**BÀI THU HOẠCH II**

TÌM HIỂU VỀ HTML, CSS, JAVA SCRIPT

Thực tập sinh: Lê Vương Khánh

Người hướng dẫn: Nguyễn Trần Xuân Lộc, Trương Tấn Sang

**MỤC LỤC**

**I, LỜI MỞ ĐẦU 1**

**II. Tìm hiểu về GIT2**

**II.1 Đôi nét về Git.2**

**II.2 Các câu lệnh cơ bản với Git3**

**II.3 Git conventional message……………………………………………………………………..… 4**

**II.4 Sử lý Git conflict………………………………………………………………………………….5**

**III. TÌM HIỂU VỀ CSS,HTML,JAVA SCRIPT…………………………………………………….....6**

**III.1 Tìm hiểu HTML…………………………………………………………………………………6**

**A)Tổng Quan……………………………………………………………………………………7**

**B) Các thành phần chính……………………………………………………………………….7**

**C) Các chức năng chính………………………………………………………………………...8**

**III.2 Tìm hiểu CSS……………………………………………………………………………………8**

**A) Tổng quan…………………………………………………………………………………….8**

**B) Các thành phần và chức năng chính………………………………………………………..9**

**III.3 Tìm hiểu JAVA SCRIPT………………………………………………………………………10**

**A) Tổng quan…………………………………………………………………………………..11**

**B) Các đặc điểm chính………………………………………………………………………...11**

**C) Các thành phần và chức năng chính………………………………………………………12**

**I, LỜI MỞ ĐẦU**

Trong thời đại số hóa và phát triển công nghệ nhanh chóng ngày nay, quản lý mã nguồn và làm việc cộng tác trong các dự án phần mềm trở nên quan trọng hơn bao giờ hết. Mã nguồn là cốt lõi của mọi dự án phần mềm, và việc quản lý nó một cách hiệu quả là một yếu tố quyết định cho sự thành công của dự án. Trong bài thu hoạch này, em sẽ trình bày những hiểu biết của mình về Git – hệ thống quản lý mã nguồn được rất nhiều lập trình viên tin tưởng .

Sau khi hoàn thành bài tập của bài thu hoạch 2 ( Thực hiện tạo project và thực hiện các thao tác tạo nhánh, pul, push, tạo merge request,...), em sẽ tiếp tục trình bày và thực hiện bài tập của bài thu hoạch 3( Thực hiện build static web có đủ 3 thành phần: header, body, footer. Push source code lên git ) lên project của bài thu hoạch 2.

**II. Tìm hiểu về Git**

**1. Đôi nét về Git**

Git là một hệ thống quản lý mã nguồn phiên bản phổ biến và mạnh mẽ. Nó được sáng tạo bởi Linus Torvalds vào năm 2005 và đã trở thành một trong những công cụ quản lý mã nguồn phổ biến nhất và không thể thiếu đối với các dự án phần mềm, cũng như các dự án công nghệ hàng đầu trên thế giới.

Một số ưu điểm của git: Tích hợp dễ dàng, khả năng khắc phục xung đột, hiệu suất cao,…

**2. Các câu lệnh cơ bản khi sử dụng Git**

| **Lệnh Git** | **Mô tả** |
| --- | --- |
| **git init** | Khởi tạo một kho lưu trữ Git mới. |
| **git clone [url]** | Sao chép một kho lưu trữ từ URL xa về máy tính. |
| **git add [file(s)]** | Thêm tệp/thư mục vào vùng chờ (staging area). |
| **git commit -m "msg"** | Tạo một commit với thông điệp mô tả thay đổi. |
| **git status** | Xem trạng thái hiện tại của các tệp. |
| **git log** | Xem lịch sử commit của kho lưu trữ. |
| **git branch** | Liệt kê tất cả các nhánh hiện có. |
| **git checkout [branch/commit]** | Di chuyển đến một nhánh hoặc commit cụ thể. |
| **git pull** | Lấy và cập nhật thay đổi từ kho lưu trữ xa. |
| **git push** | Đẩy các commit đã tạo lên kho lưu trữ xa. |
| **git merge [branch]** | Hợp nhất (merge) một nhánh vào nhánh hiện tại. |
| **git remote -v** | Liệt kê các kho lưu trữ xa đã liên kết. |
| **git fetch** | Lấy thông tin về các thay đổi từ kho lưu trữ xa mà không cập nhật thư mục làm việc. |
| **git reset [file/commit]** | Hủy bỏ các thay đổi trong vùng chờ hoặc trên commit cụ thể. |
| **git diff** | Xem sự khác biệt giữa phiên bản hiện tại và phiên bản trước đó của các tệp. |

Top of Form

**3. Conventional Commit Messages**

Conventional Commit Messages (CCM) là một hệ thống tiêu chuẩn hóa cho việc viết thông điệp commit khi làm việc với các hệ thống quản lý phiên bản (version control systems) như Git. Mục tiêu của CCM là giúp làm cho lịch sử commit của một dự án trở nên dễ đọc, dễ hiểu và dễ quản lý hơn.

Các thông điệp commit trong CCM thường bao gồm ba phần chính:

Type (Loại): Một từ khoá miêu tả loại thay đổi đã được thực hiện. Các ví dụ phổ biến của loại bao gồm "feat" (để mô tả một tính năng mới), "fix" (để mô tả một sửa lỗi), "docs" (để mô tả các thay đổi trong tài liệu), và nhiều loại khác.

Scope (Phạm vi, tầm ảnh hưởng): Tùy chọn, mô tả phạm vi của thay đổi. Ví dụ, nó có thể là tên của một thành phần cụ thể của dự án, hoặc một phạm vi khác nhau như "config," "tests," hoặc "build."

Description (Mô tả): Một mô tả ngắn gọn về những thay đổi đã được thực hiện. Nên sử dụng mô tả rõ ràng và đủ ngắn gọn để hiểu được sự thay đổi.

Một ví dụ của Conventional Commit Messages khi sử dụng Git Bash :

A black screen with white text

Description automatically generated

Trong đó :

- "feat(user-auth): add password reset functionality" là thông điệp commit sử dụng Conventional Commit Messages.

- "feat" là loại thay đổi, cho biết rằng một tính năng mới đã được thêm vào.

- "user-auth" là phạm vi hoặc phạm vi của thay đổi, cho biết rằng nó liên quan đến xác thực người dùng.

- "add password reset functionality" là mô tả ngắn gọn về thay đổi cụ thể.

**4. Sử lý Conflict trong Git**

**A) Định nghĩa**

Conflict xảy ra khi có xung đột giữa hai hoặc nhiều phiên bản của cùng một tệp trong quá trình merge (gộp) hoặc rebase (đặt lại cơ sở). Xung đột thường xảy ra khi Git không thể tự động tự động hợp nhất các thay đổi từ hai nguồn khác nhau do các thay đổi đối lập. Dưới đây là một số trường hợp phổ biến khi xung đột xảy ra trong Git: Xung đột merge (Merge Conflict), Xung đột rebase (Rebase Conflict), Xung đột cherry-pick: (Cherry-pick Conflict), Xung đột stash (Stash Conflict).

A computer screen with text

Description automatically generated

**B) Hướng giải quyết**

Sau đây là bước cơ bản để giải quyết Git conflict:

Bước 1: Sử dụng lệnh “git pull “ để tải về các thay đổi trên github

Bước 2: Mở tệp bị xung đột: Sử dụng trình soạn thảo của bạn, mở tệp bị xung đột (thường là tệp mà bạn đang thử merge, rebase hoặc cherry-pick) để chỉnh sửa nó.

Bước 3: Xác định và giải quyết xung đột: Trong tệp, bạn sẽ thấy các đoạn mã mà Git đánh dấu là xung đột. Thường, Git sẽ thêm nhãn <<<<<<<, =======, và >>>>>>> để xác định phần bị xung đột. Bạn cần chỉnh sửa tệp để giải quyết xung đột. Xoá các dòng không cần thiết, chỉnh sửa hoặc thay đổi mã để làm cho tệp hoàn thiện mà không còn xung đột.

Ví dụ:

A screenshot of a computer

Description automatically generated

Bước 4: Lưu tệp đã chỉnh sửa: Khi bạn đã giải quyết xung đột, lưu tệp lại.

Bước 5: Hoàn thành quá trình merge, rebase hoặc cherry-pick: Tiếp tục thực hiện quá trình merge, rebase hoặc cherry-pick mà bạn đang thực hiện ban đầu.

Bước 6: Hoàn thành quá trình: Sau khi bạn đã giải quyết xung đột và commit, bạn có thể hoàn thành quá trình merge, rebase hoặc cherry-pick bằng cách tiếp tục lệnh ban đầu mà bạn đang thực hiện.

**II. TÌM HIỂU HTML, CSS**

**1. HTML**

**A) Tổng quan**

HTML (Hypertext Markup Language) là một ngôn ngữ đánh dấu (markup language) được sử dụng để tạo và cấu trúc nội dung của trang web. HTML cho phép bạn định nghĩa cấu trúc và tổ chức của trang web bằng cách sử dụng các thẻ (tags) và các phần tử (elements) để đánh dấu các phần khác nhau của trang web. HTML cũng được sử dụng để tạo liên kết giữa các trang web và chèn các phương tiện như hình ảnh, video và âm thanh.

Khi phát triển và thiết kế một trang web, HTML được sử dụng kèm với CSS và JS cũng như 1 số thư viện và framework để có một sản phẩm hoàn chỉnh

**B) Các thành phần chính:**

| **Thành phần** | **Mô tả** |
| --- | --- |
| Thẻ (**<tag>**) | Đánh dấu và bao quanh các phần tử trên trang web. |
| Phần tử (**<element>**) | Bao gồm một cặp thẻ mở và thẻ đóng và nội dung bên trong. |
| Thuộc tính (**attributes**) | Cung cấp thông tin bổ sung cho các phần tử HTML. |
| Nội dung (**content**) | Những gì nằm bên trong thẻ mở và thẻ đóng của phần tử. |
| Phần đầu (**<head>**) | Chứa thông tin về trang web, chẳng hạn như tiêu đề và liên kết. |
| Phần thân (**<body>**) | Chứa nội dung hiển thị trực tiếp trên trình duyệt. |
| Tiêu đề (**<h1>**, **<h2>**,...) | Xác định tiêu đề của trang hoặc phần tiêu đề con. |
| Liên kết (**<a>**) | Tạo liên kết đến các trang web khác hoặc trang trong cùng trang web. |
| Hình ảnh (**<img>**) | Nhúng hình ảnh vào trang web. |
| Danh sách (**<ul>**, **<ol>**) | Tạo danh sách có hoặc không có thứ tự. |
| Bảng (**<table>**) | Tạo bảng dữ liệu trên trang web. |
| Biểu đồ và biểu đồ (**<form>**, **<input>**,...) | Tạo biểu đồ và biểu đồ. |
| Phần tử định nghĩa cấu trúc (**<header>**, **<nav>**,...) | Cung cấp thông tin cấu trúc. |
| Phần tử đánh dấu (**<em>**, **<strong>**,...) | Đánh dấu văn bản. |
| Phần tử HTML5 đa phương tiện (**<audio>**, **<video>**) | Nhúng âm thanh và video. |

**C) Các đặc điểm chính:**

- Ngôn ngữ đánh dấu: HTML là một ngôn ngữ đánh dấu, nghĩa là nó sử dụng các thẻ (tags) để xác định các phần tử và cấu trúc của trang web. Các thẻ HTML được đặt bên trong các cặp dấu " < " và " > ".

- Cấu trúc cơ bản: Trang web HTML cơ bản bao gồm hai phần chính: phần đầu (head) và phần thân (body). Phần đầu chứa các thông tin về trang web như tiêu đề, meta tags, và liên kết đến các tệp CSS và JavaScript. Phần thân chứa nội dung hiển thị trên trang web.

- Thẻ và phần tử: Các thẻ HTML đánh dấu các phần tử trên trang web. Ví dụ, thẻ <p> được sử dụng để đánh dấu một đoạn văn bản, thẻ <a> được sử dụng cho liên kết, và thẻ <img> để nhúng hình ảnh.

- Thuộc tính: Các phần tử HTML có thể có các thuộc tính (attributes) để cung cấp thông tin bổ sung về phần tử đó. Ví dụ, thuộc tính src trong thẻ <img> được sử dụng để chỉ định đường dẫn đến hình ảnh.

- Cấu trúc lồng nhau: HTML cho phép bạn lồng các phần tử bên trong nhau để tạo cấu trúc phức tạp hơn. Ví dụ, bạn có thể đặt một đoạn văn bản <p> bên trong một phần tử danh sách <ul> để tạo danh sách không sắp xếp.

- Liên kết và hình ảnh: HTML cho phép bạn tạo liên kết đến các trang web khác và nhúng hình ảnh, video, và âm thanh vào trang web của bạn. Điều này giúp tạo nội dung đa phương tiện và tương tác.-

- Phiên bản HTML: HTML có nhiều phiên bản khác nhau, với HTML5 là phiên bản mới nhất và phổ biến nhất. HTML5 cung cấp nhiều tính năng mới, bao gồm hỗ trợ nâng cao cho đa phương tiện, định nghĩa các phần tử cấu trúc như <header>, <nav>, và <footer>, và hỗ trợ tích hợp các ứng dụng web động.

- Tích hợp với CSS và JavaScript: HTML thường được kết hợp với CSS để kiểm soát giao diện và với JavaScript để thêm tính năng động vào trang web.

**2. CSS**

**A) Tổng quan về CSS**

CSS (Cascading Style Sheets) là một ngôn ngữ sử dụng để mô tả cách các tài liệu HTML (và các ngôn ngữ đánh dấu khác) được hiển thị trên một trình duyệt web. CSS được sử dụng để kiểm soát giao diện và bố cục của trang web, bao gồm màu sắc, kích thước, khoảng cách, vị trí và các thuộc tính khác của các phần tử trang web.

CSS bao gồm 3 loại chính:

- CSS Nội tuyến (Inline CSS): CSS nội tuyến là việc đặt mã CSS trực tiếp trong các phần tử HTML bằng cách sử dụng thuộc tính style.

Ưu điểm của CSS nội tuyến là nó dễ dàng áp dụng cho từng phần tử cụ thể, nhưng nó không được khuyến nghị sử dụng rộng rãi vì nó làm tăng kích thước mã HTML và làm cho việc quản lý và bảo trì khó khăn hơn

- CSS Ngoại tuyến (External CSS): CSS ngoại tuyến là việc viết mã CSS trong các tệp CSS riêng biệt, sau đó liên kết tệp CSS này với tài liệu HTML bằng thẻ <link>.

Trong tệp CSS (styles.css), bạn có thể đặt tất cả các quy tắc CSS của bạn. Ưu điểm của CSS ngoại tuyến là nó giúp duy trì sự phân chia giữa nội dung và kiểu dáng, dễ dàng quản lý và tái sử dụng mã CSS trên nhiều trang.

- CSS Nội bộ (Internal CSS): CSS nội bộ được đặt trong cùng tài liệu HTML bằng thẻ <style> trong phần đầu (<head>) của tài liệu.

Ưu điểm của CSS nội bộ là nó cho phép bạn định nghĩa kiểu dáng cho một trang cụ thể mà không cần tạo tệp CSS riêng biệt. Tuy nhiên, nó cũng có thể gây rối và làm tăng kích thước tài liệu HTML nếu sử dụng nhiều mã CSS nội bộ.

**B) Các thành phần và đặc điểm chính**

| **Thành phần CSS** | **Mô tả** |
| --- | --- |
| Selector (Bộ chọn) | Xác định các phần tử HTML mà bạn muốn áp dụng CSS. |
| Property (Thuộc tính) | Định nghĩa các thuộc tính mà bạn muốn thay đổi. |
| Value (Giá trị) | Xác định giá trị cho thuộc tính CSS. |
| Declaration (Khai báo) | Gồm selector, property và value, định nghĩa cách áp dụng CSS vào phần tử HTML. |
| Rule (Quy tắc) | Kết hợp một hoặc nhiều declarations để áp dụng cho một selector cụ thể. |
| Box Model (Mô hình hộp) | Quy định kích thước và cấu trúc của một phần tử HTML, bao gồm padding, border và margin. |
| Display (Hiển thị) | Xác định cách phần tử HTML được hiển thị trên trang web, ví dụ: **block**, **inline**, **none**. |
| Position (Vị trí) | Điều chỉnh vị trí của một phần tử, bao gồm **static**, **relative**, **absolute**, và **fixed**. |
| Color (Màu sắc) | Định nghĩa màu sắc cho văn bản và phông chữ. |
| Font (Phông chữ) | Quy định kiểu phông chữ, kích thước, độ dày và nhiều thuộc tính khác liên quan đến văn bản. |
| Background (Nền) | Định nghĩa hình ảnh hoặc màu nền cho phần tử. |

**3. Java Script**

**A) Tổng quan**

- Định nghĩa: JavaScript (viết tắt là JS) là một ngôn ngữ lập trình phía client (thường được thực thi trên trình duyệt web) và được sử dụng để tạo ra các tương tác trên trang web. Nó là một ngôn ngữ kịch bản đa mục đích và được sử dụng rộng rãi để thêm tính năng động và tương tác cho các trang web và ứng dụng web.

JavaScript đã trở thành một trong những ngôn ngữ lập trình phía client phổ biến nhất trên thế giới và đóng vai trò quan trọng trong việc xây dựng các trang web và ứng dụng web động, tương tác.

**B) Đặc điểm chính:**

Dưới đây là một số đặc điểm và khía cạnh quan trọng của JavaScript:

- Ngôn ngữ lập trình dựa trên văn bản (Scripting Language): JavaScript được viết dưới dạng mã nguồn văn bản và được thực thi bởi trình duyệt web của người dùng.

- Ngôn ngữ thông dịch (Interpreted Language): JavaScript là một ngôn ngữ thông dịch, có nghĩa là mã nguồn JavaScript được thực thi một dòng code tại một thời điểm, thay vì biên dịch toàn bộ mã nguồn trước khi chạy.

- Tích hợp trong trình duyệt: JavaScript là một phần quan trọng của môi trường trình duyệt web và có khả năng tương tác trực tiếp với HTML và CSS để thay đổi nội dung và hoạt động của trang web.

- Đa nền tảng (Cross-platform): JavaScript có khả năng chạy trên nhiều trình duyệt web và hệ điều hành khác nhau, làm cho nó trở thành một công cụ lý tưởng để phát triển ứng dụng web đa nền tảng.

- Hỗ trợ thư viện và framework: Có nhiều thư viện và framework JavaScript mạnh mẽ như React, Angular, và Vue.js, giúp phát triển ứng dụng web một cách hiệu quả và có cấu trúc.

- Có thể sử dụng cho cả phía máy chủ: JavaScript không chỉ được sử dụng ở phía máy khách (client-side) mà còn ở phía máy chủ (server-side) thông qua các nền tảng như Node.js.

**B) Các Thành phần và chức năng chính của JS**

| **Thành Phần/Chức Năng** | **Mô Tả** |
| --- | --- |
| Biến (Variables) | Lưu trữ và thao tác dữ liệu trong chương trình. |
| Kiểu dữ liệu (Data Types) | Hỗ trợ các kiểu dữ liệu như số (number), chuỗi (string), boolean, mảng (array), và đối tượng (object), và nhiều kiểu dữ liệu khác. |
| Câu lệnh điều kiện (Conditional Statements) | Cho phép kiểm tra và thực thi mã dựa trên điều kiện, ví dụ: **if**, **else if**, và **else**. |
| Vòng lặp (Loops) | Lặp lại các dòng mã nhiều lần, ví dụ: **for**, **while**, **do...while**. |
| Hàm (Functions) | Đóng gói khối mã và sử dụng nó nhiều lần, có thể định nghĩa và gọi hàm. |
| Mảng (Arrays) | Lưu trữ tập hợp các giá trị trong một biến, cung cấp các phương thức làm việc với mảng. |
| Đối tượng (Objects) | Tổ chức dữ liệu theo dạng các thuộc tính và phương thức, cơ sở cho lập trình hướng đối tượng. |
| Sự kiện (Events) | Xử lý sự kiện trên trang web, như bấm nút chuột, gõ phím, hoặc tương tác người dùng khác. |
| DOM (Document Object Model) | Tương tác với các phần tử HTML và thay đổi nội dung và cấu trúc của trang web. |
| AJAX (Asynchronous JavaScript and XML) | Trao đổi dữ liệu với máy chủ mà không cần phải làm tươi trang toàn bộ trang web. |
| API (Application Programming Interface) | Giao tiếp với các dịch vụ và tài nguyên bên ngoài như API của mạng xã hội, Google Maps, và các dịch vụ web khác. |
| Thư viện và Frameworks | Sử dụng thư viện và frameworks như jQuery, React, Angular, và Vue.js để phát triển ứng dụng web. |