với font chữ cố định (thường là font Courier) và được giữ lại các khoảng trắng và xuống dòng:

## Kết luận

Sau bài đọc, học viên đã được cung cấp những kiến thức sau:

* Thẻ p và ý nghĩa
* Quy tắc hiển thị HTML trên trình duyệt
* Ý nghĩa của việc dùng thẻ đóng trong HTML
* Làm quen với thẻ xuống dòng (<br>)
* Hiển thị bài thơ trên trang web với thẻ <pre>

[Bài đọc] Thẻ tạo liên kết: a

Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Dùng thẻ <a> để định nghĩa một liên kết
* Dùng thuộc tính href để định nghĩa địa chỉ của liên kết
* Dùng thuộc tính target để định nghĩa cách trình duyệt mở liên kết
* Dùng thẻ <img> bên trong thẻ <a> để tạo liên kết bằng ảnh
* Dùng thuộc tính id (id="*bookmark\_1*") để tạo bookmark cho trang web
* Dùng thuộc tính href  (href="#*bookmark\_1*") để liên kết đến bookmark có id =  *bookmark\_1*

Nội dung

Ta có thể thấy **liên kết (link)** ở hầu hết các trang web. Người dùng  nhấp chuột vào các liên kết để di chuyển giữa các trang web.

Liên kết - Siêu liên kết

Liên kết trong mã HTML còn được gọi là siêu liên kết (hyperlink).

Ta có thể  nhấp chuột (click) vào một liên kết để đi đến một trang web khác.

Khi ta di chuyển chuột lên trên một liên kết, con trỏ chuột sẽ biến thành hình bàn tay.

**Chú ý:** Ta thường dùng chữ để hiển thị liên kết.  Nhưng ta cũng có thể dùng ảnh hoặc các thành phần HTML khác để hiển thị liên kết.

Cú pháp của liên kết

Trong mã HTML, liên kết được định nghĩa bằng thẻ **<a>**:

<a href="*url*">*tên liên kết hiển thị ở đây*</a>

Thuộc tính **href**  để chỉ ra địa chỉ trang web(URL) mà liên kết sẽ dẫn đến (http://bob.codegym.vn/lessons/).  
Phần **chữ của liên kết** giữa 2 thẻ <a> </a> là phần người dùng nhìn thấy trên trang web(Codegym's lessons).

Nhấp chuột vào phần **chữ của liên kết** sẽ chuyển ta đến địa chỉ trang web trong thuộc tính **href.**

**Chú ý:** Nếu không có dấu gạch chéo (/) phía sau địa chỉ thư mục con của tên miền (http://bob.codegym.vn/lessons), ta có thể sẽ gửi 2 request đến server. Tuy nhiên nhiều server sẽ tự động thêm dấu gạch chéo vào địa chỉ trang web, sau đó tạo một request mới để xử lý tiếp. Bởi vì http://bob.codegym.vn/lessons và http://bob.codegym.vn/lessons/ là 2 URL khác nhau.

Liên kết cục bộ

Ví dụ trên dùng một địa chỉ URL tuyệt đối (địa chỉ đầy đủ của trang web).

Ta có thể dùng một liên kết cục bộ khi trỏ đến địa chỉ trên trang web của chính mình.

Liên kết cục bộ có thuộc tính **href**là một địa chỉ URL tương đối (không có http://www...).

Màu sắc của liên kết

Mặc định, liên kết sẽ có màu như sau (trên tất cả trình duyệt):

* Liên kết chưa bị click vào sẽ có màu xanh và được gạch chân.
* Liên kết đã bị click sẽ có màu tím và được gạch chân.
* Liên kết đang bị click lên sẽ có màu đỏ và được gạch chân.

Ta có thể thay đổi màu mặc định bằng cách dùng các thuộc tính CSS sau:

Thuộc tính target của một liên kết

Thuộc tính **target** cho trình duyệt biết cách mở ra trang web trong thẻ **href**.

Thuộc tính **target**có thể là một trong các giá trị sau:

* \_blank - Mở liên kết trong một cửa sổ hoặc một tab mới của trình duyệt
* \_self - Mở liên kết trong tab/cửa sổ hiện tại của trình duyệt (Không cần khai báo cũng được vì đây là mặc định của thẻ <a>)
* \_parent - Mở liên kết trong parent frame
* \_top - Mở liên kết trong cửa sổ chính
* framename - Mở liên kết trong một frame có tên là framename (Tìm hiểu thêm về thẻ <frame>)

Ví dụ dưới đây sẽ mở liên kết trong một tab/cửa sổ mới của trình duyệt:

**Tip:** Ta có thể dùng target="\_top" để mở liên kết trong tab mới khi đường dẫn đặt trong một frame (xem lại ví dụ

Codegym's lessons đầu tiên phía trên):

Dùng ảnh làm liên kết

Ảnh thường được dùng làm liên kết:

**Chú ý:** Thuộc tính border:0; để IE9 (và các phiên bản trước đó) không hiển thị một đường viền xung quanh ảnh khi dùng ảnh làm liên kết.

Dùng liên kết để tạo một đánh dấu trang(bookmark)

Bookmark được dùng để người đọc nhảy đến phần khác trên cùng một trang web.

Bookmarks rất hữu ích khi trang web quá dài.

Để tạo một bookmark, ta thêm liên kết vào danh sách đánh dấu trang.

Khi liên kết được click, trang web sẽ cuộn xuống phần đã được đánh dấu.

Ví dụ

Đầu tiên, tạo một bookmark bằng thuộc tính **id** của thẻ:  
<h2 id="C4">Chapter 4</h2>

Sau đó, thêm liên kết cục bộ cho bookmark ở cùng một trang web:

<a href="#C4">Jump to Chapter 4</a>

Hoặc thêm liên kết sang một trang khác:  
<a href="http://bob.codegym.vn/clean-code#C6">Go to Chapter 6</a>

Đường dẫn bên ngoài

Có thể liên kết đến các trang bên ngoài bằng một địa chỉ URL đầy đủ hoặc đường dẫn tương đối so với trang web hiện tại.

Ví dụ dưới dùng một đường dẫn URL đầy đủ để liên kết đến trang web:

Ví dụ dưới liên kết đến trang web trong thư mục user của website hiện tại:  
  
   
Ví dụ dưới liên kết đến trang web trong cùng thư mục với trang web hiện tại:

Kết luận

Sau bài đọc, học viên đã được cung cấp những kiến thức sau:

* Dùng thẻ **<a>** để định nghĩa một liên kết
* Dùng thuộc tính **href** để định nghĩa địa chỉ của liên kết
* Dùng thuộc tính **target** để định nghĩa cách trình duyệt mở liên kết
* Dùng thẻ **<img>** bên trong thẻ <a> để tạo liên kết bằng ảnh
* Dùng thuộc tính **id** (id="*bookmark\_1*") để tạo bookmark cho trang web
* Dùng thuộc tính **href** (href="#*bookmark\_1*") để liên kết đến bookmark có id =  *bookmark\_1*

## [Bài đọc] Thẻ hiển thị hình ảnh: img

## Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Hướng dẫn hiển thị hình ảnh trên trình duyệt bằng thẻ HTML
* Hướng dẫn sử dụng thuộc tính src của thẻ <img>
* Ý nghĩa của thuộc tính alt
* Ý nghĩa của thuộc tính width và height
* Ý nghĩa của thuộc tính CSS float
* Ý nghiax của thẻ <map>
* Ý nghĩa của thẻ <area>
* Ý nghĩa của thuộc tính usemap của thẻ <img>

## Nội dung

Có 3 định dạng ảnh thường dùng trên các trang web: JPG, PNG, GIF

### Ví dụ

### Cú pháp

Trong mã HTML, hình ảnh được định nghĩa bằng thẻ **<img>.**

Thẻ ***<img>*** là thẻ rỗng, chỉ gồm các thuộc tính và không có thẻ đóng.

Thuộc tính **src** để chỉ ra URL (địa chỉ web) của hình ảnh:

<img src="*url*" alt="*some\_text*" style="width:width;height:height;">

### Thuộc tính *alt*

Thuộc tính **alt**tạo ra một dòng chữ thay thế cho hình ảnh trong trường hợp ảnh không hiển thị được (do mạng chậm, đường dẫn của thuộc tính **src**bị sai, hoặc khi người duyệt web dùng một phần mềm đọc màn hình - screen reader).

Nếu trình duyệt không tìm thấy hình ảnh, nó sẽ hiển thị giá trị của thuộc tính **alt:**

 Ta luôn luôn phải khai báo thuộc tính **alt.**Một trang web không được coi là hợp lệ nếu không có đủ các thuộc tính **alt**của thẻ **<img>**.

### Đọc thêm:

### Phần mềm đọc màn hình(screen readers)

Một phần mềm đọc màn hình sẽ đọc mã HTML, lọc ra các văn bản và đọc cho người dùng nghe nội dung trang web. Phần mềm đọc màn hình giúp hỗ trợ cho người mù hoặc thị lực yếu hay người khó học.

### Cỡ ảnh - Chiều rộng và chiều cao

Ta dùng thuộc tính **style** để chỉ định chiều rộng và chiều cao của ảnh.

Giá trị tính theo pixel (thêm px sau số chỉ chiều rộng/cao):

 Ngoài ra, ta có thể dùng thuộc tính  **width** và **height**. Với hai thuộc tính này, giá trị mặc định là tính theo pixel:  **Chú ý:** Luôn định rõ chiều cao và chiều rộng của hình ảnh. Nếu không trang web sẽ bị rung rinh khi hiển thị hình ảnh.

### Nên dùng thuộc tính width và height, hay style?

Cả 3 thuộc tính **width**, **height**, and **style**đều hợp lệ trong HTML5.

Tuy nhiên, ta nên dùng thuộc tính **style** để  tránh việc kích thước ảnh bị các thuộc tính css khác điều khiển:

### Đường dẫn ảnh

Trong dự án, ảnh thường được lưu vào một thư mục riêng. ta thường dùng đường dẫn tương đối cho thuộc tính **src**.

Trong đường dẫn ảnh phải ghi đầy đủ tên thư mục vào đường dẫn ảnh:

#### **Ví dụ:**

<img src="/images/html5.gif" alt="HTML5 Icon" style="width:128px;height:128px;">

### Ảnh động

Ta có thể hiển thị ảnh động (định dạng GIF) như ảnh thông thường, không cần thay đổi cú pháp.

### Dùng ảnh làm liên kết

 Để dùng ảnh làm liên kết, chỉ cần đặt thẻ <img> vào trong thẻ <a>:

**Chú ý:** thêm *border:0;* vào **style** để các phiên bản IE9 và trước đó không hiển thị đường viền quanh hình ảnh dùng làm liên kết.

### Đặt vị trí cho ảnh

Dùng thuộc tính CSS **float** để đặt ảnh bên trái hoặc bên phải văn bản:

### Bản đồ ảnh

Ta có thể dùng thẻ <map> để định nghĩa một bản-đồ-ảnh(image-map). Một bản đồ ảnh là một hình ảnh có các vùng có thể nhấp chuột vào.

Thuộc tính **usemap**của thẻ *<img>*  tương ứng với thuộc tính **name**của thẻ <map> .

Thẻ <map> có các thẻ <area> để định nghĩa các vùng có thể nhấp chuột trong bản-đồ-ảnh:

## Kết luận

Sau bài đọc, học viên đã được cung cấp kiến thức về:

* Dùng thẻ **<img>** để định nghĩa một hình ảnh
* Dùng thuộc tính **src** để định nghĩa URL của hình ảnh
* Dùng thuộc tính **alt** để định nghĩa dòng chữ thay thế cho ảnh khi ảnh không hiển thị được
* Dùng thuộc tính **width** và **height** để định nghĩa kích thước của ảnh
* Dùng thuộc tính CSS **width** and **height** để chắc chắn định nghĩa kích thước ảnh (không bị các thuộc tính CSS khác ghi đè)
* Dùng thuộc tính CSS **float** để xếp vị trí ảnh
* Dùng thẻ **<map>** để định nghĩa một bản-đồ-ảnh(image-map)
* Dùng thẻ **<area>** để định nghĩa các vùng có thể click bản-đồ-ảnh
* Dùng thuộc tính **usemap**của thẻ <img> để trỏ ảnh đến thẻ <map> tương ứng

**Chú ý:** Việc load ảnh nhanh hay chậm phụ thuộc độ lớn bức ảnh. Ảnh lớn quá sẽ làm chậm tốc độ load trang web. Phải lưu ý khi dùng ảnh cho trang web.

## [Bài đọc] Sử dụng các thẻ tạo danh sách: ul, ol, li

## Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Dùng thẻ <ul> để định nghĩa một danh sách không xếp thứ tự
* Dùng thuộc tính CSS list-style-type để định nghĩa biểu tượng cho các phần tử trong danh sách
* Dùng thẻ <ol> để định nghĩa một danh sách xếp thứ tự
* Dùng thuộc tính type để định nghĩa kiểu đánh thứ tự
* Dùng thẻ <li> để định nghĩa một phần tử  trong danh sách
* Dùng thẻ <dl> để định nghĩa một danh sách miêu tả
* Dùng thẻ <dt> để định nghĩa một thuật ngữ
* Dùng thẻ <dd> để miêu tả thuật ngữ ngay phía trên nó trong danh sách miêu tả
* Có thể lồng các danh sách bên trong các danh sách
* Các phần tử trong danh sách có thể chứa các thành phần HTML khác
* Dùng thuộc tính CSS float:left hoặc display:inline để hiển thị danh sách nằm ngang

## Nội dung

Ví dụ về danh sách trong HTML

| **Một danh sách không xếp thứ tự:**  * **Item** * **Item** * **Item** * **Item** | **Một danh sách có xếp thứ tự:**  1. **First item** 2. **Second item** 3. **Third item** 4. **Fourth item** |
| --- | --- |

### Danh sách không xếp thứ tự

Ta định nghĩa danh sách không xếp thứ tự bằng thẻ **<ul>**, viết tắt của **u**nordered **l**ist. Mỗi phần tử của danh sách được định nghĩa bằng thẻ  **<li>** , viết tắt của **l**ist**i**tem.

Mặc định, các phần tử của danh sách sẽ được hiển thị kèm theo một dấu tròn phía trước:

#### **Các biểu tượng cho phần tử trong danh sách không xếp thứ tự**

Ta dùng thuộc tính CSS **list-style-type** để định kiểu cho biểu tượng của phần tử trong danh sách:

|  |  |
| --- | --- |
| **Giá trị** | **Miêu tả** |
| disc | (Mặc định) Các biểu tượng là hình tròn màu đen |
| circle | Các biểu tượng là hình tròn rỗng |
| square | Các biểu tượng là hình vuông |
| none | Không hiển thị các biểu tượng |

Ví dụ dùng giá trị **disc**  
  
 Ví dụ dùng giá trị **circle**   
Ví dụ dùng giá trị **square**  
  
Ví dụ dùng giá trị **none**

### Danh sách xếp theo thứ tự

Ta định nghĩa một danh sách xếp theo thứ tự bằng thẻ **<ol>** (**o**rdered **l**ist). Mỗi phần tử của danh sách được định nghĩa bằng thẻ **<li>** (**l**ist **i**tem).

Mặc định thì các phần tử sẽ được hiển thị kèm số thứ tự:

### Thuộc tính type của danh sách xếp thứ tự

Thuộc tính **type** của thẻ <ol>  dùng để định kiểu cho biểu tượng các phần tử:

|  |  |
| --- | --- |
| **Type** | **Miêu tả** |
| type="1" | Các phần tử sẽ được đánh thứ tự bằng số (Mặc định) |
| type="A" | Các phần tử sẽ được đánh thứ tự bằng chữ cái hoa |
| type="a" | Các phần tử sẽ được đánh thứ tự bằng chữ cái thường |
| type="I" | Các phần tử sẽ được đánh thứ tự bằng chữ số La Mã hoa |
| type="i" | Các phần tử sẽ được đánh thứ tự bằng chữ số La Mã thường |

Đánh thứ tự bằng số: type="1"

Đánh thứ tự bằng chữ cái hoa: type="A"Đánh thứ tự bằng chữ cái thường: type="a"   
  
Đánh thứ tự bằng chữ số La Mã hoa: type="I"Đánh thứ tự bằng chữ số La Mã thường: type="i"

### Danh sách miêu tả

Mã HTML cũng có thẻ danh sách miêu tả.

Một danh sách miêu tả là danh sách các thuật ngữ cùng với miêu tả của chúng.

Thẻ **<dl>**định nghĩa một danh sách miêu tả, trong đó, thẻ **<dt>** định nghĩa thuật ngữ (tên gọi), và thẻ  **<dd>** miêu tả các thuật ngữ:

### Danh sách bên trong danh sách

Ta có thể lồng danh sách vào một danh sách khác, tạo thành danh sách bên trong danh sách, tiếng Anh gọi là nested list:

**Chú ý:** Mỗi phần tử không chỉ có thể chứa danh sách khác mà còn có thể chứa các thành phần HTML khác như ảnh hay liên kết,...

### Danh sách nằm ngang

Danh sách trong HTML có thể được định kiểu theo nhiều cách bằng CSS.

Một cách làm phổ biến là định kiểu chuyển danh sách thành nằm ngang để tạo thành một menu:

## Kết luận

Sau bài đọc, học viên đã được cung cấp những kiến thức về:

* Dùng thẻ **<ul>** để định nghĩa một danh sách không xếp thứ tự
* Dùng thuộc tính CSS**list-style-type** để định nghĩa biểu tượng cho các phần tử trong danh sách
* Dùng thẻ **<ol>** để định nghĩa một danh sách xếp thứ tự
* Dùng thuộc tính **type** để định nghĩa kiểu đánh thứ tự
* Dùng thẻ **<li>** để định nghĩa một phần tử  trong danh sách
* Dùng thẻ **<dl>** để định nghĩa một danh sách miêu tả
* Dùng thẻ **<dt>** để định nghĩa một thuật ngữ
* Dùng thẻ **<dd>** để miêu tả thuật ngữ ngay phía trên nó trong danh sách miêu tả
* Có thể lồng các danh sách bên trong các danh sách
* Các phần tử trong danh sách có thể chứa các thành phần HTML khác
* Dùng thuộc tính CSS **float:left** hoặc **display:inline**để hiển thị danh sách nằm ngang

## [Bài đọc] Chú thích trong HTML

## Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Ý nghĩa của chú thích trong HTML
* Chú thích dựa theo điều kiện và ý nghĩa của nó

## Nội dung

#### **Chú thích**

Trong quá trình soạn thảo mã HTML, đôi lúc bạn sẽ cần:

* Ghi một vài câu chú thích nào đó (những câu chú thích này không hiển thị ra màn hình)
* Hoặc vô hiệu hóa một đoạn mã HTML để kiểm tra sự thực thi của các đoạn mã khác.

Khi đó, chúng ta sẽ đặt những nội dung mà mình muốn chú thích vào bên trong cặp dấu

<!--  nội dung chú thích được viết ở đây -->

Ví dụ:

*<!-- Câu này là chú thích nên KHÔNG được hiển thị -->*

<p>Đoạn này không phải chú thích nên được hiển thị</p>

*<!--<p>Đoạn này là chú thích nên KHÔNG được hiển thị</p>-->*

Các câu chú thích có thể được viết trên nhiều dòng

*<!--<p>Câu này là chú thích nên KHÔNG được hiển thị</p>*

*<p>Câu này là chú thích nên KHÔNG được hiển thị</p>*

*<p>Câu này là chú thích nên KHÔNG được hiển thị</p>-->*

<p>Câu này không phải chú thích nên được hiển thị</p>

**Lưu ý: Phải có dấu chấm than (!) ở thẻ mở.**

#### **Chú thích dựa theo điều kiện**

Chú thích dựa theo điều kiện chỉ làm việc trên trình duyệt Internet Explorer (IE), còn đối với các trình duyệt khác thì chúng sẽ bị bỏ qua.

Chú thích dựa theo điều kiện được hỗ trợ từ phiên bản IE5 trở đi. Bạn có thể sử dụng chúng để đưa ra các hướng dẫn điều kiện cho các phiên bản khác nhau của IE.

Nếu bạn xem ví dụ này bằng trình duyệt IE8 thì trên màn hình sẽ hiển thị dòng chữ "BẠN ĐANG SỬ DỤNG TRÌNH DUYỆT Internet Explorer 8". Còn nếu xem bằng các trình duyệt khác thì sẽ không thấy dòng chữ đó.

Explain

<!DOCTYPE html>

<html>

<body>

*<!--[if IE 8]>*

*<p>BẠN ĐANG SỬ DỤNG TRÌNH DUYỆT Internet Explorer 8</p>*

*<![endif]-->*

</body>

</html>

Nếu bạn xem ví dụ này bằng trình duyệt IE phiên bản cũ hơn 9 thì trên màn hình sẽ hiển thị dòng chữ "BẠN ĐANG SỬ DỤNG TRÌNH DUYỆT Internet Explorer phiên bản cũ hơn 9". Còn nếu xem bằng các trình duyệt khác thì sẽ không thấy dòng chữ đó.

<!DOCTYPE html>

<html>

<body>

*<!--[if lt IE 9]>*

*<p>BẠN ĐANG SỬ DỤNG TRÌNH DUYỆT Internet Explorer phiên bản cũ hơn 9</p>*

*<![endif]-->*

</body>

</html>

Trình duyệt IE rất hạn chế trong việc hỗ trợ định dạng trang web (bạn thử xem trang webcoban.vn bằng trình duyệt Chrome và IE sẽ dễ dàng nhận thấy sự yếu kém của IE). Dó đó, chúng ta cần phải sử dụng chú thích điều kiện để nhúng các tập tin định dạng dành cho trình duyệt IE. Nếu người dùng đang xem trang web bằng trình duyệt IE thì những tập tin định dạng này sẽ được thực thi giúp trang web hiển thị trên IE được tốt hơn.

## Kết luận

Sau bài đọc, học viên đã được cung cấp những kiến thức sau:

* Ý nghĩa của chú thích trong HTML
* Chú thích dựa theo điều kiện và ý nghĩa của nó

[Bài đọc] Sử dụng HTML entity

Mục tiêu

Bài đọc cung cấp cho học viên những kiến thức sau:

* Cách hiển thị các kí tự đặc biệt trong HTML
* Giới thiệu các entity ứng với các kí tự đặc biệt trong HTML

Nội dung

Trong HTML, khi chúng ta muốn hiển thị các ký tự đặc biệt (được dành riêng cho mã HTML, chẳng hạn như ">", "<", "&"...) thì chúng ta cần sử dụng entity.

Một entity được bắt đầu bằng dấu "&" và kết thúc bằng dấu ";". Có thể sử dụng entity theo tên hoặc theo mã số.

Chẳng hạn, để hiển thị ký tự "<" thì chúng ta phải viết "&lt;" hoặc "&#60;". Trong đó, "lt" là tên của ký tự "nhỏ hơn", còn "#60" là mã số của nó.

Sau đây là danh sách các entity thường được sử dụng:

|  |  |  |  |
| --- | --- | --- | --- |
| **Ký tự muốn hiển thị** | **Mô tả** | **Tên của entity** | **Mã số của entity** |
|  | non-breaking space | &nbsp; | &#160; |
| < | less than | &lt; | &#60; |
| > | greater than | &gt; | &#62; |
| & | ampersand | &amp; | &#38; |
| " | double quotation mark | &quot; | &#34; |
| ' | single quotation mark (apostrophe) | &apos; | &#39; |
| ¢ | cent | &cent; | &#162; |
| £ | pound | &pound; | &#163; |
| ¥ | yen | &yen; | &#165; |
| € | euro | &euro; | &#8364; |
| © | copyright | &copy; | &#169; |
| ® | registered trademark | &reg; | &#174; |

Kết luận

Sau bài đọc, học viên đã được cung cấp các kiến thức sau:

* Cách hiển thị các kí tự đặc biệt trong HTML
* Giới thiệu các entity ứng với các kí tự đặc biệt trong HTML

## [\*Đọc thêm] Các hệ thống quản lý mã nguồn và phiên bản

#### Tầm quan trọng của các hệ thống quản lý mã nguồn

Quản lý phiên bản (mã nguồn) là một hệ thống lưu trữ các thay đổi của một tập tin hoặc tập hợp các tập tin theo thời gian, do đó nó giúp bạn có thể quay lại một phiên bản xác định nào đó sau này.

Nếu bạn là một nhà thiết kế đồ hoạ hoặc thiết kế website, bạn muốn lưu trữ tất cả các phiên bản của một bức ảnh hoặc bố cục, thì sử dụng Hệ Thống Quản Lý phiên bản (Version Control System - VCS) là một cách làm rất khôn ngoan. Một VCS cho phép bạn: khôi phục lại phiên bản cũ của các file, khôi phục lại phiên bản cũ của toàn bộ dự án, xem lại các thay đổi đã được thực hiện theo thời gian, xem ai là người thực hiện thay đổi cuối cùng có thể gây ra sự cố, hay xem ai là người đã gây ra sự cố đó và còn nhiều hơn thế nữa. Sử dụng VCS còn đồng nghĩa với việc khi bạn làm rối tung mọi thứ lên hay vô tình xoá mất các file đi, bạn có khôi phục lại chúng một cách dễ dàng. Hơn nữa, tất cả quá trình này có thể được thực hiện rất nhanh chóng và không hề tốn quá nhiều công sức.

#### Phân loại hệ thống quản lý mã nguồn

#### **Hệ thống quản lý mã nguồn cục bộ**

Nhiều người chọn phương pháp quản lý phiên bản bằng cách copy các file sang một thư mục khác (có thể là các thư mục được đặt tên theo thời gian). Đây là một phương pháp rất phổ biến bởi vì nó rất đơn giản, tuy nhiên nó cũng rất dễ gây ra lỗi. Bạn sẽ rất dễ quên rằng bạn đang ở trong thư mục nào hay vô tình sửa hoặc sao chép nhầm file mà bạn không muốn.

#### **Hệ thống quản lý mã nguồn tập trung**

Vấn đề nghiêm trọng tiếp theo mà mọi người thường mắc phải là họ cần cộng tác với các lập trình viên khác trong hệ thống. Để vượt qua trở ngại này, Hệ Thống Quản Lý Phiên Bản Tập Trung (Centralized Version Control Systems - CVCSs) được phát triển. Các hệ thống này, ví dụ như CVS, Subversion, và Perforce, bao gồm một máy chủ có chứa tất cả các tập tin đã được "phiên bản hoá", và danh sách các máy khách có quyền thay đổi các tập tin này trên máy chủ trung tâm đó.

Mô hình này cung cấp rất nhiều lợi thế, đặc biết so với việc quản lý cục bộ. Ví dụ, tất cả người dùng đều biết một phần nào đó những việc mà những người khác trong dự án đang làm. Người quản lý có quyền quản lý ai có thể làm gì theo ý muốn; và việc này dễ dàng hơn nhiều so với việc phải quản lý ở từng cơ sở dử liệu ở từng máy riêng biệt.

Tuy nhiên, mô hình này cũng có những bất cập nghiêm trọng. Dễ nhận thấy nhất đó là "sự cố tập trung" mà máy chủ trung tâm mắc phải. Nếu máy chủ đó không hoạt động trong một giờ, nghĩa là trong khoảng thời gian đó không ai có thể cộng tác với những người còn lại hoặc lưu trữ các thay đổi đã được phiên bản hoá của bất kỳ tập tin nào mà người đó đang thao tác. Nếu ổ cứng lưu trữ cơ sở dữ liệu trung tâm bị hỏng, và các sao lưu dự phòng chưa được tạo ra tính đến thời điểm đó, bạn sẽ mất toàn bộ lịch sử của dự án đó, ngoại trừ những phiên bản cục bộ mà người dùng có được trên máy tính cá nhân. Các hệ thống quản lý phiên bản cục bộ phải đối diện với vấn đề tương tự như thế này mỗi khi toàn bộ lịch sử của dự án được lưu ở một nơi, bạn có nguy cơ mất tất cả.

#### **Hệ thống quản lý mã nguồn phân tán**

Đã tới lúc cần tới các Hệ Thống Quản Lý Phiên Bản Phân Tán - Distributed Version Control Systems (DVCSs). Trong các DVCS (ví dụ như Git, Mercurial, Bazaar hay Darcs), các máy khách không chỉ check out - sao chép về máy cục bộ, phiên bản mới nhất của các tập tin: chúng sao chép (mirror) toàn bộ kho chứa (repository). Chính vì vậy nếu như một máy chủ nào mà các hệ thống quản lý phiên bản này (mỗi máy khách là một hệ thống riêng biệt) đang cộng tác ngừng hoạt động, thì kho chứa từ bất kỳ máy khách nào cũng có thể dùng để sao chép ngược trở lại máy chủ để khôi phục lại toàn bộ hệ thống.

Ngoài ra, phần lớn các hệ thống này xử lý rất tốt việc quản lý nhiều kho chứa từ xa, vì thế bạn có thể cộng tác với nhiều nhóm người khác nhau theo những cách khác nhau trong cùng một dự án. Việc này cho phép bạn cài đặt nhiều loại "tiến trình công việc" (workflow) không thể thực hiện được với các hệ thống tập trung, ví dụ như các mô hình phân cấp.

#### Một số hệ thống quản lý mã nguồn thông dụng

**CVS** là chữ viết tắt của Concurrent Versions System từ tiếng Anh. CVS có nghĩa là "hệ thống các phiên bản đồng thời". CVS còn có tên gọi khác là Concurrent Versioning System. Đây là hệ thống quản lý các phiên bản phần mềm mã nguồn mở được đề xuất và phát triển bởi [Dick Grune](https://vi.wikipedia.org/w/index.php?title=Dick_Grune&action=edit&redlink=1) vào thập niên 1980. CVS được dùng để theo vết tất cả các công việc và các thay đổi trong một tập hợp các tập tin. CVS cho phép nhiều lập trình viên và các thành viên khác trong một tổ phát triển phần mềm cộng tác với nhau.

**SVN - Subversion**là phần mềm được dùng để quản lý tài nguyên của một dự án, hiệu quả trong làm việc theo nhóm. Có thể hiểu SVN giống như là một hệ thống server – client cho phép bạn download và upload các thành phần của một dự án. **SVN có một số ưu điểm như miễn phí, q**uản lý mã nguồn dạng tập trung , quản lý mã nguồn dự án rất tốt nếu như bạn đang xây dựng ứng dụng có nhiều thành viên tham gia. Mọi thay đổi đều được SVN ghi nhận lại chẳng hạn như tập tin, thời gian, thành viên… để dùng lại trong những trường hợp cần thiết. SVN còn cho phép bạn có thể khôi phục lại các phiên bản trước đó.

**Git** - Hệ thống quản lý source phân tán, khác so với các hệ thống source control khác như SVN hay CVS. Quản lý phân tán của Git là một repositories không cần có chung một nơi để lưu trữ, mà mỗi thành viên sẽ có một repository ở local của họ. Tất cả thao tác ta làm việc với Git đều ở trên máy của ta, local repository, khi quyết định đưa những thay đổi đó lên server ta chỉ cần một thao tác "push" nó lên server. Chúng ta vẫn có thể share thay đổi của chúng ta cho thành viên khác, bằng cách commit hoặc update trực tiếp từ máy của họ mà không phải thông qua repositories gốc trên server (thông qua share ssh cho nhau). Và dĩ nhiên là mọi thao tác đều mang theo thông tin history với Git.

[Thực hành] Tạo dự án mới trên GitHub

Mục tiêu

Luyện tập thao tác tạo dự án mới trên GitHub.

Mô tả

Trong phần này, chúng ta sẽ luyện tập thao tác quan trọng khi làm việc với GitHub đó là tạo một dự án mới.

Sau khi thực hiện xong các bước hướng dẫn thì hãy nộp bài tập bằng cách:

* Mô tả lại ý nghĩa của các câu lệnh đã sử dụng
* Dán link của Repository vừa tạo lên

Hướng dẫn

Trong phần hướng dẫn này, chúng ta sẽ tạo một dự án mới trên GitHub và bắt đầu đưa mã nguồn lên đó. Các bước thực hiện như sau:

1. Đi đến trang: [*https://github.com*](https://github.com/)

2. Nhấn vào mục "New repository" để bắt đầu tạo một Repository mới:

3. Trong màn hình tạo Repository mới, chúng ta điền tên của dự án mới vào. Chẳng hạn chúng ta đặt tên dự án mới là "*my-new-project*". Sau đó nhấn và nút "Create repository".

Kết quả sẽ được một Repository mới. Giao diện ban đầu của Repository này chứa các hướng dẫn cần thiết để làm việc với nó:

4. Sau khi đã tạo được Remote Repository trên GitHub, bây giờ chúng ta sẽ mở cửa sổ Terminal lên để tạo Local Repository và đồng bộ nó tới Remote Repository trên.

Chúng ta sẽ tạo một thư mục mới trên máy tính của mình, có thể đặt tên bất kỳ. Chẳng hạn chúng ta có thể đặt tên thưc mục là "*my-new-project*" (giống với tên của Repository).

5. Chuyển Terminal đến thư mục "*my-new-project*" vừa tạo và sử dụng lệnh *git init* để khởi tạo Repository mới.

6. Tiếp theo, sử dụng lệnh *git remote add origin* để kết nối tới Remote Repository (chú ý sử dụng dúng đường dẫn của Repository mà bạn vừa tạo):

git remote add origin https://github.com/codegym-vn/my-new-project.git

Như vậy là Local Repository và Remote Repository đã được liên kết với nhau. Bây giờ chúng ta có thể bắt đầu làm việc trên dự án của mình.

7. Trong thư mục "*my-new-project*", chúng ta tạo một file mới có tên là "README.md". Đây là file dùng để chứa mô tả về dự án này, hầu hết tất cả các Repository đều có file này để giới thiệu cho người khác biết về mục đích của dự án. Chúng ta có thể viết một nội dung bất kỳ vào file này.

8. Sử dụng câu lệnh *git add* để đưa file README.md vào trong chỉ mục của Git.

git add README.md

9. Sử dụng câu lệnh *git commit* để lưu lại các thay đổi.

git commit -m "Add README.md file"

Trong câu lệnh trên, tuỳ chọn *-m* được sử dụng để ghi một "*message*" đánh dấu cho thay đổi vừa rồi. Nó giải thích rằng chúng ta vừa thêm một file README.md mới. Chúng ta cần ghi các thông điệp có ý nghĩa để dễ dàng cho việc hiểu và cộng tác nhóm sau này.

10. Sử dụng câu lệnh *git push* để đẩy các thay đổi ở Local Repository lên trên Remote Repository.

git push origin master

Như vậy là chúng ta đã hoàn thành một luồng làm việc cơ bản với GitHub. Local Repository và Remote Repository đã được đồng bộ với nhau.

Thử quay lại trang dự án ở trên GitHub thì chúng ta sẽ thấy một giao diện như sau:

11. Hãy lặp lại các bước từ 7 đến 10 để tạo các file mới.

## [Thực hành] Clone dự án từ Git

### Mục tiêu

Luyện tập thao tác Clone dự án có sẵn từ GitHub.

### Mô tả

Trong phần này, chúng ta sẽ luyện tập thao tác cơ bản nhất khi làm việc với Git đó là sao chép một Repository có sẵn về máy của mình.

Sau khi thực hiện xong các bước hướng dẫn thì hãy hoàn thành bài thực hành này bằng cách mô tả lại cú pháp và ý nghĩa của câu lệnh clone.

### Hướng dẫn

Trong phần hướng dẫn này, chúng ta sẽ sao chép (clone) một Repository có sẵn từ GitHub về máy của mình. Các bước thực hiện như sau:

1. Đi đến trang [*https://github.com/codegym-vn/wf-hello*](https://github.com/codegym-vn/wf-hello)

2. Nhấn vào nút "Clone or download" rồi copy đường dẫn của Repository

3. Mở màn hình Terminal trên máy tính của mình, di chuyển đến thư mục mình muốn chứa các Repository (sử dụng lệnh *cd*).

4. Dùng lệnh *git clone* để sao chép Repository về máy của mình:

git clone https:*//github.com/codegym-vn/wf-hello.git*

Kết quả:

5. Mở thư mục vừa sao chép về, chúng ta sẽ thấy các file nằm trong đó:

Như vậy, trong phần này chúng ta đã học cách để sử dụng câu lệnh *git clone* để sao chép một Repository có sẵn.

## [Thực hành] Sử dụng thẻ HTML cơ bản

## Mục đích

Luyện tập các thẻ cơ bản <h1>, <p>, <a>, <img>

## Mô tả

Tạo trang web hiển thị nội dung như sau. Dữ liệu trên trang được định dạng ứng với các thẻ.

Để hoàn thành bài thực hành, học viên cần:

* Đưa mã nguồn lên GitHub
* Dán link của repository lên phần nộp bài trên CodeGymX

## Hướng dẫn

Bước 1: Tạo cấu trúc tài liệu HTML (xem trong tab HTML)

Bước 2: Đưa từng nội dung vào giữa cặp thẻ <body></body> (xem trong tab HTML)

Bước 3: Đưa tiếp nội dung vào (xem trong tab HTML)

Bước 4: Chèn ảnh tạo liên kết  (xem trong tab HTML)

Bước 5: Chạy chương trình. Xem kết quả hiển thị.

#### **Tổng kết**

Vậy trong bài thực hành này chúng ta đã luyện tập với các thẻ cơ bản HTML như <a>, <p>, <h1>, <img>

[Bài tập] Sử dụng Git

Mục tiêu

Luyện tập sử dụng các thao tác cơ bản với *git.*

Mô tả

Trong phần này, chúng ta sẽ thực hiện các bước thao tác cơ bản nhất để bắt đầu làm việc với Git và GitHub. Bắt đầu từ việc tạo dự án trên GitHub cho đến việc đưa các file mã nguồn lên GitHub.

Để hoàn thành bài tập, cần nộp link của dự án lên CodeGymX.

Hướng dẫn

**Bước 1:** Tạo repository trên Github (https://github.com) đặt tên GitProject

**Bước 2:** Tạo thư mục chứa dự án trên máy tính đặt tên thư mục là GitProject

**Bước 3:** Clone dự án từ github về thư mục trên máy

**Bước 4:** Tạo file README.md đưa một số nội dung vào

**Bước 5:** Đẩy file README.md từ thư mục trên máy tính lên Github

**Bước 6:** Tạo thư mục source, chứa file index.html, news.html. Đưa nội dung bất kỳ vào 2 file.

**Bước 7:** Đẩy thư mục source chứa 2 file vừa tạo lên Github

## [Bài tập] Tạo danh sách

## Mục tiêu

Luyện tập sử dụng các thẻ danh sách

## Mô tả

Hãy sử dụng các thẻ <ul>, <ol>, <li> để tạo một danh sách như sau:

[Bài tập] Tạo trang CV cá nhân

Mục tiêu

Luyện tập việc tạo trang web và sử dụng các thẻ HTML cơ bản.

Mô tả

Tạo một trang CV cá nhân bao gồm các thông tin (Lưu ý: Không dùng các thông tin thật của mình):

* Họ và tên
* Địa chỉ
* Số điện thoại
* Email
* Ảnh
* Học vấn
* Bằng cấp
* Kỹ năng
* Kinh nghiệm
* Định hướng phát triển cá nhân

Để hoàn thành bài thực hành, học viên cần:

* Đưa mã nguồn lên GitHub
* Dán link của repository lên phần nộp bài trên CodeGymX

Hướng dẫn

* Sử dụng các thẻ h1, h2, h3, h4... để tạo các tiêu đề.
* Sử dụng thẻ <p> để tạo các đoạn văn bản.
* Sử dụng thẻ <img> để hiển thị ảnh.
* Sử dụng thẻ <a> để tạo các liên kết.

Học viên tham khảo CV demo tại file đính kèm.

|  |  |
| --- | --- |
|  | [Screen Shot 2022-03-30 at 10.43.20.png](https://james.codegym.vn/pluginfile.php/64842/mod_assign/introattachment/0/Screen%20Shot%202022-03-30%20at%2010.43.20.png?forcedownload=1) |