



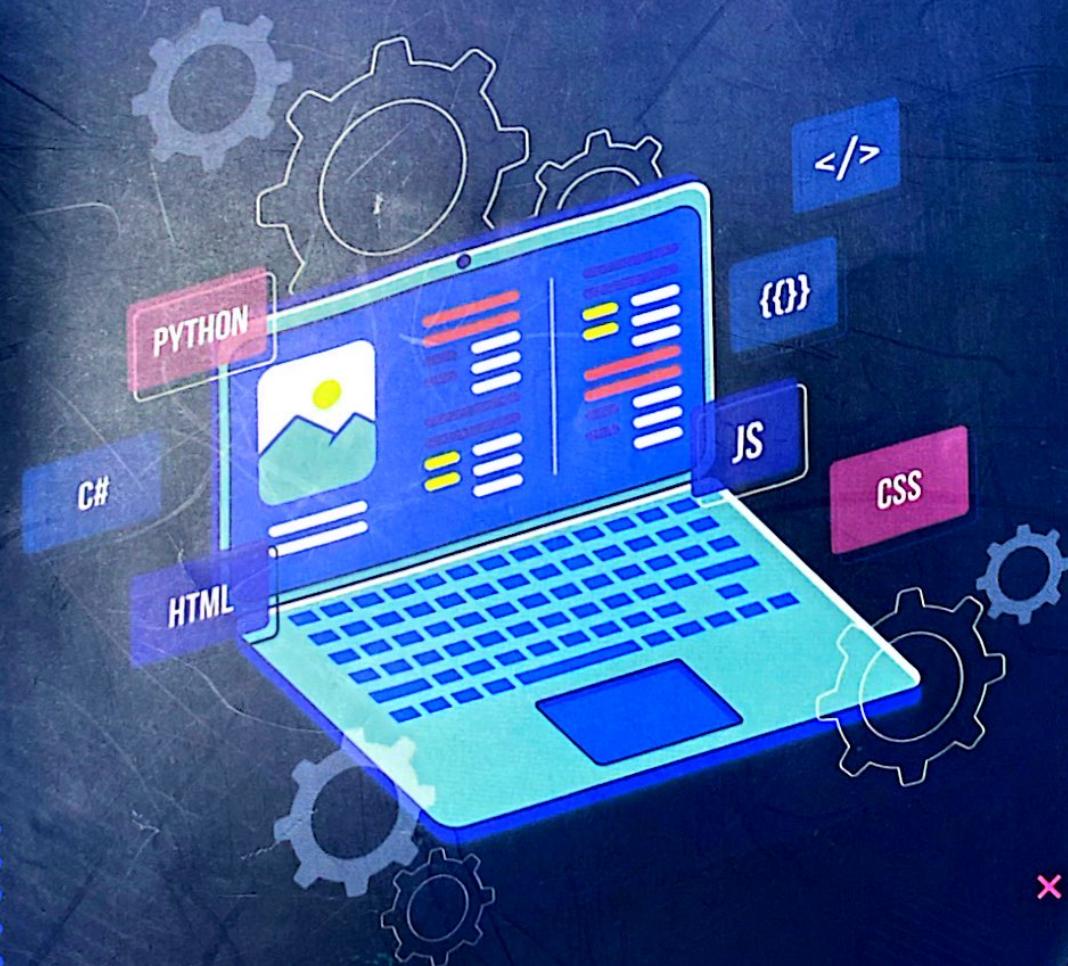
Tác giả: Đào Xuân Hiệp

# SỔ TAY “DÂN IT”

# LẬP TRÌNH WEB

# LẬP TRÌNH ỨNG DỤNG

TỪ CƠ BẢN ĐẾN CHUYÊN SÂU



NHÀ XUẤT BẢN THANH NIÊN

# MỤC LỤC

## Lời mở đầu

4

## PHẦN I LẬP TRÌNH WEB

### Chương 1: Giới thiệu về lập trình Web

|  |    |
|--|----|
| » Lịch Sử và Phát Triển Của Web  | 6  |
| » Xu Hướng Và Nền Tảng Của Công Nghệ Web Trong Tương Lai                               | 7  |
| » Các Ngôn Ngữ Lập Trình Phổ Biến Cho Web: HTML/CSS, JavaScript, Python, PHP, Java,... | 9  |
| » Thống Kê Về Mức Lương Và Nhu Cầu Về Công Việc Lập Trình Web                          | 19 |

### Chương 2: Cơ Bản về HTML và CSS

|                                   |    |
|-----------------------------------|----|
| » Cấu Trúc Của Trang Web: HTML    | 22 |
| » Giao Diện và Thiết Kế: CSS      | 38 |
| » Tạo Bố Cục Linh Hoạt và Hấp Dẫn | 41 |
| » Bài tập thực hành HTML và CSS   | 47 |

### Chương 3: JavaScript và DOM Manipulation

|   |    |
|---|----|
| » Cú Pháp Cơ Bản Của JavaScript                   | 60 |
| » Trình Biên Dịch Sử Dụng Để Lập Trình JavaScript | 61 |
| » Thao Tác Với DOM: Sự Kiện, Phần Tử, và CSS      | 66 |

|                                |    |
|--------------------------------|----|
| » Xử Lý Sự Kiện và Hiệu Ứng    | 68 |
| » Bài tập thực hành JavaScript | 75 |

## **Chương 4: Ajax và Asynchronous Programming**

|   |    |
|---|----|
| » Gửi Yêu Cầu Đến Máy Chủ                   | 85 |
| » Xử Lý Dữ Liệu JSON và API                 | 88 |
| » Cải Thiện Trải Nghiệm Người Dùng Với Ajax | 90 |

## **Chương 5: Cơ Sở Dữ Liệu và SQL, MySQL**

|  |     |
|--|-----|
| » Nguyên Lý Cơ Bản Của Cơ Sở Dữ Liệu             | 98  |
| » Ngôn Ngữ Truy Vấn SQL                          | 104 |
| » Kết Nối và Thao Tác Dữ Liệu Trong Ứng Dụng Web | 109 |
| » Bài tập thực hành SQL và MySQL                 | 116 |

# **PHẦN 2**

## **LẬP TRÌNH WEB CHUYÊN SÂU VÀ ỨNG DỤNG THỰC TẾ**

### **Chương 6: Frameworks và Thư Viện Phổ Biến**

|  |     |
|--|-----|
| » Giới Thiệu Về React, Angular, và Vue.js    | 129 |
| » Sử Dụng Thư Viện jQuery và Lodash          | 133 |
| » Kiến Thức Cơ Bản Về Frameworks và Thư Viện | 137 |
| » Bài Tập Thực Hành                          | 139 |

### **Chương 7: Authentication và Security**

|  |     |
|--|-----|
| » Xác Thực Người Dùng và Tài Khoản                         | 143 |
| » Bảo Mật Ứng Dụng Web: XSS, CSRF, SQL Injection           | 145 |
| » Sử Dụng HTTPS và Bảo Mật Dữ Liệu Trong Lập Trình Website | 147 |
| » Bài Tập Thực Hành  | 150 |

## **Chương 8: Responsive Web Design và Mobile Development**

|  |           |
|--|-----------|
| » Thiết Kế Đáp Ứng Cho Các Thiết Bị Di Động và<br>Màn Hình Nhỏ | 15        |
| » Bài Tập Cơ Bản và Nâng Cao                                   | 16        |
| » Sử Dụng Bootstrap và Media Queries                           | 16        |
| » Bài Tập Chuyên Sâu   | 17        |
| » Phát Triển Ứng Dụng Di Động Với React Native                 | 17        |
| » Lập Trình Website Với Mobile DeveLopment                     | 18        |
| <b>Giải thích từ khóa</b>                                      | <b>20</b> |

# **PHẦN I:**

# **LẬP TRÌNH**

# **WEB**





## GIỚI THIỆU VỀ LẬP TRÌNH WEB

### LỊCH SỬ PHÁT TRIỂN CỦA LẬP TRÌNH WEB

**THỜI KỲ** | Trước năm  
**TIỀN WEB** | **1990**

**1960 - 1970:** Xuất hiện các ý tưởng đầu tiên về một hệ thống thông tin toàn cầu.  
**1980:** Các mô hình đầu tiên của internet và các giao thức như TCP/IP được phát triển.

**1989:** Tim Berners-Lee phát minh World Wide Web (WWW) tại CERN, Thụy Sĩ.

**1991:** Trình duyệt web đầu tiên được phát triển.

**1993:** Mosaic, trình duyệt web đầu tiên dành cho người dùng thông thường, được phát hành.

**1994:** Các trang web đầu tiên với đuôi .com được tạo ra.

**1995:** Netscape Navigator, một trong những trình duyệt phổ biến nhất thời điểm đó, ra mắt.

**THỜI KỲ** | **2000** - **2010**

**2000:** Bubble Dot-com (sự phô trương của thị trường công nghệ internet) và sụp đổ của nhiều công ty internet.  
**2004:** Sự ra đời của các dịch vụ web 2.0 như Facebook, YouTube, và các trang blog như WordPress.

**1990 - 2000 | THỜI KỲ  
WEB ĐẦU TIÊN**



**2006:** Bắt đầu xu hướng các ứng dụng web tương tác, đánh dấu sự xuất hiện của các ứng dụng dựa trên trình duyệt (web apps).

**2010s:** Sự gia tăng của các ứng dụng di động, kết hợp giữa web và thiết bị di động.

**2015:** Xu hướng Progressive Web Apps (PWA) được giới thiệu, cung cấp trải nghiệm như ứng dụng di động trên trình duyệt web.

**2020:** Tăng cường với công nghệ như trí tuệ nhân tạo (AI), thực tế ảo và tăng cường (VR/AR) trong các ứng dụng web.

**Hiện tại:** Phát triển của web 3.0 với sự hợp nhất của blockchain, đánh dấu sự xuất hiện của các ứng dụng phi tập trung và các dịch vụ tài chính phi tập trung.

Sau năm **2010** | **THỜI KỲ  
WEB 3.0  
VÀ SAU NÀY**



Có thể thấy rằng web đã trải qua một hành trình dài từ các trang tĩnh đơn giản đến các ứng dụng tương tác phức tạp, và nó vẫn tiếp tục phát triển với sự hỗ trợ của các công nghệ mới và sự sáng tạo của cộng đồng phát triển web trên toàn thế giới.

### XU HƯỚNG VÀ NỀN TẢNG CỦA CÔNG NGHỆ WEB TRONG TƯƠNG LAI

Công nghệ lập trình web liên tục phát triển, dưới đây là một số xu hướng và nền tảng quan trọng trong lĩnh vực này:

#### ► Frontend Frameworks và Thư Viện

**React.js:** React.js, được phát triển bởi Facebook, là một thư viện JavaScript phổ biến cho việc xây dựng giao diện người dùng động và linh hoạt.

**Angular:** Angular, được phát triển bởi Google, là một framework JavaScript đầy đủ tính năng, được sử dụng để xây dựng các ứng dụng web phức tạp và linh hoạt.

**Vue.js:** Vue.js là một framework JavaScript nhẹ, dễ học và sử dụng, phổ biến trong việc xây dựng các ứng dụng web tương tác và động.

#### ► Server-side Frameworks

**Node.js:** Node.js là một môi trường chạy mã JavaScript phía máy chủ, cho phép viết mã JavaScript cả phía máy khách và máy chủ, tạo điều kiện thuận lợi cho việc xây dựng ứng dụng web đơn trang và ứng dụng thời gian thực.

**Django:** Django là một framework Python đầy đủ tính năng, được thiết kế để xây dựng các ứng dụng web nhanh chóng và bảo mật.

**Ruby on Rails:** Ruby on Rails, thường được gọi là Rails, là một framework Ruby phổ biến với cấu trúc MVC (Model-View-Controller) và tập trung vào việc tối giản quá trình phát triển ứng dụng web.

#### ► GraphQL và REST API

**GraphQL:** GraphQL là một ngôn ngữ truy vấn dành cho API, cho phép người phát triển yêu cầu chỉ các dữ liệu cần thiết, giúp tối ưu hóa tốc độ truy vấn và giảm băng thông.

**REST API:** REST (Representational State Transfer) vẫn là một kiểu kiến trúc phổ biến cho việc xây dựng các API, tập trung vào tài nguyên và các thao tác CRUD (Create, Read, Update, Delete).

#### ► WebAssembly và Progressive Web Apps (PWA)

**WebAssembly:** WebAssembly (Wasm) là một mã máy ảo dành cho web, cho phép chạy mã ngôn ngữ khác như C, C++, và Rust trên trình duyệt web, tăng tốc độ xử lý và hiệu suất của ứng dụng web.

**Progressive Web Apps (PWA):** PWA là các ứng dụng web được thiết kế để cung cấp trải nghiệm giống như ứng dụng di động, bao gồm tính năng như làm việc offline, thông báo và truy cập nhanh thông qua màn hình chính.

#### ► Công Nghệ Liên Quan

**Blockchain và Smart Contracts:** Công nghệ blockchain được tích hợp vào các ứng dụng web để tăng tính minh bạch, bảo mật giao dịch và quản lý dữ liệu.

**Machine Learning và AI:** Công nghệ trí tuệ nhân tạo và học máy được sử dụng để tối ưu hóa trải nghiệm người dùng, dự đoán hành vi người dùng và cung cấp các tính năng cá nhân hóa.

**Nền Tảng và Công Cụ Đám Mây:** Dịch vụ đám mây như AWS, Azure và Google Cloud cung cấp các công cụ và nền tảng hỗ trợ phát triển, triển khai và quản lý ứng dụng web một cách linh hoạt và tiện lợi.

*Những xu hướng và nền tảng này định hình tương lai của công nghệ lập trình web, tạo điều kiện cho việc phát triển các ứng dụng web đa dạng và mạnh mẽ.*

## CÁC NGÔN NGỮ LẬP TRÌNH PHỔ BIẾN CHO WEB: HTML/CSS, JavaScript, Python, PHP, Java,...

### 1. Ngôn ngữ lập trình HTML

#### HTML (HyperText Markup Language):

HTML là ngôn ngữ đánh dấu siêu văn bản tiêu chuẩn được sử dụng để tạo cấu trúc cơ bản của một trang web. HTML sử dụng các thẻ (tags) để xác định các thành phần trên trang web. Dưới đây là một ví dụ về cách HTML tạo một đoạn văn bản đơn giản:



```
<!DOCTYPE html>
<html>
<head>
    <title>Trang Web Của Tôi</title>
</head>
<body>
    <h1>Chào mừng bạn đến với trang web của tôi!</h1>
    <p>Đây là một đoạn văn bản trong một đoạn đơn giản.</p>
</body>
</html>
```

**<!DOCTYPE html>**: Xác định phiên bản HTML được sử dụng.

**<html>**: Bắt đầu và kết thúc một tài liệu HTML.

**<head>**: Chứa các thông tin về tài liệu như tiêu đề trang, liên kết đến các tập tin CSS, và các thông tin meta khác.

**<body>**: Chứa nội dung hiển thị trên trang web.

## 2. Ngôn ngữ lập trình CSS

**CSS (Cascading Style Sheets)**: CSS là ngôn ngữ lập trình đơn giản được sử dụng để định dạng và trang trí cho các trang web HTML. CSS cho phép bạn kiểm soát màu sắc, font chữ, khoảng cách, độ rộng, chiều cao và các thuộc tính khác của các phần tử HTML. Dưới đây là một ví dụ đơn giản về việc sử dụng CSS:

```
h1 {
    color: blue;
    font-family: Arial, sans-serif;
}

p {
    color: green;
    font-size: 16px;
}
```

Trong ví dụ trên, CSS được sử dụng để định dạng các phần tử **h1** và **p**. Đoạn mã trên đặt màu chữ của **h1** thành màu xanh dương và chữ của **p** thành màu xanh lá cây, và đặt kích thước font của **p** là 16 pixel.

**HTML và CSS thường được sử dụng cùng nhau để tạo ra các trang web đẹp và tương tác. HTML xác định cấu trúc và nội dung của trang web, trong khi CSS xác định giao diện và kiểu dáng của trang web đó.**

## 3. Ngôn ngữ lập trình JavaScript

**JavaScript**: JavaScript là một ngôn ngữ lập trình thông dịch (interpreted) đa nhiệm, đa nền tảng (multi-paradigm) và độc lập với trình duyệt (browser-independent). Ngôn ngữ này cho phép bạn tạo ra các trang web tương tác và động, có khả năng thực hiện các hành động như kiểm tra dữ liệu đầu vào, tương tác với người dùng và thay đổi nội dung của trang web mà không cần tải lại trang.

### ► Các đặc điểm cơ bản của JavaScript

**Độ Dễ (Loose Typing)**: JavaScript không yêu cầu xác định kiểu dữ liệu của biến khi khai báo chúng. Kiểu dữ liệu của biến có thể thay đổi trong quá trình thực thi của chương trình.

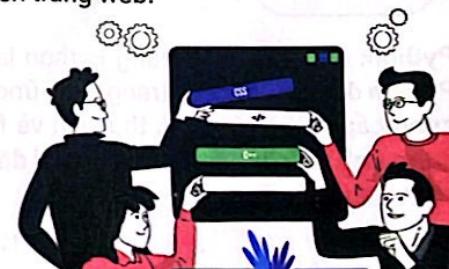
**Hàm (Functions)**: JavaScript hỗ trợ việc khai báo hàm, cho phép gói gọn mã lệnh vào các đoạn mã có thể được gọi lại một cách dễ dàng.

**Đối Tượng (Objects)**: JavaScript là một ngôn ngữ dựa trên đối tượng, có nghĩa là mọi thứ trong JavaScript đều là một đối tượng hoặc có thể được chuyển đổi thành đối tượng.

**Sự Kiện (Events)**: JavaScript cho phép xử lý các sự kiện như click chuột, nhập liệu từ bàn phím, và nhiều sự kiện khác trên trang web.

### DOM (Document Object Model)

JavaScript sử dụng DOM để tương tác với các phần tử trên trang web. DOM cho phép bạn thay đổi nội dung, cấu trúc và kiểu dáng của trang web.



► Ví dụ đơn về JavaScript

```
// JavaScript source code
<!DOCTYPE html>
<html>
<head>
    <title>JavaScript Example</title>
    <script>
        function showMessage() {
            alert("Hello, World!");
        }
    </script>
</head>
<body>
    <button onclick="showMessage()">Click Me!</button>
</body>
</html>
```

Trong ví dụ này, JavaScript được sử dụng để định nghĩa hàm `showMessage()`, và khi người dùng nhấn vào nút, hộp thoại thông báo sẽ xuất hiện với nội dung "Hello, World!"

*JavaScript không chỉ được sử dụng trong trình duyệt web mà còn trong nhiều môi trường lập trình khác như Node.js để phát triển các ứng dụng máy chủ.*

## 4. Python

**Python:** Lập trình web bằng Python là quá trình sử dụng ngôn ngữ lập trình Python để xây dựng các trang web, ứng dụng web, và các dịch vụ web. Python cung cấp nhiều công cụ, thư viện và framework mạnh mẽ để phát triển ứng dụng web hiện đại và linh hoạt. Dưới đây là một số khía cạnh quan trọng về lập trình web bằng Python:

## 4.1. Python Frameworks cho Lập Trình Web

### 4.1.1 | Django

**Django:** Django là một framework lập trình web Python mạnh mẽ và linh hoạt. Nó cung cấp một cấu trúc dự án tổ chức tốt, hỗ trợ quản lý tài nguyên như cơ sở dữ liệu, và đi kèm với các tính năng bảo mật tích hợp. Đây là một ví dụ đơn giản về cách tạo một trang web Django:

- Cài đặt Django: `pip install django`
- Ví dụ về ứng dụng Django

```
# views.py
from django.http import HttpResponse

def hello(request):
    return HttpResponse("Hello, World!")

# urls.py
from django.urls import path
from . import views

urlpatterns = [
    path('hello/', views.hello, name='hello'),
]
```

### 4.1.2 | Flask

**Flask:** Flask là một micro framework Python nhẹ nhàng và dễ sử dụng. Nó không yêu cầu cấu hình phức tạp và giúp bạn xây dựng các ứng dụng web một cách linh hoạt. Flask được thiết kế để làm cho việc bắt đầu với lập trình web Python trở nên dễ dàng.

- Cài đặt Flask: `pip install flask`
- Ví dụ về ứng dụng Flask

```
from flask import Flask
app = Flask(__name__)

@app.route('/')
def hello():
    return "Hello, World!"
```

#### 4.1.3 | FastAPI

**FastAPI:** FastAPI là một framework Python mới nhưng rất nhanh và hiệu quả cho việc xây dựng các API. Nó kết hợp sức mạnh của Python với kiểu dữ liệu được kiểm tra tĩnh để tạo ra các API hiệu quả và an toàn.

- ▶ **Cài đặt FastAPI:** pip install fastapi uvicorn
- ▶ **Ví dụ về ứng dụng FastAPI**

```
from fastapi import FastAPI
app = FastAPI()

@app.get('/')
def read_root():
    return {"Hello": "World"}
from fastapi import FastAPI
app = FastAPI()

@app.get('/')
def read_root():
    return {"Hello": "World"}
```

## 4.2. Các Công Cụ và Thư Viện

**Requests:** Thư viện Requests cho phép gửi các yêu cầu HTTP/1.1 dễ dàng, là một cách tiện lợi để tương tác với các API và các trang web khác.

**Beautiful Soup:** Là một thư viện cho việc trích xuất dữ liệu từ các trang web. Nó giúp phân tích cú pháp (parse) HTML và XML một cách dễ dàng.

**SQLAlchemy:** Thư viện này cung cấp một giao diện SQL linh hoạt cho Python, giúp làm việc với cơ sở dữ liệu một cách thuận tiện và bảo mật.

## 4.3. Ứng Dụng và Dịch Vụ Web Python Phổ Biến

**Instagram:** Một trong những mạng xã hội lớn nhất thế giới được xây dựng bằng Python và Django.

**Youtube:** YouTube sử dụng Python trong hệ thống gợi ý video và xử lý dữ liệu người dùng.

**Dropbox:** Dịch vụ lưu trữ đám mây phổ biến này được viết bằng Python.

## 4.4. Ưu Điểm của Lập Trình Web Bằng Python

**Đọc mã dễ hiểu:** Python có cú pháp đơn giản và rõ ràng, làm cho mã nguồn dễ đọc, hiểu và duy trì.

**Nhiều thư viện và framework:** Python cung cấp nhiều thư viện và framework mạnh mẽ giúp giảm thiểu thời gian và công sức khi phát triển các ứng dụng web.

**Cộng đồng lớn:** Python có một cộng đồng lớn và tích cực, điều này có nghĩa là có nhiều tài liệu, hỗ trợ và tài nguyên sẵn có để giúp đỡ khi gặp vấn đề.

Với những ưu điểm và công cụ mạnh mẽ của mình, Python đã trở thành một trong những lựa chọn phổ biến cho việc phát triển các ứng dụng web và dịch vụ trực tuyến.

## 5. Ngôn ngữ lập trình PHP

**PHP** (viết tắt của *Hypertext Preprocessor*) là một ngôn ngữ lập trình Script (kịch bản) được sử dụng phổ biến trong phát triển ứng dụng web động.

Dưới đây là một số điểm quan trọng về PHP:

### ► Lịch Sử và Xuất Xứ

- PHP được tạo ra bởi Rasmus Lerdorf vào năm 1994 như một dự án cá nhân để quản lý trang web cá nhân của anh ấy.
- PHP ở đầu tiên là viết tắt của "Personal Home Page," nhưng sau đó được đổi lại thành "Hypertext Preprocessor."

### ► Cú Pháp và Tính Năng

- PHP được viết trực tiếp trong mã HTML, giúp chuyển đổi các trang HTML thành các trang web động có thể tương tác với cơ sở dữ liệu.
- PHP hỗ trợ nhiều loại cơ sở dữ liệu, bao gồm MySQL, PostgreSQL, SQLite và nhiều hơn nữa.
- Nó hỗ trợ các thư viện mạnh mẽ cho việc xử lý chuỗi, tệp tin, và các thao tác liên quan đến web.

### ► Sử Dụng và Ứng Dụng

- PHP được sử dụng chủ yếu trong việc phát triển các ứng dụng web, bao gồm trang web cá nhân, trang web doanh nghiệp, diễn đàn, và các ứng dụng thương mại điện tử.
- Nó thường được kết hợp với MySQL để tạo ra các hệ thống quản lý cơ sở dữ liệu động.

### ► Framework PHP Phổ Biến

**Laravel:** Laravel là một framework PHP mạnh mẽ với cú pháp rõ ràng, hỗ trợ các tính năng như routing, ORM, và giao diện người dùng đẹp mắt.

**Symfony:** Symfony là một framework PHP đầy đủ tính năng, được thiết kế để xây dựng các ứng dụng web phức tạp và linh hoạt.

**CodeIgniter:** CodeIgniter là một framework PHP nhẹ nhàng và nhanh chóng, thích hợp cho các dự án nhỏ và đòi hỏi tài nguyên ít.

### ► Ưu Điểm Của PHP

**Dễ Học và Sử Dụng:** Cú pháp rõ ràng và dễ hiểu, giúp người mới học lập trình dễ dàng tiếp cận.

**Tương Thích Rộng Rãi:** PHP hoạt động trên hầu hết các hệ thống máy chủ và hỗ trợ nhiều cơ sở dữ liệu phổ biến.

**Cộng Đồng Lớn:** PHP có một cộng đồng đông đảo và hỗ trợ, điều này đồng nghĩa với việc có nhiều tài liệu và giúp đỡ trực tuyến.

*PHP vẫn là một trong những ngôn ngữ lập trình phổ biến nhất trong lĩnh vực phát triển ứng dụng web động, và nó tiếp tục đóng vai trò quan trọng trong ngành công nghệ thông tin. Đây cũng là 1 ngôn ngữ chính mình sẽ phân tích chuyên sâu trong cuốn sách này!*

## 6. Ngôn ngữ lập trình Java

Java là một ngôn ngữ lập trình phổ biến và mạnh mẽ, được thiết kế để làm cho việc phát triển ứng dụng linh hoạt và dễ dàng.

Dưới đây là một số điểm quan trọng về Java:

### ► Lịch Sử và Xuất Xứ

- Java được phát triển bởi James Gosling và đồng đội tại Sun Microsystems vào những năm 1990.
- Java được giới thiệu chính thức vào năm 1995, và từ đó trở đi đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất thế giới.

### ► Tính Đa Nhiệm (Multithreading) và Độ Bền

- Java hỗ trợ lập trình đa nhiệm, cho phép viết các ứng dụng có thể xử lý nhiều tác vụ đồng thời.
- Java được thiết kế để chạy trên nhiều nền tảng khác nhau mà không cần thay đổi mã nguồn, điều này gọi là "Write Once, Run Anywhere" (WORA).

### ► Cú Pháp và Cấu Trúc

- Java có cú pháp rõ ràng và dễ đọc, giúp việc phát triển và duy trì mã nguồn trở nên dễ dàng.
- Nó sử dụng hệ thống kiểu dữ liệu tĩnh (static typing) và quản lý bộ nhớ tự động, giúp người lập trình tránh được nhiều lỗi thường gặp như tràn bộ nhớ (Memory Overflow).

### ► Ứng Dụng và Dự Án Của Java

**Ứng Dụng Desktop:** Java được sử dụng để phát triển các ứng dụng desktop đa nhiệm và linh hoạt.

**Ứng Dụng Web và Server-side:** Java là một trong những ngôn ngữ chính được sử dụng trong phát triển các ứng dụng web và dịch vụ server-side thông qua các framework như Spring và Hibernate.

**Ứng Dụng Di Động:** Java được sử dụng trong việc phát triển ứng dụng di động Android, là hệ thống di động phổ biến nhất trên thế giới.

### ► Cộng Đồng và Thư Viện

- Java có một cộng đồng lớn và đa dạng, với hàng ngàn tài liệu, diễn đàn và nguồn tài nguyên trực tuyến.
- Có nhiều thư viện và framework mạnh mẽ, giúp nhà phát triển xây dựng các ứng dụng phức tạp và hiệu quả.

**Java là một ngôn ngữ lập trình đa nhiệm, linh hoạt và phổ biến, được sử dụng trong nhiều lĩnh vực như phát triển ứng dụng Desktop, Web, Di động và Server-Side.**

## THỐNG KÊ VỀ MỨC LƯƠNG VÀ NHU CẦU VỀ CÔNG VIỆC LẬP TRÌNH WEB

### 1. Tại Việt Nam

#### ► Mức Lương

**Lương Bắt Đầu:** Lập trình viên web mới tốt nghiệp ở Việt Nam thường kiếm được từ 5 triệu đến 10 triệu VND mỗi tháng, tùy thuộc vào vị trí và kỹ năng.

**Lương Trung Bình:** Lập trình viên web với kinh nghiệm 2 - 5 năm có thể kiếm từ 10 triệu đến 20 triệu VND mỗi tháng.

**Lương Cao Nhất:** Các chuyên gia với kinh nghiệm trên 5 năm và kiến thức sâu rộng có thể kiếm được từ 20 triệu đến 50 triệu VND mỗi tháng.

#### ► Nhu Cầu Công Việc

**Tăng Trưởng Nghề Nghiệp:** Nhu cầu về lập trình viên web ở Việt Nam đang tăng mạnh, đặc biệt là trong lĩnh vực phát triển ứng dụng di động và các công nghệ web tiên tiến như React.js, Node.js và Angular.

**Nhu Cầu Chuyên Môn:** Có nhu cầu đặc biệt về các chuyên gia về các framework phổ biến như Laravel (PHP) và Django (Python). Công ty phần mềm và công ty công nghệ tại Việt Nam thường tìm kiếm những nhân viên có kỹ năng đa nhiệm và sự hiểu biết sâu về các công nghệ mới.

### 2. Tại Thế Giới

#### ► Mức Lương

**Lương Bắt Đầu:** Tại các quốc gia phát triển như Hoa Kỳ và các quốc gia châu Âu, lập trình viên web mới tốt nghiệp thường kiếm được từ \$50,000 đến \$80,000 mỗi năm.



**Lương Trung Bình:** Lập trình viên web với kinh nghiệm 2 - 5 năm có thể kiếm từ \$80,000 đến \$120,000 mỗi năm.

**Lương Cao Nhất:** Các chuyên gia và các vị trí quản lý có thể kiếm được từ \$120,000 trở lên mỗi năm, đặc biệt là ở các thành phố như San Francisco và New York City.

#### ► Nhu Cầu Công Việc

**Tăng Trưởng Nghề Nghiệp:** Nhu cầu về lập trình viên web ở các quốc gia phát triển vẫn tiếp tục tăng mạnh, đặc biệt trong các công nghệ như React.js, Angular, và Node.js.

**Nhu Cầu Chuyên Môn:** Có nhu cầu đặc biệt về các chuyên gia về trí tuệ nhân tạo, phân tích dữ liệu, và công nghệ blockchain. Công ty công nghệ, công ty tài chính và các công ty khởi nghiệp ở các thành phố công nghệ lớn đều tìm kiếm những chuyên gia có kiến thức sâu rộng về lập trình web và công nghệ liên quan.

### 3. Những lưu ý để học và làm việc tốt đối với vị trí lập trình web

#### ► Học

**Tập Trung Vào Kiến Thức Cơ Bản:** Hiểu sâu về HTML, CSS và JavaScript là quan trọng. Đây là nền tảng của mọi ứng dụng web.

**Thực Hành Liên Tục:** Lập trình web yêu cầu việc thực hành liên tục. Xây dựng các dự án nhỏ và tham gia vào các dự án mã nguồn mở.

**Học Cách Đọc Tài Liệu (Documentation):** Biết cách đọc và hiểu tài liệu của các thư viện và framework giúp bạn nhanh chóng tiếp cận công nghệ mới.

**Hiểu Về Cấu Trúc Dữ Liệu và Thuật Toán:** Kiến thức về cấu trúc dữ liệu và thuật toán giúp bạn viết mã hiệu quả và hiệu năng cao.

**Git và Quản Lý Phiên Bản:** Nắm vững việc sử dụng Git và các dịch vụ quản lý phiên bản như GitHub để làm việc nhóm và theo dõi thay đổi mã nguồn.

**Làm Quen Với Một Số Framework và Thư Viện:** Học một hoặc vài framework như React, Angular hoặc Vue.js để xây dựng ứng dụng phức tạp và dễ bảo trì.

**Hiểu Về Các Nguyên Tắc Thiết Kế (Design Principles):** Hiểu về UI/UX design giúp bạn xây dựng các trang web hấp dẫn và dễ sử dụng.

#### ► Làm Việc

**Thảo Luận và Trao Đổi Kiến Thức:** Tham gia vào cộng đồng lập trình, diễn đàn trực tuyến và các sự kiện lập trình để học hỏi từ người khác và giải quyết vấn đề cùng nhau.

**Luyện Tập Kỹ Năng Tự Học:** Học cách tự giải quyết vấn đề, tự đọc tài liệu và tự học công nghệ mới. Lập trình web luôn đòi hỏi việc học suốt đời.

**Chấp Nhận và Học Từ Lỗi:** Mỗi lỗi là một cơ hội học hỏi. Đừng sợ thất bại, hãy tìm hiểu từ những sai lầm và cố gắng sửa chúng.

**Thực Hành Sự Linh Hoạt (Agility):** Công nghệ liên tục thay đổi. Hãy linh hoạt và sẵn lòng học hỏi và chuyển đổi sang công nghệ mới khi cần thiết.

**Hiểu Rõ Yêu Cầu Kinh Doanh (Business Requirements):** Lập trình web thường liên quan đến việc giải quyết các vấn đề kinh doanh. Hiểu rõ yêu cầu của khách hàng giúp bạn viết mã nguồn hữu ích và hiệu quả.

**Tham Gia Vào Dự Án Thực Tế:** Tham gia vào các dự án thực tế, thậm chí làm việc freelance để có kinh nghiệm thực tế và xây dựng portfolio.

**Chăm Sóc Sức Khỏe:** Làm việc lâu dài đòi hỏi sức khỏe tốt. Hãy đảm bảo bạn duy trì lịch trình làm việc cân đối, tập thể dục và chăm sóc tinh thần của mình.

*Nhớ rằng, hành trình học và làm việc trong lĩnh vực lập trình web là một hành trình liên tục của sự học hỏi và phát triển. Luôn mở lòng và sẵn lòng thay đổi để theo kịp với sự phát triển không ngừng của ngành công nghệ thông tin.*



## CƠ BẢN VỀ HTML VÀ CSS

### CẤU TRÚC CỦA TRANG WEB: HTML

Cấu trúc của một trang web HTML thường bao gồm nhiều thành phần để định hình giao diện và nội dung. Dưới đây là một cấu trúc cơ bản của một trang web HTML:

```
<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Tiêu đề của Trang Web</title>
    <link rel="stylesheet" href="styles.css"> 
</head>
<body>
    <header>
        <nav>
            <ul>
                <li><a href="#">Trang Chủ</a></li>
                <li><a href="#">Dịch Vụ</a></li>
                <li><a href="#">Giới Thiệu</a></li>
                <li><a href="#">Liên Hệ</a></li>
            </ul>
        </nav>
    </header>
    <main>
        <section id="intro">
            <h1>Xin chào!</h1>
            <p>Chào mừng bạn đến với trang web của tôi.</p>
        </section>
        <section id="features">
            <h2>Tính Năng</h2>
        </section>
    </main>
</body>
```

```
<ul>
    <li>Tính Năng 1</li>
    <li>Tính Năng 2</li>
    <li>Tính Năng 3</li>
</ul>
</section>
</main>
<footer>
    <p>© 2023 Tên Công Ty. Bản quyền được bảo lưu.</p>
</footer>
</body>
</html>
```

**<!DOCTYPE html>**: Khai báo phiên bản HTML.

**<html lang="vi">**: Định nghĩa ngôn ngữ của trang web (ở đây là tiếng Việt).

**<head>**: Chứa các thông tin về trang web như kích thước chữ, tiêu đề trang, và các liên kết tới các tệp CSS hoặc JavaScript.

**<body>**: Chứa nội dung hiển thị trên trang web.

**<header>**: Khu vực chứa định hình phần đầu trang web, bao gồm menu điều hướng.

**<nav>**: Định hình phần menu điều hướng.

**<main>**: Khu vực chứa nội dung chính của trang web.

**<section>**: Phân chia nội dung thành các phần nhỏ, giúp tổ chức trang web.

**<footer>**: Khu vực chứa thông tin chân trang như bản quyền và liên hệ.

Trong ví dụ trên, tệp CSS được liên kết từ một tệp riêng biệt (**styles.css**). Điều này cho phép bạn tách biệt CSS từ mã HTML, giúp quản lý giao diện trang web dễ dàng hơn.

Trình biên dịch sử dụng để lập trình HTML/CSS: Có nhiều công cụ IDE (Integrated Development Environment) và trình soạn thảo mã nguồn code mạnh mẽ được sử dụng để lập trình HTML và CSS.

## 1. Visual Studio Code (VS Code)

- ▶ **Trang Chủ:** Visual Studio Code
- ▶ **Link tải:** <https://code.visualstudio.com/download>
- ▶ **Tính Năng:**
  - Hỗ trợ mã màu (syntax highlighting) cho HTML, CSS, và nhiều ngôn ngữ khác.
  - Hỗ trợ tự động hoàn thiện mã (auto-completion).
  - Gợi ý lệnh (command suggestion) cho các lệnh và thuộc tính CSS.
  - Cài đặt các extension mở rộng để tăng khả năng mở rộng và tính năng.

## 2. Sublime Text

- ▶ **Trang Chủ:** Sublime Text
- ▶ **Link tải:** <https://www.sublimetext.com/3>
- ▶ **Tính Năng:**
  - Giao diện đẹp và tinh tế.
  - Tích hợp nhanh và mượt mà.
  - Hỗ trợ nhiều ngôn ngữ lập trình, bao gồm HTML và CSS.
  - Cài đặt các packages mở rộng để mở rộng tính năng.

## 3. Atom

- ▶ **Trang Chủ:** Atom
- ▶ **Link tải:** <https://github.blog/2022-06-08-sunsetting-atom/>
- ▶ **Tính Năng:**
  - Phát triển bởi GitHub, Atom là một trình soạn thảo mã nguồn mở.
  - Hỗ trợ tùy chỉnh giao diện và chức năng thông qua các packages.
  - Hỗ trợ git integration và syntax highlighting cho HTML, CSS và nhiều ngôn ngữ khác.

## 4. Brackets

- ▶ **Trang Chủ:** Brackets
- ▶ **Link tải:** <https://brackets.io/>
- ▶ **Tính Năng:**
  - Được thiết kế đặc biệt cho lập trình web với tính năng Extract, giúp trích xuất thông tin từ các file PSD và tạo mã CSS từ đó.
  - Hỗ trợ Live Preview để xem trực tiếp sự thay đổi trong trình duyệt khi bạn chỉnh sửa mã.

## 5. Notepad++

- ▶ **Trang Chủ:** Notepad++
- ▶ **Link tải:** <https://notepad-plus-plus.org/>
- ▶ **Tính Năng:**
  - Giao diện đơn giản và dễ sử dụng.
  - Hỗ trợ syntax highlighting cho nhiều ngôn ngữ lập trình, bao gồm HTML và CSS.
  - Hỗ trợ tùy chỉnh thông qua plugins.

Tất cả các công cụ trên đều hỗ trợ lập trình HTML và CSS. Lựa chọn công cụ phụ thuộc vào sở thích cá nhân và yêu cầu công việc cụ thể của bạn.

Dưới đây là một bảng liệt kê các thẻ HTML cơ bản và các thuộc tính CSS tương ứng để giúp bạn theo dõi hành trình của mình khi học HTML và CSS. Bảng này không bao gồm tất cả các thẻ và thuộc tính có thể sử dụng, nhưng nó bao gồm những cái quan trọng và thường xuyên được sử dụng trong phát triển web các bạn cần học và hiểu nó để có thể tuỳ biến khi phát triển ứng dụng web kết hợp nhiều ngôn ngữ.

| Thẻ HTML     | Mô Tả  | Thuộc Tính CSS         |
|--------------|--|------------------------|
| <!DOCTYPE>   | Định dạng phiên bản HTML của trang web                       |                        |
| <html>       | Thẻ gói bên ngoài tất cả các nội dung trên trang web         |                        |
| <head>       | Chứa các tiêu đề, liên kết tới các tệp CSS, JavaScript, v.v. |                        |
| <title>      | Đặt tiêu đề cho trang web                                    |                        |
| <meta>       | Thông tin về trang web, như mô tả và tác giả                 |                        |
| <link>       | Liên kết với các tệp CSS                                     |                        |
| <style>      | Định dạng CSS nội tuyến                                      |                        |
| <script>     | Liên kết với các tệp JavaScript                              |                        |
| <body>       | Chứa toàn bộ nội dung hiển thị trên trang web                |                        |
| <h1> to <h6> | Tiêu đề cấp độ 1 đến cấp độ 6                                | font-size, font-weight |
| <p>          | Đoạn văn bản   | margin, padding        |
| <a>          | Liên kết   | text-decoration, color |
| <img>        | Hiển thị hình ảnh  | width, height          |
| <ul>         | Danh sách không thứ tự                                       | list-style-type        |
| <ol>         | Danh sách có thứ tự  | list-style-type        |
| <li>         | Mục danh sách  |                        |

| Thẻ HTML   | Mô Tả  | Thuộc Tính CSS  |
|------------|--|-----------------|
| <table>    | Bảng   | border, width   |
| <tr>       | Dòng trong bảng                              |                 |
| <th>       | Ô tiêu đề trong bảng                         | text-align      |
| <td>       | Ô dữ liệu trong bảng                         | text-align      |
| <form>     | Mẫu (form) để gửi dữ liệu                    |                 |
| <input>    | Trường nhập liệu                             | width, height   |
| <textarea> | Ô văn bản lớn                                | width, height   |
| <button>   | Nút  | width, height   |
| <label>    | Nhãn cho các trường nhập liệu                |                 |
| <select>   | Hộp dropdown                                 | width, height   |
| <option>   | Tùy chọn trong hộp dropdown                  |                 |
| <div>      | Hộp chứa cho CSS và JavaScript               | margin, padding |
| <span>     | Phần nhỏ của văn bản hoặc hình ảnh           |                 |
| <hr>       | Đường kẻ ngang (horizontal rule)             | border, width   |
| <br>       | Xuống dòng (line break)                      |                 |
| <meta>     | Thông tin về trang web, như mô tả và tác giả |                 |

## 6. Thủ (Tag)

Một thủ trong HTML là một phần của mã nguồn HTML, bắt đầu với dấu `<` và kết thúc bằng dấu `>`. Thủ thường đánh dấu các phần tử trên trang web và mô tả loại dữ liệu hoặc nội dung của phần tử đó.

► Ví dụ

```
<p>
```

Đây là một đoạn văn bản.`</p>`

Trong ví dụ trên, `<p>` là một thủ mở (bắt đầu một phần tử), và `</p>` là một thủ đóng (kết thúc phần tử).

## 7. Thuộc Tính (Attributes)

Thuộc tính cung cấp thông tin bổ sung về phần tử HTML và thường được đặt trong thủ mở. Thuộc tính giúp xác định các đặc tính hoặc hành vi của phần tử đó.

► Ví dụ

```
<a href="https://www.example.com" title="Trang chính">
```

Đây là liên kết đến trang chính.`</a>`

Trong ví dụ trên, `href` và `title` là các thuộc tính của thủ `<a>`. Thuộc tính `href` chỉ định địa chỉ liên kết, và thuộc tính `title` cung cấp thông tin gợi ý khi di chuột qua liên kết.

## 8. Giá Trị (Value)

Trong một số thuộc tính, bạn cũng cần xác định giá trị cho thuộc tính đó. Giá trị nằm trong dấu nháy kép (`" "`) và cung cấp giá trị cụ thể cho thuộc tính.

► Ví dụ

```
<input type="text" value="Nhập văn bản ở đây">
```

Trong ví dụ trên, "Nhập văn bản ở đây" là giá trị của thuộc tính `value` của thủ `<input>`.

Những khái niệm về thủ, thuộc tính và giá trị này rất quan trọng khi bạn xây dựng các trang web và muốn hiểu cách cấu trúc và tương tác với các yếu tố trên trang.

## 9. Cấu Trúc Cơ Bản của Trang HTML

### 9.1. <!DOCTYPE>

`<!DOCTYPE>` (Document Type Declaration) khai báo phiên bản HTML được sử dụng trong tài liệu. Nó giúp trình duyệt hiểu cách phải hiển thị trang web, dựa trên phiên bản HTML được sử dụng.

► Ví dụ

```
<!DOCTYPE html>
```

### 9.2. <html>

`<html>` đánh dấu đầu và cuối của tài liệu HTML. Mọi phần tử trên trang web đều nằm trong thủ `<html>`.

► Ví dụ

```
<html lang="vi">
...
</html>
```

### 9.3. <head>

`<head>` chứa các phần thông tin về tài liệu như tiêu đề, kích thước chữ, tệp CSS hoặc JavaScript được liên kết, và các thủ meta cho SEO.

► Ví dụ

```

<head>
    <title>Tiêu đề trang web</title>
    <link rel="stylesheet" href="styles.css">
    <meta charset="UTF-8">
    <meta name="description" content="Mô tả trang web">
    ...
</head>

```

#### 9.4. <title>

**<title>** xác định tiêu đề của trang web, hiển thị trên thanh tiêu đề của trình duyệt. Đây là thông tin quan trọng cho SEO và trải nghiệm người dùng.

► Ví dụ

```
<title>Tiêu đề trang web</title>
```

#### 9.5. <body>

**<body>** chứa nội dung hiển thị của trang web, bao gồm văn bản, hình ảnh, liên kết và các phần tử đa phương tiện.

► Ví dụ

```

body>
    <h1>Đây là tiêu đề cấp 1</h1>
    <p>Đây là một đoạn văn bản.</p>
    
    ...
</body>

```

Các thành phần này đều là bắt buộc trong một trang HTML và đóng vai trò quan trọng trong việc xây dựng cấu trúc cơ bản của một trang web.

Tiêu đề và đoạn văn bản: **<h1>, <p>, <br>, <hr>**.

Các thẻ HTML như **<h1>, <p>, <br>, và <hr>** được sử dụng để định dạng và hiển thị tiêu đề, đoạn văn bản, dòng trống và đường ngang trong trang web.

#### 9.6. <h1>, <h2>, <h3>, <h4>, <h5>, <h6> - Tiêu Đề

Thẻ **<h1>** đến **<h6>** được sử dụng để tạo tiêu đề với độ quan trọng giảm dần từ **<h1>** (quy mô lớn nhất) đến **<h6>** (quy mô nhỏ nhất).

► Ví dụ

html

```

<h1>Đây là tiêu đề cấp 1 </h1>
<h2>Đây là tiêu đề cấp 2 </h2>
<h3>Đây là tiêu đề cấp 3 </h3>

```

#### 9.7. <p> - Đoạn Văn Bản

Thẻ **<p>** được sử dụng để định dạng và hiển thị một đoạn văn bản.

► Ví dụ

html

```
<p>Đây là một đoạn văn bản. </p>
```

#### 9.8. <br> - Dòng Trống

Thẻ **<br>** tạo ra một dòng trống trong văn bản, đưa các phần tử tiếp theo xuống dòng mới.

► Ví dụ

html

```
<p>Đây là dòng thứ nhất. <br> Đây là dòng thứ hai </p>
```

#### 9.9. <hr> - Đường Ngang

Thẻ **<hr>** tạo ra một đường ngang (horizontal rule) để phân chia các phần của trang web.

## ► Ví dụ

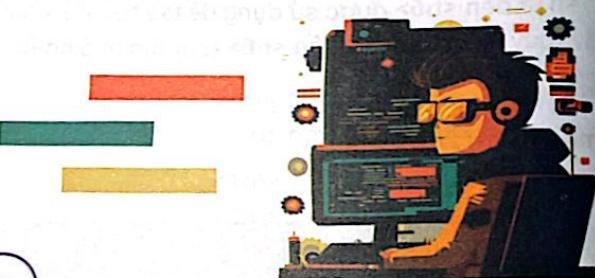
**html**

<p> Đây là phần văn bản trước đường ngang. </p>

<hr>

<p> Đây là phần văn bản sau đường ngang. </p>

Những thẻ này giúp bạn tổ chức và hiển thị nội dung trang web một cách hợp lý, làm cho trang web của bạn trở nên dễ đọc và trực quan hơn.

**10. HTML nâng cao****10.1. HTML Forms (Biểu Mẫu HTML)**

## ► Validation và Constraints (Xác nhận và Ràng Buộc)

**Mã code: HTML**

```
<form>
  <label for="email">Email:</label>
  <input type="email" id="email" name="email" required pattern="[^@]+@[a-zA-Z]{2,}\.+[a-zA-Z]{2,}$">
  <button type="submit">Submit</button>
</form>
```

Trong ví dụ này, **type="email"** đánh dấu một ô đầu vào là địa chỉ **email** và **pattern** định nghĩa một biểu thức chính quy để kiểm tra địa chỉ **email** hợp lệ.

## ► Input Types (Loại Đầu Vào)

**Mã code: HTML**

```
<input type="date">
<input type="color">
<input type="range" min="0" max="100" step="5">
```

Các loại đầu vào **date**, **color**, và **range** cho phép người dùng chọn ngày, màu sắc, hoặc giá trị trong một phạm vi cụ thể.

**10.2. HTML Multimedia (Đa Phương Tiện HTML)**

## ► Audio và Video

**Mã code: HTML**

```
<audio controls>
  <source src="audio.mp3" type="audio/mpeg">
  Your browser does not support the audio element.
</audio>

<video width="320" height="240" controls>
  <source src="movie.mp4" type="video/mp4">
  Your browser does not support the video element.
</video>
```

Trong ví dụ này, **controls** hiển thị thanh điều khiển cho audio và video. Bạn cũng có thể thiết lập các định dạng và thuộc tính khác cho **<source>**.

## ▶ Ảnh và Hình Ảnh Động

Mã code: HTML

```


```

Sử dụng thẻ `<img>` để hiển thị hình ảnh và hình ảnh động. Thuộc tính `alt` cung cấp mô tả cho hình ảnh.

## 10.3. HTML Semantic Elements (Các Phần Tử Ngữ Cảnh)

Mã code: HTML

```
<header>
  <h1>Website Title</h1>
  <nav>
    <ul>
      <li><a href="#home">Home</a></li>
      <li><a href="#about">About</a></li>
      <li><a href="#contact">Contact</a></li>
    </ul>
  </nav>
</header>

<article>
  <h2>Article Title</h2>
  <p>Content of the article goes here.</p>
</article>
```

Các phần tử `<header>`, `<nav>`, và `<article>` được sử dụng để tạo cấu trúc ngữ cảnh cho trang web của bạn, giúp các công cụ tìm kiếm và trình đọc màn hình hiểu cấu trúc trang web.

## 10.4. HTML Tables (Bảng HTML)

Mã code: HTML

```
<table>
  <tr>
    <th>Header 1</th>
    <th>Header 2</th>
  </tr>
  <tr>
    <td>Row 1, Cell 1</td>
    <td>Row 1, Cell 2</td>
  </tr>
  <tr>
    <td>Row 2, Cell 1</td>
    <td>Row 2, Cell 2</td>
  </tr>
</table>
```

Sử dụng thẻ `<table>` và các thẻ như `<tr>`, `<th>`, và `<td>` để tạo bảng trên trang web.

## 10.5. HTML Drag and Drop (Kéo và Thả HTML)

Kéo và thả là một tính năng rất phổ biến. Bạn có thể chọn một đối tượng và di chuyển nó đến một vị trí khác. Trong HTML5, kéo/thả là một phần của chuẩn: bất cứ phần tử nào cũng có thể thực hiện việc kéo thả. Hầu hết các trình duyệt web hiện tại đều hỗ trợ tính năng này.

## Mã code: HTML

```

<div id="draggable" draggable="true">Drag me!</div>
<div id="droppable">Drop here!</div>

<script>
  const draggableElement = document.getElementById('draggable');
  const droppableElement = document.getElementById('droppable');

  draggableElement.ondragstart = (event) => {
    event.dataTransfer.setData('text/plain', event.target.id);
  };

  droppableElement.ondragover = (event) => {
    event.preventDefault();
  };

  droppableElement.ondrop = (event) => {
    const id = event.dataTransfer.getData('text');
    const draggableElement = document.getElementById(id);
    droppableElement.appendChild(draggableElement);
    event.dataTransfer.clearData();
  };
</script>

```

Trong ví dụ này, `draggable="true"` cho phép phần tử được kéo và sự kiện `ondragstart`, `ondragover`, và `ondrop` được sử dụng để xử lý các sự kiện kéo và thả.

## 10.6. HTML Canvas (Bảng Vẽ HTML)

Phần tử `<canvas>` trong HTML được sử dụng để vẽ các đối tượng đồ họa, nó chiếm một vùng trên trang web, qua một ngôn ngữ dạng script (thường là `JavaScript`). Phần tử `<canvas>` chỉ chứa khối đồ họa. Bạn phải dùng một script để vẽ các đối tượng đồ họa. Canvas cung cấp nhiều cách để vẽ đường thẳng, hình hộp, hình tròn, viết chữ, chèn ảnh,...

## Mã code: HTML

```

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

<script>
  const canvas = document.getElementById('myCanvas');
  const ctx = canvas.getContext('2d');
  ctx.fillStyle = 'green';
  ctx.fillRect(10, 10, 150, 80);
</script>

```

Sử dụng thẻ `<canvas>` và `JavaScript` để vẽ hình trực tiếp trên trang web.

## 10.7. HTML Geolocation (Vị trí Địa Lý HTML)

Trong HTML5, các hàm API về Geolocation giúp lấy vị trí (địa lý) của người dùng. Tuy nhiên để sử dụng được tính năng này bạn cần sự cho phép của người dùng. Định vị còn hoạt động phụ thuộc vào thiết bị mà người dùng truy cập, cần cung cấp được tọa độ vị trí ví dụ như GPS trong smart phone.

## ► Cú pháp sử dụng hàm định vị

`navigator.geolocation.getCurrentPosition(success, error);`

## ► Trong đó:

- **Success:** là một hàm callback do bạn định nghĩa, hàm này sẽ nhận được giá trị về đối tượng `Position` để định vị khi hàm `getCurrentPosition` gọi thành công

- **Error:** là một hàm callback do bạn định nghĩa, hàm này sẽ nhận được đối tượng `PositionError` nếu như hàm `getCurrentPosition` có lỗi.



## Mã code: HTML

```
<button onclick="getLocation()">Get Location</button>

<script>
    function getLocation() {
        if (navigator.geolocation) {
            navigator.geolocation.getCurrentPosition(showPosition);
        } else {
            alert('Geolocation is not supported by this browser.');
        }
    }

    function showPosition(position) {
        alert("Latitude: " + position.coords.latitude + "\nLongitude: " +
        position.coords.longitude);
    }
</script>
```

Trong ví dụ này, sử dụng `navigator.geolocation.getCurrentPosition()` để lấy vị trí địa lý của người dùng và hiển thị nó trong một cửa sổ cảnh báo.

## GIAO DIỆN VÀ THIẾT KẾ: CSS

CSS (Cascading Style Sheets) là ngôn ngữ định dạng và trang trí cho các trang web. Nó cho phép bạn kiểm soát giao diện và bố cục của trang web, bao gồm màu sắc, font chữ, kích thước, khoảng cách, và nhiều yếu tố thiết kế khác. Dưới đây là một số khái niệm cơ bản về CSS khi nó được sử dụng để thiết kế giao diện trang web:

### 1. Cách Liên Kết CSS với HTML

Có thể liên kết CSS với HTML thông qua một số cách:

► **CSS Ngoại Tuyến (External):** CSS được viết trong một tệp riêng biệt và được liên kết với HTML bằng thẻ `<link>` trong phần `<head>` của trang HTML.

```
<head>
    <link rel="stylesheet" href="styles.css">
</head>
```

► **CSS Nội Tuyến (Internal/Embedded):** CSS được viết bên trong thẻ `<style>` trong phần `<head>` của trang HTML.

```
<head>
    <style>
        p {
            color: blue;
        }
    </style>
</head>
```

► **CSS Nội Tuyến (Inline):** CSS được viết trực tiếp trong các phần tử HTML thông qua thuộc tính `style`.

```
<p style="color: red;">Văn bản màu đỏ</p>
```

### 2. Lựa Chọn Phần Tử (Selectors)

CSS sử dụng các lựa chọn để xác định các phần tử HTML mà bạn muốn áp dụng các quy tắc CSS.

► **Ví dụ**

- ▶ **Phân Tử HTML:** *p, h1, div.*
- ▶ **Lớp (Class):** *.class-name.*
- ▶ **ID:** *#unique-id.*
- ▶ **Lựa Chọn Kết Hợp:** *element.class.*

### 3. Thuộc Tính CSS Phổ Biến

- ▶ **Màu Sắc:** *color, background-color.*
- ▶ **Phông Chữ:** *font-family, font-size, font-weight.*
- ▶ **Bố Cục và Kích Thước:** *width, height, margin, padding.*
- ▶ **Hiển Thị:** *display.*
- ▶ **Đường Viền:** *border.*

### 4. Bố Cục và Định Dạng

- ▶ **Căn Lề (Margin và Padding):** Điều chỉnh khoảng cách xung quanh phần tử.
- ▶ **Hiển Thị và Độ Bao Phủ (Display and Float):** Điều khiển việc hiển thị và sắp xếp các phần tử.
- ▶ **Bố Cục Tự Động (Flexbox và Grid):** Các công cụ mạnh mẽ để xây dựng bố cục đàn hồi và đa chiều.

### 5. Độ Ưu Tiên (Specificity) và Kế Thừa

- ▶ **Độ Ưu Tiên:** Xác định quy tắc CSS nào sẽ được áp dụng khi có nhiều quy tắc áp dụng đối với cùng một phần tử.
- ▶ **Kế Thừa:** Một số thuộc tính CSS của phần tử cha có thể được kế thừa bởi các phần tử con, nhưng không phải tất cả.



40

CSS là một công cụ quan trọng để tạo ra trải nghiệm người dùng tốt trên các trang web. Bằng cách sử dụng CSS, bạn có thể tùy chỉnh giao diện người dùng và tạo ra các trang web trực quan, linh hoạt, và thú vị.

### 6. Trình soạn thảo mã nguồn HTML/CSS

- ▶ **Visual Studio Code:** Visual Studio Code là một trình soạn thảo mã nguồn mở miễn phí với hỗ trợ mạnh mẽ cho HTML, CSS, và JavaScript. Nó cung cấp gợi ý mã, kiểm tra lỗi, và các tiện ích mở rộng mạnh mẽ.
- ▶ **Sublime Text:** Sublime Text là một trình soạn thảo mã nguồn phổ biến với giao diện tinh tế và khả năng mở rộng. Nó hỗ trợ nhiều ngôn ngữ lập trình, bao gồm HTML và CSS.
- ▶ **Atom:** Atom là một trình soạn thảo mã nguồn mở được phát triển bởi GitHub. Nó có khả năng tùy chỉnh cao và hỗ trợ HTML và CSS thông qua các gói mở rộng.

## TẠO BỐ CỤC LINH HOẠT VÀ HẤP ĐÃNG

### 1. Sử Dụng CSS Grid và Flexbox

- ▶ **CSS Grid:** CSS Grid Layout là một công cụ mạnh mẽ cho việc tạo bố cục linh hoạt. Nó cho phép bạn xác định các hàng và cột, giúp tạo ra bố cục đa chiều một cách dễ dàng.

```
.container {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-gap: 20px;
}
```

41

- **Flexbox:** Flexbox là một kỹ thuật mạnh mẽ để sắp xếp và căn chỉnh các phần tử một cách dễ dàng, đặc biệt là trong việc xây dựng các thành phần giao diện người dùng.

```
.container {
  display: flex;
  justify-content: space-between;
}
```

## 2. Responsive Design (Thiết Kế Đáp Ứng)

- **Sử dụng Media Queries:** Sử dụng media queries để điều chỉnh bố cục và kiểu dáng dựa trên kích thước màn hình hoặc thiết bị.

```
@media (max-width: 768px) {
  .container {
    flex-direction: column;
  }
}
```

Sử dụng đơn vị linh hoạt như **percentage (%)** và **em/rem** để thiết lập kích thước và khoảng cách dựa trên phần trăm của kích thước cha hoặc font chữ.

## 3. Font và Màu Sắc Hợp Lý

- **Sử dụng font chữ đáp ứng (responsive fonts):** như **vw** (viewport width) hoặc **em** để font chữ thích ứng với kích thước màn hình.

```
body {
  font-size: 2vw;
}
```

- Lựa chọn **màu sắc phù hợp** và **tạo sự đối contrast** giữa văn bản và nền để cải thiện độ đọc.

```
body {
  font-size: 2vw;
}
```

## 4. Sử Dụng Thẻ Semantic HTML

- **Sử dụng các thẻ HTML semantic:** như **<header>**, **<nav>**, **<section>**, **<article>**, **<aside>**, **<footer>** để cải thiện độ đọc và hỗ trợ máy tìm kiếm hiểu cấu trúc trang web.

## 5. Thủ Nghiệm và Tối Ưu Hóa

- **Kiểm Tra Trên Nhiều Thiết Bị:** Kiểm tra trang web của bạn trên nhiều thiết bị và trình duyệt để đảm bảo rằng bố cục hoạt động đúng cách.
- **Hiệu Ứng và Chuyển Động Hợp Lý:** Sử dụng hiệu ứng và chuyển động nhưng hãy chú ý đến tốc độ tải trang, tránh sử dụng hiệu ứng quá nhiều có thể làm giảm trải nghiệm người dùng.

Tạo bố cục linh hoạt và hấp dẫn đòi hỏi kiến thức sâu rộng về CSS và HTML, và việc thử nghiệm và điều chỉnh liên tục để đảm bảo trang web hoạt động tốt trên mọi thiết bị và kích thước màn hình.

## 6. CSS nâng cao

**CSS nâng cao** cung cấp nhiều tính năng linh hoạt và mạnh mẽ để tùy chỉnh giao diện của trang web. Dưới đây là một số chủ đề CSS nâng cao chi tiết kèm ví dụ cụ thể:

## 6.1. CSS Selectors (Bộ Chọn CSS)

### ► Class Selectors (Bộ Chọn Lớp)

Mã code: css

```
.button {
    background-color: #4CAF50;
    color: white;
    padding: 10px 20px;
    border: none;
    border-radius: 5px;
}
```

Trong ví dụ này, `.button` là một lớp CSS được áp dụng cho các phần tử có class "button".

### ► ID Selectors (Bộ Chọn ID)

Mã code: css

```
#header {
    background-color: #333;
    color: white;
    padding: 10px;
}
```

Trong ví dụ này, `#header` là một ID CSS được áp dụng cho một phần tử có id "header".

## 6.2. CSS Pseudo-classes (Pseudo-classes CSS)

Mã code: css

```
a:hover {
    color: red;
}

input:focus {
    border-color: green;
}
```

Trong ví dụ này, `:hover` và `:focus` là các pseudo-classes CSS được sử dụng để tùy chỉnh giao diện khi người dùng di chuột qua hoặc tập trung vào các phần tử.

## 6.3. CSS Flexbox (Hộp Linh Hoạt CSS)

Mã code: css

```
.container {
    display: flex;
    justify-content: space-between;
}

.item {
    flex: 1;
    margin: 10px;
}
```

Trong ví dụ này, `.container` là một hộp linh hoạt, và `.item` là các phần tử con trong hộp này. `justify-content: space-between;` sẽ tạo ra khoảng trống đều giữa các phần tử con.

## 6.4. CSS Grid Layout (Bố Cục Lưới CSS)

Mã code: css

```
.container {  
    display: grid;  
    grid-template-columns: 1fr 1fr 1fr;  
    grid-gap: 10px;  
}  
  
.item {  
    padding: 20px;  
    text-align: center;  
}
```

Trong ví dụ này, `.container` là một lưới với 3 cột có chiều rộng bằng nhau, `grid-gap: 10px;` tạo ra khoảng trống 10px giữa các ô.

## 6.5. CSS Transitions (Chuyển Động CSS)

Mã code: css

```
.button {  
    background-color: #4CAF50;  
    color: white;  
    padding: 10px 20px;  
    border: none;  
    border-radius: 5px;  
    transition: background-color 0.3s ease;  
}  
  
.button:hover {  
    background-color: #45a049;  
}
```

Trong ví dụ này, khi di chuột qua `.button`, màu nền sẽ chuyển đổi một cách mềm dẻo trong 0.3 giây với hiệu ứng "ease".

## 6.6. CSS Animations (Hoạt Ánh CSS)

Mã code: css

```
@keyframes move {  
    0% { transform: translateX(0); }  
    50% { transform: translateX(100px); }  
    100% { transform: translateX(0); }  
}  
  
.box {  
    width: 100px;  
    height: 100px;  
    background-color: #4CAF50;  
    animation: move 2s infinite;  
}
```

Trong ví dụ này, `.box` sẽ thực hiện một hoạt ảnh dịch chuyển ngang từ vị trí ban đầu đến vị trí 100px qua trục X, rồi quay lại vị trí ban đầu. Hoạt ảnh sẽ kéo dài trong 2 giây và lặp vô hạn.

# BÀI TẬP THỰC HÀNH HTML/CSS

## 1. Bài tập cơ bản về HTML/CSS

### 1.1. Bài Tập 1: Tạo Trang Hồ Sơ Cá Nhân

#### Đề Bài:

► **Tạo một trang web hồ sơ cá nhân bao gồm các phần tử sau:**

- Hình ảnh cá nhân.
- Tên, tuổi, và quê quán.
- Một đoạn văn giới thiệu bản thân.
- Danh sách các kỹ năng (ví dụ: HTML, CSS, JavaScript).
- Liên kết đến các trang mạng xã hội của bạn (nếu có).

## ▶ Mã nguồn:

## 1.1.1 | File: index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Hồ Sơ Cá Nhân</title>
    <link rel="stylesheet" href="styles.css">
</head>

<body>
    <div class="profile">
        
        <h1>Nguyễn Văn A</h1>
        <p>Tuổi: 25</p>
        <p>Quê Quán: Hà Nội</p>
        <p>Chào mừng bạn đến với trang hồ sơ cá nhân của tôi. Tôi là một nhà phát triển web với kiến thức sâu rộng về HTML, CSS và JavaScript.</p>
        <h2>Kỹ Năng</h2>
        <ul>
            <li>HTML</li>
            <li>CSS</li>
            <li>JavaScript</li>
        </ul>
        <h2>Liên Hệ</h2>
        <p>Email: nguyenvana@example.com</p>
        <p>GitHub: <a href="https://github.com/">github.com/nguyenvana</a></p>
    </div>
</body>
</html>
```

## 1.1.2 | File: Style.css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}
```

```
profile {
    max-width: 600px;
    margin: 20px auto;
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

img {
    width: 150px;
    height: 150px;
    border-radius: 50%;
}
```

## 1.2. Bài Tập 2: Tạo Trang Danh Sách Sản Phẩm

## Đề Bài:

- ▶ **Tạo một trang web hiển thị danh sách sản phẩm bao gồm:** Tên sản phẩm, hình ảnh, và giá tiền.

## ▶ Mã nguồn:

## 1.2.1 | File: index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Danh Sách Sản Phẩm</title>
    <link rel="stylesheet" href="styles.css">
</head>
```

```
<body>
    <div class="product">
        <div class="product-item">
            
            <h2>Sản Phẩm 1</h2>
            <p>Giá: $20</p>
        </div>
        <div class="product-item">
            
            <h2>Sản Phẩm 2</h2>
            <p>Giá: $25</p>
        </div>
        <div class="product-item">
            
            <h2>Sản Phẩm 3</h2>
            <p>Giá: $30</p>
        </div>
    </div>
</body>

</html>
```

## 1.2.2 | File: Style.css

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

.product {
    display: flex;
    justify-content: space-around;
    max-width: 800px;
    margin: 20px auto;
}

.product-item {
    background-color: white;
```

50

```
padding: 20px;
border-radius: 10px;
box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
text-align: center;
}

img {
    width: 150px;
    height: 150px;
}
```

## 1.3. Bài Tập 3: Tạo Trang Đăng Nhập

## Đề Bài:

► **Tạo một trang web đăng nhập bao gồm:** các trường nhập tên người dùng và mật khẩu, cùng với nút "Đăng Nhập".

► **Mã nguồn:**

## 1.3.1 | File: index.html

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Đăng Nhập</title>
    <link rel="stylesheet" href="styles.css">
</head>

<body>
    <div class="login">
        <h1>Đăng Nhập</h1>
```

51

```

<form>
    <label for="username">Tên Người Dùng:</label>
    <input type="text" id="username" name="username" required><br>
    <label for="password">Mật Khẩu:</label>
    <input type="password" id="password" name="password" required><br>
    <button type="submit">Đăng Nhập</button>
</form>
</div>
</body>

</html>

```

## 1.3.2 | File Css: Style.css

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.login {
    background-color: white;
    padding: 40px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}

form {
    margin-top: 20px;
}

```



## 1.4. Bài Tập 4: Tạo Trang Web Responsive

## Đề Bài:

- Tạo một trang web: sử dụng media queries để trở nên responsive (tự điều chỉnh theo kích thước màn hình).

- Mã nguồn:

## 1.4.1 | File html: index.html

```

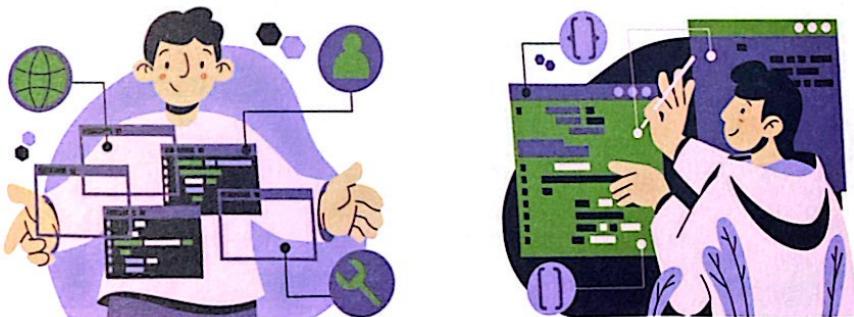
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Trang Responsive</title>
    <link rel="stylesheet" href="styles.css">
</head>

<body>
    <div class="content">
        <h1>Trang Responsive</h1>
        <p>Chào mừng bạn đến với trang web responsive!.</p>
    </div>
</body>

</html>

```



## 1.4.2 | File CSS: Style.CSS

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f0f0f0;
}

.content {
    max-width: 600px;
    margin: 20px auto;
    background-color: white;
    padding: 20px;
    border-radius: 10px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    text-align: center;
}

@media (max-width: 768px) {
    .content {
        padding: 10px;
    }
}

@media (max-width: 480px) {
    .content {
        font-size: 14px;
    }
}

```

## 1.5. Bài Tập 5: Tạo Trang Web Động với CSS và JavaScript

## Đề Bài:

- Tạo một trang web với một hình ảnh: Khi di chuột qua hình ảnh, hình ảnh sẽ thay đổi thành một hình ảnh khác và hiển thị một hiệu ứng animation (ví dụ: zoom).

- Mã nguồn:

## 1.5.1 | File Html: index.html

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Trang Web Động</title>
    <link rel="stylesheet" href="styles.css">
</head>

<body>
    <div class="image-container">
        
    </div>

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

</html>

```

## 1.5.2 | File CSS: Style.CSS

```

body {
    font-family: Arial, sans-serif;
    margin: 0;
}

```

```

padding: 0;
background-color: #f0f0f0;
display: flex;
justify-content: center;
align-items: center;
height: 100vh;
}

.image-container {
  overflow: hidden;
}

#dynamic-image {
  width: 300px;
  height: 200px;
  transition: transform 0.3s ease;
}

#dynamic-image:hover {
  transform: scale(1.2);
}

```

### 1.5.3 File Javascript: script.js

```

const image = document.getElementById('dynamic-image');
image.addEventListener('mouseover', function() {
  image.src = 'image2.jpg';
});

image.addEventListener('mouseout', function() {
  image.src = 'image1.jpg';
});

```

56

## 2. Bài tập tự thực hành về HTML và CSS chuyên sâu

**2.1. Tạo Trang Web Đơn Giản:** Tạo một trang web với tiêu đề, đoạn văn mô tả, danh sách không thứ tự và danh sách có thứ tự.

**2.2. Tạo Trang Web Tự Giới Thiệu:** Thiết kế một trang web giới thiệu về bản thân bạn với ảnh hồ sơ, thông tin liên hệ và sở thích cá nhân.

**2.3. Tạo Trang Blog:** Xây dựng một trang blog với các bài viết, mỗi bài viết bao gồm tiêu đề, hình ảnh, nội dung và ngày đăng.

**2.4. Tạo Trang Danh Mục Sản Phẩm:** Tạo một trang web với tiêu đề, đoạn văn mô tả, danh sách không thứ tự và danh sách có thứ tự.

**2.5. Thiết Kế Giao Diện Web Đơn Giản:** Tạo giao diện web đơn giản với một phần header, một phần content và một phần footer. Tùy chỉnh màu sắc, kích thước và font chữ.

**2.6. Thiết Kế Trang Web Responsive:** Sử dụng media queries để tối ưu hóa trang web cho các thiết bị di động. Đảm bảo rằng giao diện của bạn phản ánh đúng trên cả máy tính và điện thoại di động.

**2.7. Thiết Kế Trang Web Động:** Sử dụng hiệu ứng CSS như hover, transition và animation để làm cho các yếu tố trên trang web phản ánh hiệu ứng khi di chuột qua hoặc khi tương tác.

**2.8. Thiết Kế Trang Web Thẻ Thảo Luận (Card-based Layout):** Thiết kế trang web sử dụng giao diện thẻ thảo luận, trong đó mỗi thẻ bao gồm hình ảnh, tiêu đề và nội dung ngắn.

**2.9. Tạo Trang Web Grid-Based:** Sử dụng CSS Grid hoặc Flexbox để xây dựng một trang web với bố cục linh hoạt dựa trên lưới.

**2.10. Thiết Kế Trang Web Animate:** Sử dụng CSS và JavaScript để tạo các hiệu ứng animate phức tạp như slide-in, fade-in, hoặc parallax scrolling.

**2.11. Thiết Kế Trang Web Tùy Chỉnh với CSS Custom Properties:** Sử dụng CSS Custom Properties (biến CSS) để tạo giao diện có thể tùy chỉnh được mà không cần sửa mã nguồn CSS.

57

Những bài tập này cung cấp một loạt các thách thức từ cơ bản đến nâng cao để giúp bạn rèn kỹ năng lập trình web của mình. Hãy thử thực hiện chúng và tùy chỉnh giao diện theo ý tưởng và sở thích cá nhân của bạn để tạo ra trang web độc đáo.

### 3. Những lưu ý để học tốt và làm việc tốt HTML và CSS

Học tốt và làm việc hiệu quả với HTML và CSS khi xây dựng trang web đòi hỏi sự chú ý đến chi tiết và sự kiên nhẫn. Dưới đây là một số lưu ý giúp bạn học tốt và làm việc tốt với HTML và CSS khi xây dựng trang web:

#### 3.1. Học

- ▶ **Hiểu Rõ HTML và CSS:** Hiểu sâu về cấu trúc và cách hoạt động của HTML và CSS. Học về các thẻ, thuộc tính, và các lựa chọn CSS.
- ▶ **Responsive Web Design:** Học cách thiết kế trang web sao cho nó phản ánh đúng trên các thiết bị và kích thước màn hình khác nhau.
- ▶ **CSS Layouts:** Hiểu về các kiểu layout như float, flexbox, và grid. Biết cách sử dụng chúng để thiết kế các cấu trúc trang web phức tạp.
- ▶ **CSS Selectors và Pseudo-classes:** Hiểu về các lựa chọn CSS như class selectors, ID selectors, và pseudo-classes như :hover, :active, và :nth-child.
- ▶ **CSS Box Model:** Hiểu rõ về box model và làm thế nào các thuộc tính margin, padding, border, và content ảnh hưởng đến việc hiển thị các phần tử trên trang web.

#### 3.1. Học

- ▶ **Thực Hành Liên Tục:** Xây dựng các dự án nhỏ và thử nghiệm các thuộc tính CSS để thấy rõ cách chúng hoạt động trên trình duyệt.
- ▶ **Thiết Kế Đa Dạng:** Thử thiết kế các loại trang web khác nhau để hiểu cách xử lý các yêu cầu thiết kế đa dạng.
- ▶ **Tham Gia Cộng Đồng:** Tham gia vào các diễn đàn và cộng đồng trực tuyến để trao đổi kinh nghiệm và hỏi đáp với các nhà phát triển khác.
- ▶ **Kiểm Tra Trên Nhiều Trình Duyệt và Thiết Bị:** Kiểm tra trang web của bạn trên nhiều trình duyệt và thiết bị để đảm bảo rằng nó hiển thị đúng cách.

▶ **Chăm Sóc Sự Chi Tiết:** Chú ý đến chi tiết như việc chọn màu sắc, kích thước font, và khoảng cách giữa các phần tử. Sự chú ý đến chi tiết là chìa khóa để có giao diện trang web chuyên nghiệp.

▶ **Sử Dụng Công Cụ Gỡ Lỗi (DevTools):** Học cách sử dụng công cụ gỡ lỗi trong trình duyệt để tìm và sửa lỗi CSS một cách nhanh chóng.

▶ **Thiết Lập Tự Động Hóa (Automation):** Sử dụng các công cụ và kịch bản tự động hóa để tối ưu hóa quá trình xây dựng và kiểm thử trang web.

▶ **Duỗi Sự Linh Hoạt:** Trang web cần phải linh hoạt và dễ dàng mở rộng. Hãy thiết kế và viết mã nguồn sao cho chúng dễ dàng được điều chỉnh và mở rộng trong tương lai.

*Nhớ rằng, sự chú ý đến chi tiết, kiên nhẫn và lòng đam mê là chìa khóa để học tốt và làm việc tốt với HTML và CSS khi xây dựng trang web.*

### CHƯƠNG III

## VÀ DOM MANIPULATION

### CÚ PHÁP CƠ BẢN CỦA JAVASCRIPT

JavaScript là một ngôn ngữ lập trình đa năng được sử dụng chủ yếu trong phát triển web. Dưới đây là một số cú pháp cơ bản và chi tiết của JavaScript các bạn cần học và hiểu nó để có thể tuỳ biến khi phát triển ứng dụng web kết hợp nhiều ngôn ngữ:

| Thẻ HTML | Mô Tả và Sử Dụng   | Cú Pháp JavaScript Liên Quan   |
|----------|--|--|
| <div>    | Hộp chứa cho các phần tử, thường được sử dụng để tạo các phần tử giao diện.  | document.createElement('div')<br>element.classList.add('class-name')<br>parentElement.appendChild(element) |
| <input>  | Trường nhập liệu (text, password, checkbox, v.v.)                            | document.createElement('input')<br>element.setAttribute('type', 'text')                                    |
| <button> | Nút hoặc nút nhấn, thường được sử dụng để gọi hàm JavaScript.                | document.createElement('button')<br>element.addEventListener('click', functionName)                        |
| <form>   | Mẫu (form) để gửi dữ liệu, thường được sử dụng để xử lý sự kiện form submit. | document.createElement('form')<br>element.addEventListener('submit', functionName)                         |

|                     |  |   |
|---------------------|--|---|
| <select>            | Hộp dropdown để chọn một tùy chọn, thường được sử dụng trong các biểu mẫu. | document.createElement('select')<br>option = document.createElement('option')<br>parentElement.appendChild(option)      |
| <ul>, <ol>, <li>    | Danh sách không thứ tự, có thứ tự, và mỗi mục trong danh sách.             | document.createElement('ul')<br>li = document.createElement('li')<br>parentElement.appendChild(li)                      |
| <table>, <tr>, <td> | Bảng, dòng trong bảng, ô trong bảng.                                       | document.createElement('table')<br>tr = document.createElement('tr')<br>td = document.createElement('td')               |
| <a>                 | Liên kết, thường được sử dụng để chuyển hướng người dùng đến các trang.    | document.createElement('a')<br>element.setAttribute('href', 'link')   |
| <img>               | Hiển thị hình ảnh.   | document.createElement('img')<br>element.setAttribute('src', 'image-url')<br>element.setAttribute('alt', 'description') |

### TRÌNH BIÊN DỊCH SỬ DỤNG ĐỂ LẬP TRÌNH JAVASCRIPT

Có nhiều IDE (Integrated Development Environment) và trình soạn thảo mã nguồn code tốt cho việc lập trình JavaScript. Dưới đây là một số IDE và trình soạn thảo phổ biến dành cho việc lập trình JavaScript:

#### ► Visual Studio Code (VS Code):

Trang Chủ: <https://code.visualstudio.com/>

Tính Năng: • Hỗ trợ syntax highlighting cho JavaScript và các ngôn ngữ khác.  
• IntelliSense: Tự động hoàn thiện mã và gợi ý hàm và biến.  
• Debugging tích hợp và Git integration.  
• Hỗ trợ một loạt các extension để mở rộng tính năng.

► **WebStorm:**

**Trang Chủ:** WebStorm

**Tính Năng:**

- IntelliSense thông minh và code completion.
- Debugger tích hợp.
- Kiểm thử và phân tích mã.
- Hỗ trợ Node.js và Angular, React, Vue.js frameworks.

► **Sublime Text:**

**Trang Chủ:** Sublime Text

**Tính Năng:**

- Giao diện đẹp và tinh tế.
- Tích hợp nhanh và mượt mà.
- Hỗ trợ syntax highlighting cho JavaScript và nhiều ngôn ngữ khác.
- Hỗ trợ nhiều packages mở rộng.

► **Atom:**

**Trang Chủ:** Atom

**Tính Năng:**

- Phát triển bởi GitHub, Atom là một trình soạn thảo mã nguồn mở.
- Hỗ trợ syntax highlighting cho JavaScript và nhiều ngôn ngữ khác.
- Có thể tùy chỉnh qua các packages và themes.
- Hỗ trợ Git integration.

► **Notepad++:**

**Trang Chủ:** Notepad++

**Tính Năng:**

- Giao diện đơn giản và dễ sử dụng.
- Hỗ trợ syntax highlighting cho JavaScript và nhiều ngôn ngữ khác.
- Tích hợp với nhiều công cụ hữu ích.

► **Eclipse IDE for JavaScript and Web Developers:**

**Trang Chủ:** Eclipse

**Tính Năng:**

- Hỗ trợ JavaScript Development Tools (JSDT).
- Debugging, profiling, và code completion cho JavaScript.
- Hỗ trợ HTML, CSS, và XML editing.

## 1. Biến (Variables)

Khai báo biến sử dụng var, let hoặc const.

- **Var:** Đã lỗi thời, không nên sử dụng.
- **Let:** Cho phép thay đổi giá trị của biến.
- **Const:** Khai báo hằng số, giá trị không thay đổi

```
let variableName = value;
const PI = 3.14;
```

## 2. Kiểu Dữ Liệu (Data Types)

- **Number:** Số nguyên hoặc số thập phân.
- **String:** Chuỗi ký tự.
- **Boolean:** Giá trị true hoặc false.
- **Array:** Dãy giá trị.
- **Object:** Đối tượng.



```
let number = 10;
let string = "Hello, World!";
let boolean = true;
let array = [1, 2, 3];
let object = { key: "value" };
```

## 3. Toán Tử (Operators)

- **Arithmetic Operators:** +, -, \*, /, %.
- **Comparison Operators:** ==, !=, >, <, >=, <=.
- **Logical Operators:** && (and), || (or), ! (not).

## 4. Câu Lệnh Điều Khiển (Control Structures)

### ► If...else Statement:

```
if (condition) {
    // Nếu điều kiện đúng, thực hiện lệnh này
} else {
    // Nếu điều kiện sai, thực hiện lệnh này
}
```

### ► For Loop:

```
for (let i = 0; i < 5; i++) {
    // Lặp 5 lần, i bắt đầu từ 0 và tăng lên 1 sau mỗi lần lặp
}
```

### ► While Loop:

```
while (condition) {
    // Lặp khi điều kiện đúng
}
```

## 5. Hàm (Functions)

Khai báo hàm bằng từ khóa function.

```
function functionName(parameters) {
    // Thực hiện các lệnh với tham số được truyền vào
    return result; // Trả về giá trị (nếu cần)
}
```

## 6. Sự Kiện (Events)

JavaScript được sử dụng để xử lý các sự kiện trên trang web, ví dụ như khi người dùng nhấn nút hoặc chuột.

```
const button = document.getElementById("myButton");
button.addEventListener("click", function() {
    // Thực hiện các lệnh khi nút được nhấn
});
```

## 7. Đối Tượng (Objects)

JavaScript là ngôn ngữ hướng đối tượng. Đối tượng là một tập hợp các thuộc tính và phương thức.

```
let person = {
    name: "John",
    age: 30,
    sayHello: function() {
        console.log("Hello, " + this.name + "!");
    }
};
person.sayHello(); // In ra "Hello, John!"
```

Ngôn ngữ này rất mạnh mẽ và có thể thực hiện nhiều chức năng phức tạp khác nhau khi được kết hợp với các thư viện và framework khác như React, Angular, hoặc Node.js.

## THAO TÁC VỚI DOM: SỰ KIỆN, PHẦN TỬ, VÀ CSS

JavaScript có thể được sử dụng để tương tác với DOM (Document Object Model), là biểu diễn của trang web được trình duyệt hiểu được. Dưới đây là các phương pháp thao tác với DOM bao gồm xử lý sự kiện, thao tác với các phần tử và CSS:

### 1. Xử Lý Sự Kiện (Event Handling)

Xử lý sự kiện là cách JavaScript phản ứng khi người dùng tương tác với trang web. Dưới đây là cách gắn kết sự kiện với một phần tử:

▶ HTML:

```
<button id="myButton">Click me</button>
```

▶ JavaScript:

```
const button = document.getElementById("myButton");

button.addEventListener("click", function() {
    // Thực hiện các lệnh khi nút được nhấn
});
```

### 2. Thao Tác Với Phần Tử (DOM Manipulation)

Có thể thêm, xoá và sửa đổi các phần tử trong DOM bằng JavaScript.

▶ Thêm Phần Tử:

### Javascript:

```
const newElement = document.createElement("div"); // Tạo phần tử mới
newElement.textContent = "Hello, World!"; // Đặt nội dung cho phần tử
document.body.appendChild(newElement); // Thêm phần tử vào cuối body
```

▶ Xoá Phần Tử:

```
const elementToRemove = document.getElementById("myElement");
elementToRemove.parentNode.removeChild(elementToRemove); // Xoá phần tử
```

### Sửa Đổi Thuộc Tính và Nội Dung:

```
const element = document.getElementById("myElement");
element.textContent = "New content"; // Sửa đổi nội dung
element.setAttribute("class", "newClass"); // Sửa đổi thuộc tính (ví dụ: class)
```

### 3. CSS và Lớp (CSS and Classes)

JavaScript cũng có thể thay đổi CSS và lớp của các phần tử.

▶ Thay Đổi CSS:

```
const element = document.getElementById("myElement");
element.style.color = "red"; // Đổi màu chữ thành đỏ
element.style.backgroundColor = "yellow"; // Đổi màu nền thành vàng
```

### ► Thao Tác Với Lớp:

```
const element = document.getElementById("myElement");
element.classList.add("newClass"); // Thêm một lớp mới
element.classList.remove("oldClass"); // Xoá một lớp
element.classList.toggle("active"); // Nếu có lớp "active", xoá; nếu không có, thêm vào
element.classList.contains("checkClass"); // Kiểm tra xem phần tử có lớp "checkClass" không
```

Lưu ý rằng việc sử dụng JavaScript để thay đổi CSS trực tiếp trong DOM có thể gây ra hiệu suất kém nếu được sử dụng quá mức. Trong các ứng dụng lớn, thường tốt hơn là thay đổi các lớp và để CSS xử lý việc hiển thị.



## XỬ LÝ SỰ KIỆN VÀ HIỆU ỨNG

Xử lý sự kiện và hiệu ứng là một phần quan trọng trong việc làm cho trang web trở nên tương tác và thú vị. Dưới đây là cách xử lý sự kiện và thêm hiệu ứng bằng JavaScript:

### 1. Xử Lý Sự Kiện (Event Handling)

#### 1.1. Sử dụng addEventListener:

```
const button = document.getElementById("myButton");

button.addEventListener("click", function() {
    // Thực hiện các lệnh khi nút được nhấn
});
```

### 1.2. Xử lý sự kiện trong HTML (Không ưu tiên sử dụng):

```
<button id="myButton" onclick="myFunction()">Click me</button>

<script>
function myFunction() {
    // Thực hiện các lệnh khi nút được nhấn
}
</script>
```

### 2. Hiệu Ứng (Effects):

#### 2.1. Sử dụng setTimeout để Tạo Độ Trễ

```
// Hiển thị thông báo sau 3 giây
setTimeout(function() {
    alert("Chào mừng bạn đến với trang web của tôi!");
}, 3000); // 3000 ms = 3 giây
```

#### 2.2. Thay Đổi Thuộc Tính CSS (CSS Properties)

```
const element = document.getElementById("myElement");

element.style.backgroundColor = "yellow";
element.style.color = "red";
element.style.fontSize = "20px";
```

### 2.3. Sử Dụng Thư Viện Hiệu Ứng (Ví dụ: jQuery)

Sử dụng **jQuery** để dễ dàng thêm hiệu ứng và tương tác vào trang web. Đầu tiên, bạn cần bao gồm thư viện **jQuery** vào trang web của bạn:

▶ **Html:**

```
<script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
```

Sau đó, bạn có thể sử dụng các phương thức **jQuery** để thêm hiệu ứng:

```
$(document).ready(function() {
    $("#myElement").hide(); //Ẩn phần tử
    $("#myElement").fadeIn(1000); //Xuất hiện phần tử trong vòng 1 giây
    $("#myElement").slideUp(); //Trượt phần tử lên trên
    $("#myElement").slideDown(); //Trượt phần tử xuống dưới
});
```

### 3. Thư Viện Hiệu Ứng (Library for Effects)

Ngoài **jQuery**, có nhiều thư viện khác như **GreenSock Animation Platform (GSAP)**, **anime.js**, và **Velocity.js** để tạo hiệu ứng phức tạp hơn.



▶ **Html:**

```
<script src="https://cdnjs.cloudflare.com/ajax/libs/gsap/3.9.1/gsap.min.js"></script>
```

▶ **Javascript:**

```
// Ví dụ sử dụng GSAP
gsap.to("#myElement", { duration: 2, x: 100, rotation: 360, backgroundColor: "red" });
```

Xử lý sự kiện và hiệu ứng trong **JavaScript** cho phép bạn tạo ra trải nghiệm người dùng động và hấp dẫn trên trang web của bạn.

### 4. JavaScript nâng cao

**JavaScript** nâng cao cung cấp nhiều tính năng mạnh mẽ như đối tượng, hàm, sự kiện, Promise, async/await và nhiều chủ đề phức tạp khác. Bên dưới đây là một số chủ đề **JavaScript** nâng cao chi tiết kèm ví dụ cụ thể:

#### 4.1. Hàm (Functions) và Đối Tượng (Objects)

▶ **Hàm Arrow (Arrow Functions):**

```
const add = (a, b) => a + b;
console.log(add(2, 3)); // Output: 5
```

▶ **Phương Thức Đối Tượng (Object Methods):**

```
const person = {
    firstName: 'John',
    lastName: 'Doe',
    fullName: function() {
        return `${this.firstName} ${this.lastName}`;
    }
};
console.log(person.fullName()); // Output: John Doe
```

## 4.2. Async/Await và Promise:

### ▶ Promise:

```
const fetchData = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve('Data fetched successfully!');
    }, 2000);
  });
};

fetchData().then(data => {
  console.log(data); // Output: Data fetched successfully!
});
```

### ▶ Async/Await:

```
const fetchData = () => {
  return new Promise((resolve, reject) => {
    setTimeout(() => {
      resolve('Data fetched successfully!');
    }, 2000);
  });
};

const fetchDataAsync = async () => {
  const data = await fetchData();
  console.log(data); // Output: Data fetched successfully!
};

fetchDataAsync();
```

## 4.3 Sự Kiện (Events) và Lập Trình Bất Đồng Bộ (Asynchronous Programming)

### ▶ Sự Kiện Click:

```
javascript
const button = document.querySelector('#myButton');
button.addEventListener('click', () => {
  console.log('Button clicked!');
});
```

### ▶ Lập Trình Bất Đồng Bộ (Callbacks):

```
javascript
function fetchData(callback) {
  setTimeout(() => {
    callback('Data fetched successfully!');
  }, 2000);
}

fetchData(data => {
  console.log(data); // Output: Data fetched successfully!
});
```

## 3.4. Map và Filter:

### ▶ Map:

```
const numbers = [1, 2, 3, 4, 5];
const squaredNumbers = numbers.map(num => num * num);
console.log(squaredNumbers); // Output: [1, 4, 9, 16, 25]
```

## ▶ Filter:

```
const numbers = [1, 2, 3, 4, 5];
const evenNumbers = numbers.filter(num => num % 2 === 0);
console.log(evenNumbers); // Output: [2, 4]
```

**3.5. Closure và Currying:**

## ▶ Closure:

```
function outerFunction(outerParam) {
  return function(innerParam) {
    console.log(outerParam + innerParam);
  };
}

const closureExample = outerFunction(10);
closureExample(5); // Output: 15
```

## ▶ Currying:

```
function multiply(a) {
  return function(b) {
    return a * b;
  };
}

const multiplyByTwo = multiply(2);
console.log(multiplyByTwo(3)); // Output: 6
```

Trên đây ví dụ này chỉ là một số ví dụ cơ bản của các chủ đề JavaScript nâng cao. JavaScript cung cấp rất nhiều tính năng và kỹ thuật mạnh mẽ để tạo ra mã JavaScript linh hoạt và hiệu quả. Học thêm về mỗi chủ đề và thực hành chúng trong các dự án thực tế sẽ giúp bạn trở thành một nhà phát triển JavaScript nâng cao.

**BÀI TẬP THỰC HÀNH JAVASCRIPT****1. Bài Tập Cơ Bản****1.1. Bài Tập: Tính Tổng Hai Số**

▶ Đề Bài: Viết một chương trình JavaScript để tính tổng của hai số nguyên được nhập từ người dùng

Mã nguồn:

```
let num1 = parseInt(prompt("Nhập số thứ nhất: "));
let num2 = parseInt(prompt("Nhập số thứ hai: "));
let sum = num1 + num2;
console.log("Tổng của hai số là: " + sum);
```



## 1.2. Bài Tập: Kiểm Tra Số Nguyên Tố

► Đề Bài: Viết một hàm JavaScript để kiểm tra xem một số có phải là số nguyên tố hay không

Mã nguồn:

```
function isPrime(number) {
    if (number <= 1) return false;
    for (let i = 2; i <= Math.sqrt(number); i++) {
        if (number % i === 0) {
            return false;
        }
    }
    return true;
}

let num = parseInt(prompt("Nhập một số: "));
if (isPrime(num)) {
    console.log(num + " là số nguyên tố.");
} else {
    console.log(num + " không là số nguyên tố.");
}
```

## 2. Bài Tập Trung Bình

### 2.1. Bài Tập: Xoay Chuỗi

► Đề Bài: Viết một hàm JavaScript để xoay một chuỗi theo chiều kim đồng hồ

Mã nguồn:



```
function rotateString(str) {
    return str.slice(-1) + str.slice(0, -1);
}

let inputString = prompt("Nhập chuỗi cần xoay: ");
let rotatedString = rotateString(inputString);
console.log("Chuỗi sau khi xoay: " + rotatedString);
```

## 2.2. Bài Tập: Sắp Xếp Mảng

► Đề Bài: Viết một chương trình JavaScript để sắp xếp một mảng số nguyên theo thứ tự tăng dần.

Mã nguồn:

```
let numbers = [5, 2, 9, 1, 5, 6];
numbers.sort(function(a, b) {
    return a - b;
});
console.log("Mảng sau khi sắp xếp: " + numbers);
```

## 3. Bài Tập Nâng Cao

### 3.1. Bài Tập: Đảo Ngược Chuỗi

► Đề Bài: Viết một hàm JavaScript để đảo ngược một chuỗi

Mã nguồn:

```
function reverseString(str) {
    return str.split("").reverse().join("");
}

let inputString = prompt("Nhập chuỗi cần đảo ngược: ");
let reversedString = reverseString(inputString);
console.log("Chuỗi sau khi đảo ngược: " + reversedString);
```

### 3.2. Bài Tập: Đồ Án Mini - Quản Lý Công Việc

- Đề Bài: Xây dựng một ứng dụng web đơn giản để quản lý công việc (Thêm công việc, Xóa công việc, Đánh dấu công việc đã hoàn thành)

Mã nguồn:

```
<input type="text" id="task" placeholder="Nhập công việc">
<button onclick="addTask()">Thêm công việc</button>
<ul id="taskList"></ul>

<script>
function addTask() {
    let task = document.getElementById("task").value;
    if (task !== "") {
        let taskList = document.getElementById("taskList");
        let listItem = document.createElement("li");
        listItem.textContent = task;
        listItem.onclick = function() {
            this.classList.toggle("completed");
        };
        taskList.appendChild(listItem);
        document.getElementById("task").value = "";
    }
}
</script>
```

Bạn hãy thử giải quyết chúng một cách tự duy tự lập và nếu gặp vấn đề, hãy tham khảo mã nguồn để hiểu rõ hơn.

### 4. Một số bài tập ứng dụng nâng cao

#### 4.1. Bài Tập 1: Ứng Dụng Quản Lý Danh Ba

- Đề Bài: Xây dựng một ứng dụng quản lý danh bạ. Người dùng có thể thêm, sửa đổi, xóa và tìm kiếm danh bạ.

Lời giải:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Quản Lý Danh Bạ</title>
</head>
<body>
    <input type="text" id="name" placeholder="Nhập tên">
    <input type="text" id="phone" placeholder="Nhập số điện thoại">
    <button onclick="addContact()">Thêm Liên Hệ</button>
    <ul id="contactList"></ul>

    <script>
        function addContact() {
            var name = document.getElementById("name").value;
            var phone = document.getElementById("phone").value;
            if (name !== "" && phone !== "") {
                var contactList = document.getElementById("contactList");
                var listItem = document.createElement("li");
                listItem.textContent = "Tên: " + name + ", SĐT: " + phone;
                contactList.appendChild(listItem);
                document.getElementById("name").value = "";
                document.getElementById("phone").value = "";
            }
        }
    </script>
</body>
</html>
```

## 4.2. Bài Tập 2: Ứng Dụng Quản Lý Thẻ Ghi Chú

► Đề Bài: Xây dựng một ứng dụng quản lý thẻ ghi chú. Người dùng có thể thêm, chỉnh sửa, xóa và tìm kiếm các ghi chú

*Giải:*

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Quản Lý Thẻ Ghi Chú</title>
</head>
<body>
    <input type="text" id="note" placeholder="Nhập ghi chú">
    <button onclick="addNote()">Thêm Ghi Chú</button>
    <ul id="noteList"></ul>

    <script>
        function addNote() {
            var noteText = document.getElementById("note").value;
            if (noteText !== "") {
                var noteList = document.getElementById("noteList");
                var listItem = document.createElement("li");
                listItem.textContent = noteText;
                listItem.onclick = function() {
                    var newNote = prompt("Chỉnh sửa ghi chú", this.textContent);
                    if (newNote !== null) {
                        this.textContent = newNote;
                    }
                };
                noteList.appendChild(listItem);
                document.getElementById("note").value = "";
            }
        }
    </script>
</body>
</html>

```

### 4.3. Bài Tập 3: Ứng Dụng Quản Lý Đơn Hàng

► **Đề Bài:** Xây dựng một ứng dụng quản lý đơn hàng. Người dùng có thể thêm sản phẩm, xem danh sách đơn hàng và tính tổng số tiền

*Lời giải:*

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Quản Lý Đơn Hàng</title>
</head>
<body>
    <input type="text" id="product" placeholder="Nhập sản phẩm">
    <input type="number" id="price" placeholder="Nhập giá">
    <button onclick="addProduct()">Thêm Sản Phẩm</button>
    <ul id="orderList"></ul>
    <p id="totalPrice">Tổng tiền: $0</p>

    <script>
        var totalPrice = 0;

        function addProduct() {
            var product = document.getElementById("product").value;
            var price = parseInt(document.getElementById("price").value);
            if (product !== "" && !isNaN(price)) {
                var orderList = document.getElementById("orderList");
                var listItem = document.createElement("li");
                listItem.textContent = product + " - $" + price;
                orderList.appendChild(listItem);
                totalPrice += price;
                document.getElementById("totalPrice").textContent = "Tổng tiền: $" +
totalPrice;
                document.getElementById("product").value = "";
                document.getElementById("price").value = "";
            }
        }
    </script>
</body>
</html>

```

Những bài tập này đều tập trung vào việc xây dựng các ứng dụng thực tế với JavaScript và DOM manipulation. Hãy thử tự giải quyết và nếu gặp vấn đề, bạn có thể tham khảo lời giải để học hỏi thêm.

## 5. Bài tập tự thực hành chuyên sâu

### 5.1. Bài Tập

**Xây Dựng Máy Tính Đơn Giản:** Tạo một ứng dụng máy tính đơn giản với các số và các phép toán cơ bản như cộng, trừ, nhân, chia.

**Kiểm Tra Số Nguyên Tố:** Viết một hàm để kiểm tra xem một số có phải là số nguyên tố hay không.

**Tạo Danh Sách:** Cho một mảng các mục, sử dụng JavaScript để thêm chúng vào một danh sách (ul hoặc ol) trên trang web.

**Thay Đổi Màu Nền:** Cho một danh sách các mục, khi người dùng di chuột qua một mục, thay đổi màu nền của mục đó.

**Xây Dựng Ứng Dụng Đếm Ngược:** Xây dựng một ứng dụng đếm ngược từ một số cho trước đến 0, cập nhật giá trị trên trang mỗi giây.

**Tìm Kiếm Thông Tin:** Xây dựng một ô tìm kiếm, khi người dùng nhập từ khóa hiển thị các kết quả phù hợp từ một danh sách dữ liệu.

**Tạo Ứng Dụng Đánh Giá (Rating App):** Xây dựng một ứng dụng đánh giá với các sao (star ratings). Khi người dùng di chuột qua hoặc nhấp vào một số sao, hiển thị số sao đã chọn.

**Tạo Trang Chuyển Động (Scrolling Page):** Tạo một trang web có thể cuộn (scrollable) với các phần tử xuất hiện và biến mất khi người dùng cuộn trang.

**Tạo Ứng Dụng Thư Chat Tiện Lợi:** Xây dựng một ứng dụng thư chat đơn giản sử dụng WebSockets hoặc XMLHttpRequest để gửi và nhận tin nhắn.

**Tạo Trình Đơn (Dropdown Menu):** Tạo một trình đơn rơi xuống (dropdown menu) với các mục và các mục con. Khi người dùng di chuột qua một mục, hiển thị các mục con.

**Tạo Trò Chơi Bắn Súng (Shooting Game):** Xây dựng một trò chơi bắn súng đơn giản sử dụng DOM Manipulation và các sự kiện như nhấp chuột và phím bàn phím.

Những bài tập này sẽ giúp bạn rèn luyện kỹ năng lập trình JavaScript và DOM Manipulation từ cơ bản đến nâng cao và chuyên sâu. Hãy thử thực hiện chúng và tùy chỉnh chức năng và giao diện theo ý tưởng của bạn để tạo ra các ứng dụng web đa dạng.

### 5.2. Khi sử dụng ngôn ngữ lập trình JavaScript, có một số lưu ý quan trọng để bạn nên xem xét

**Kiến Thức Cơ Bản:** Đảm bảo bạn hiểu rõ kiến thức cơ bản về JavaScript, bao gồm cú pháp, biến, kiểu dữ liệu, cấu trúc điều kiện, vòng lặp và hàm.

**Xử Lý Sự Kiện:** JavaScript thường được sử dụng để xử lý sự kiện trên trang web. Hãy hiểu cách thêm và xử lý các sự kiện như nhấn nút, nhấn chuột và gửi dữ liệu.

**DOM Manipulation:** DOM (Document Object Model) cho phép bạn tương tác với phần tử HTML. Hiểu cách lấy phần tử, thay đổi nội dung và thuộc tính, và tạo hoặc xóa phần tử là quan trọng.

**An Toàn Bảo Mật:** Tránh việc tin tưởng dữ liệu từ người dùng một cách mù quáng. Luôn kiểm tra và xác thực dữ liệu trước khi xử lý nó để tránh các tấn công bảo mật.

**Quản Lý Bộ Nhớ:** JavaScript không có thu gom rác tự động, vì vậy bạn cần phải quản lý bộ nhớ một cách cẩn thận. Đảm bảo giải phóng bộ nhớ khi không cần thiết để tránh rò rỉ bộ nhớ.

**Hiệu Năng:** Hiểu cách tối ưu hóa mã JavaScript để tăng hiệu năng ứng dụng. Tránh việc sử dụng vòng lặp lồng nhau quá nhiều hoặc thực hiện quá nhiều thay đổi DOM.

**Thử Nghiệm và Gỡ Lỗi:** Sử dụng công cụ gỡ lỗi trong trình duyệt (ví dụ: DevTools) để tìm lỗi trong mã JavaScript của bạn. Thực hiện thử nghiệm đầy đủ trước khi triển khai.

**Học Các Thư Viện và Framework:** JavaScript có nhiều thư viện và framework mạnh mẽ như React, Vue.js và Angular. Học cách sử dụng chúng để tạo ứng dụng phức tạp hơn.

**Gữ Mã Dự Án Sạch Sẽ:** Sử dụng các quy tắc lập trình tốt, thực hiện quản lý phiên bản, và tạo mã dự án có cấu trúc để dễ bảo trì và phát triển.

**Học Liên Tục:** JavaScript là một ngôn ngữ phát triển nhanh chóng, vì vậy hãy theo dõi các xu hướng mới và học liên tục để cải thiện kỹ năng lập trình của bạn.

**Học Về Hệ Thống Ecosystem:** JavaScript không chỉ là ngôn ngữ, mà còn liên quan đến hệ sinh thái bao gồm Node.js (cho phía máy chủ), npm (trình quản lý gói), và nhiều công cụ phát triển khác.

**Tích Hợp Testing và Tự Động Hóa:** Hãy học cách viết kiểm thử đơn vị và kiểm thử tích hợp để đảm bảo sự ổn định của ứng dụng của bạn. Sử dụng công cụ tự động hóa để tối ưu hóa quy trình phát triển.

Nhớ rằng JavaScript là một ngôn ngữ linh hoạt và có thể được sử dụng trên nhiều nền tảng, từ phía máy chủ đến phía máy khách. Việc hiểu rõ cách làm việc với JavaScript đúng cách sẽ giúp bạn phát triển ứng dụng hiệu quả và an toàn.

## CHƯƠNG IV

# AJAX VÀ ASYNCHRONOUS PROGRAMMING

## GỬI YÊU CẦU ĐẾN MÁY CHỦ

**Ajax (Asynchronous JavaScript and XML):** Ajax là một công nghệ cho phép trang web gửi và nhận dữ liệu từ máy chủ mà không cần phải tải lại toàn bộ trang web. Nó sử dụng kỹ thuật lập trình bất đồng bộ (asynchronous programming) để thực hiện các yêu cầu đối với máy chủ mà không làm gián đoạn trải nghiệm người dùng trên trang web.

Dưới đây là một số điểm quan trọng về Ajax và lập trình bất đồng bộ:

### 1. Ajax

**XMLHttpRequest Object:** Để gửi yêu cầu Ajax, bạn sử dụng đối tượng XMLHttpRequest trong JavaScript. Đối tượng này cho phép bạn gửi yêu cầu đến máy chủ và nhận dữ liệu trả về mà không cần tải lại trang web.

**Dữ Liệu Truyền Đì và Nhận Về:** Bạn có thể truyền dữ liệu đến máy chủ thông qua yêu cầu Ajax dưới dạng tham số, và máy chủ sẽ trả về dữ liệu dưới dạng văn bản (thường là JSON hoặc XML) mà bạn có thể xử lý trong mã JavaScript.

**Thực Hiện Yêu Cầu Bất Đồng Bộ:** Ajax cho phép thực hiện các yêu cầu đến máy chủ mà không chờ đợi phản hồi. Điều này giúp tránh tình trạng trang web bị đứng lại khi chờ đợi dữ liệu từ máy chủ.

**Xử Lý Phản Hồi (Response):** Sau khi nhận được phản hồi từ máy chủ, bạn có thể xử lý dữ liệu và cập nhật trang web một cách động dựa trên thông tin nhận được.

## 2. Lập Trình Bất Đồng Bộ

**Callback Functions:** Trong lập trình bất đồng bộ, callback functions được sử dụng để xử lý kết quả của các yêu cầu bất đồng bộ. Callbacks là hàm được truyền vào và được gọi khi yêu cầu hoặc tác vụ bất đồng bộ hoàn thành.

**Promises và Async/Await:** Promises là một cách để xử lý các tác vụ bất đồng bộ trong JavaScript. Async/Await là một cú pháp ngôn ngữ giúp viết mã bất đồng bộ một cách đơn giản và dễ đọc hơn, sử dụng Promises.

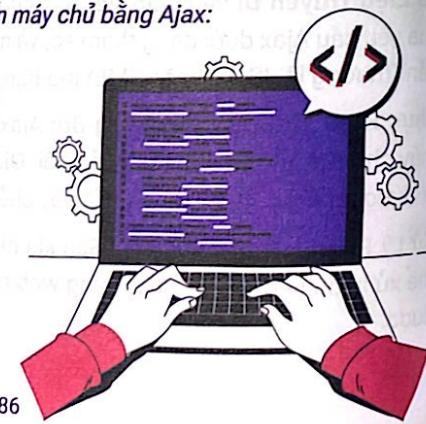
**Xử Lý Lỗi:** Trong lập trình bất đồng bộ, xử lý lỗi đặc biệt quan trọng. Sử dụng try-catch blocks (hoặc .catch()) với Promises, hoặc try-catch với Async/Await để xử lý lỗi một cách an toàn.

**Tránh Callback Hell:** Khi sử dụng callback functions lồng nhau quá nhiều, có thể dẫn đến callback hell, một cấu trúc mã khó đọc và bảo trì. Để tránh điều này, hãy sử dụng Promises hoặc Async/Await để làm cho mã của bạn gọn gàng hơn.

Lập trình bất đồng bộ và sử dụng Ajax đòi hỏi hiểu biết sâu rộng về JavaScript và cách xử lý các tác vụ không đồng bộ. Học và thực hành các kỹ thuật này sẽ giúp bạn xây dựng các ứng dụng web tương tác và linh hoạt hơn.

Để gửi yêu cầu đến máy chủ từ một trang web, bạn có thể sử dụng Ajax trong JavaScript. Dưới đây là một ví dụ về cách gửi yêu cầu GET và POST đến máy chủ bằng Ajax:

### ▶ Gửi Yêu Cầu GET Đến Máy Chủ



```
var xhr = new XMLHttpRequest();
xhr.open("GET", "url_may_chu", true);
xhr.onreadystatechange = function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
        var response = xhr.responseText;
        // Xử lý dữ liệu nhận được từ máy chủ ở đây
        console.log(response);
    }
};
xhr.send();
```

### ▶ Gửi Yêu Cầu POST Đến Máy Chủ

```
var xhr = new XMLHttpRequest();
xhr.open("POST", "url_may_chu", true);
xhr.setRequestHeader("Content-Type", "application/x-www-form-urlencoded");
xhr.onreadystatechange = function () {
    if (xhr.readyState == 4 && xhr.status == 200) {
        var response = xhr.responseText;
        // Xử lý dữ liệu nhận được từ máy chủ ở đây
        console.log(response);
    }
};
var data = "key1=value1&key2=value2"; // Dữ liệu bạn muốn gửi đi,
// định dạng chuỗi query
xhr.send(data);
```

### ▶ Giải thích trong cả hai ví dụ trên

`xhr.open(method, url, async)`: Phương thức open mở kết nối đến máy chủ. method có thể là "GET" hoặc "POST". url là địa chỉ máy chủ bạn muốn gửi yêu cầu đến. async là một giá trị boolean, đặt là true nếu bạn muốn thực hiện yêu cầu bất đồng bộ.

**xhr.setRequestHeader(header, value):** Phương thức này đặt các header cho yêu cầu HTTP. Trong ví dụ POST, chúng ta đặt header "Content-Type" thành "application/x-www-form-urlencoded" để chỉ định dữ liệu sẽ được gửi dưới dạng chuỗi query.

**xhr.onreadystatechange:** Sự kiện này được gọi mỗi khi trạng thái của đối tượng XMLHttpRequest thay đổi. Trong hàm xử lý sự kiện này, chúng ta kiểm tra nếu trạng thái là 4 (yêu cầu đã hoàn thành) và mã trạng thái là 200 (OK), sau đó chúng ta xử lý dữ liệu nhận được từ máy chủ.

**xhr.send(data):** Phương thức này gửi yêu cầu đến máy chủ. Trong trường hợp POST, bạn có thể truyền dữ liệu bạn muốn gửi đi thông qua tham số data.

Lưu ý rằng việc sử dụng XMLHttpRequest trực tiếp đã lỗi thời và không được khuyến khích trong các ứng dụng web hiện đại. Thay vào đó, người ta thường sử dụng các thư viện như fetch hoặc framework như Axios để gửi yêu cầu đến máy chủ một cách hiện đại và dễ đọc hơn.

## XỬ LÝ DỮ LIỆU JSON VÀ API

Xử lý dữ liệu JSON và làm việc với API (Application Programming Interface) là một phần quan trọng trong lập trình web hiện đại. Dưới đây là cách bạn có thể xử lý dữ liệu JSON và tương tác với các API bằng JavaScript:

### 1. Xử Lý Dữ Liệu JSON

#### 1.1. Parse JSON (Chuyển đổi JSON thành Object)

```
var jsonString = '{"name": "John", "age": 30, "city": "New York"}';
var jsonObject = JSON.parse(jsonString);
console.log(jsonObject.name); // Output: John
```

### 1.2. Stringify Object thành JSON (Chuyển đổi Object thành JSON)

```
var obj = { name: "John", age: 30, city: "New York" };
var jsonString = JSON.stringify(obj);
console.log(jsonString); // Output: '{"name": "John", "age": 30, "city": "New York"}'
```

## 2. Làm Việc với API

### 2.1. Sử dụng Fetch API để Gửi Yêu Cầu GET

```
fetch('https://api.example.com/data')
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

### 2.2. Sử dụng Fetch API để Gửi Yêu Cầu POST

```
var data = {
  name: 'John',
  age: 30
};

fetch('https://api.example.com/save', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json'
  },
  body: JSON.stringify(data)
})
  .then(response => response.json())
  .then(data => console.log(data))
  .catch(error => console.error('Error:', error));
```

### 2.3. Sử dụng Axios để Gửi Yêu Cầu GET và POST

- ▶ **Cài Đặt Axios:** npm install axios
- ▶ **Sử Dụng Axios**

```
const axios = require('axios');

// Gửi Yêu Cầu GET
axios.get('https://api.example.com/data')
  .then(response => console.log(response.data))
  .catch(error => console.error('Error:', error));

// Gửi Yêu Cầu POST
var data = {
  name: 'John',
  age: 30
};

axios.post('https://api.example.com/save', data)
  .then(response => console.log(response.data))
  .catch(error => console.error('Error:', error));
```

Khi làm việc với API, hãy nhớ kiểm tra tài liệu của API để biết các yêu cầu và định dạng dữ liệu chính xác. Các thư viện như Axios giúp quản lý các yêu cầu API một cách thuận tiện hơn và cung cấp các tính năng bảo mật và xử lý lỗi tốt hơn so với việc sử dụng Fetch API trực tiếp.

## CẢI THIỆN TRẢI NGHIỆM NGƯỜI DÙNG VỚI AJAX

Cải thiện trải nghiệm người dùng với Ajax để cập đến việc sử dụng công nghệ Ajax (Asynchronous JavaScript and XML) để tối ưu hóa trang web và cung cấp trải nghiệm người dùng mượt mà và không gián đoạn. Dưới đây là một số cách bạn có thể cải thiện trải nghiệm người dùng bằng cách sử dụng Ajax:

### 1. Tải Dữ Liệu Một Cách Bất Đồng Bộ

Sử dụng Ajax để tải dữ liệu từ máy chủ mà không cần tải lại toàn bộ trang web. Điều này giúp trang web tải nhanh hơn và người dùng không cảm thấy gián đoạn.

### 2. Gửi Dữ Liệu Mà Không Tải Lại Trang

Sử dụng Ajax để gửi dữ liệu mà không cần chuyển hướng hoặc tải lại trang. Điều này thích hợp cho các biểu mẫu và tương tác người dùng mà không làm gián đoạn trải nghiệm của họ.

### 3. Tìm Kiếm Tự Động và Gợi Ý

Sử dụng Ajax để tìm kiếm tự động và hiển thị gợi ý ngay khi người dùng nhập thông tin vào ô tìm kiếm. Điều này giúp người dùng tiết kiệm thời gian và nâng cao trải nghiệm tìm kiếm của họ.

### 4. Phân Trang Linh Hoạt

Khi người dùng chuyển đến một trang mới, sử dụng Ajax để tải nội dung của trang đó mà không cần tải lại toàn bộ trang web. Điều này tạo ra trải nghiệm người dùng mượt mà hơn.

### 5. Xử Lý Lỗi Một Cách Thông Minh

Sử dụng Ajax để xử lý lỗi một cách thông minh và hiển thị thông báo lỗi mà không gián đoạn trang web. Thông báo lỗi nên được hiển thị một cách thân thiện và dễ hiểu cho người dùng.



## 6. Nạp Dữ Liệu Động

Sử dụng Ajax để nạp dữ liệu động khi người dùng cuộn trang hoặc thực hiện các tương tác khác. Điều này giúp giữ cho trang web nhanh chóng và linh hoạt mà không cần tải toàn bộ trang lại.

## 7. Thay Đổi Giao Diện Mà Không Tải Lại Trang

Sử dụng Ajax để thay đổi nội dung hoặc giao diện trang web mà không cần tải lại trang. Điều này giúp cải thiện trải nghiệm người dùng khi họ tương tác với trang web mà không phải chờ đợi. Sử dụng Ajax một cách thông minh và hiệu quả có thể tối ưu hóa trải nghiệm người dùng, giúp trang web của bạn trở nên linh hoạt, nhanh chóng và dễ sử dụng.

*Dưới đây là một ví dụ chi tiết về cách sử dụng Ajax để cải thiện trải nghiệm người dùng. Trong ví dụ này, chúng ta sẽ sử dụng HTML, JavaScript, và một API giả mạo để lấy dữ liệu và hiển thị nó trên trang web mà không cần tải lại trang.*

### ▶ HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajax Example</title>
</head>

<body>
    <div id="data-container"></div>

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

</html>
```

### ▶ JavaScript (script.js)

```
document.addEventListener("DOMContentLoaded", function () {
    var dataContainer = document.getElementById("data-container");

    // Sử dụng fetch API để gửi yêu cầu và lấy dữ liệu từ API giả mạo
    fetch("https://jsonplaceholder.typicode.com/todos")
        .then(function (response) {
            if (!response.ok) {
                throw new Error("Network response was not ok");
            }
            return response.json();
        })
        .then(function (data) {
            // Xử lý dữ liệu nhận được từ API
            data.forEach(function (item) {
                var todoItem = document.createElement("div");
                todoItem.textContent = "ID: " + item.id + " - " + item.title;
                dataContainer.appendChild(todoItem);
            });
        })
        .catch(function (error) {
            console.error("There has been a problem with your fetch operation:", error);
        });
});
```

Trong ví dụ này:

- HTML file chứa một `<div>` với `id="data-container"` để hiển thị dữ liệu từ API.
- Trong file JavaScript (script.js), chúng ta sử dụng `fetch API` để gửi yêu cầu `GET` đến một API giả mạo (<https://jsonplaceholder.typicode.com/todos>).
- Khi dữ liệu được nhận từ API, chúng ta xử lý và hiển thị nó trong phần tử có `id="data-container"` trên trang web mà không cần tải lại toàn bộ trang.

» **Lưu ý:** Vui lòng chạy mã này trên máy chủ hoặc môi trường phát triển có hỗ trợ CORS (Cross-Origin Resource Sharing), vì trình duyệt sẽ từ chối yêu cầu Ajax đến các domain khác nếu không có sự cho phép.

## 8. Bài Tập Code

### 8.1. Hiển Thị Dữ Liệu từ API

- Sử dụng fetch API để lấy danh sách người dùng từ một API.
- Hiển thị thông tin người dùng trên trang web mà không cần tải lại trang.

```
document.addEventListener("DOMContentLoaded", function () {
    var userList = document.getElementById("user-list");

    fetch("https://jsonplaceholder.typicode.com/users")
        .then(function (response) {
            return response.json();
        })
        .then(function (data) {
            data.forEach(function (user) {
                var listItem = document.createElement("li");
                listItem.textContent = user.name;
                userList.appendChild(listItem);
            });
        })
        .catch(function (error) {
            console.error("Error:", error);
        });
});
```

### 8.2. Gửi Dữ Liệu Đến Máy Chủ

- Tạo một biểu mẫu HTML với các trường nhập thông tin (ví dụ: tên, email).
- Sử dụng Ajax để gửi dữ liệu từ biểu mẫu đến máy chủ khi người dùng nhấn nút "Gửi".
- Hiển thị thông báo khi dữ liệu đã được gửi thành công.

## ▶ HTML

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Ajax Form Example</title>
</head>

<body>
    <form id="user-form">
        <label for="name">Name:</label>
        <input type="text" id="name" name="name" required><br><br>
        <label for="email">Email:</label>
        <input type="email" id="email" name="email" required><br><br>
        <button type="submit">Submit</button>
    </form>

    <div id="message"></div>

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

</html>
```



## ▶ Javascript

```

document.addEventListener("DOMContentLoaded", function () {
    var userForm = document.getElementById("user-form");
    var messageDiv = document.getElementById("message");

    userForm.addEventListener("submit", function (event) {
        event.preventDefault();

        var formData = new FormData(userForm);

        fetch("https://jsonplaceholder.typicode.com/posts", {
            method: "POST",
            body: formData
        })
        .then(function (response) {
            if (response.ok) {
                return response.json();
            }
            throw new Error("Network response was not ok");
        })
        .then(function (data) {
            messageDiv.textContent = "Data sent successfully. Post
ID: " + data.id;
        })
        .catch(function (error) {
            console.error("Error:", error);
        });
    });
});

```

Trong ví dụ này:

- **Hiển Thị Dữ Liệu từ API:** Dùng fetch API để lấy danh sách người dùng từ API và hiển thị tên người dùng trên trang web.
- **Gửi Dữ Liệu Đến Máy Chủ:** Sử dụng Ajax để gửi dữ liệu từ biểu mẫu đến máy chủ khi người dùng nhấn nút "Gửi". Khi dữ liệu được gửi thành công, hiển thị thông báo với ID của bài viết mới.

## 9. Bài tập tự thực hành chuyên sâu

**Tải Dữ Liệu Từ API:** Sử dụng XMLHttpRequest hoặc Fetch API để tải dữ liệu từ một API công cộng (ví dụ: JSONPlaceholder) và hiển thị dữ liệu lên trang web.

**Gửi Dữ Liệu Đến Máy Chủ:** Tạo một biểu mẫu (form) thu thập thông tin từ người dùng. Sử dụng Ajax để gửi dữ liệu từ biểu mẫu đến máy chủ và hiển thị phản hồi trên trang mà không cần tải lại toàn bộ trang.

**Autocomplete Suggest Box:** Tạo một ô tìm kiếm với chức năng autocomplete. Sử dụng Ajax để gửi yêu cầu khi người dùng nhập và hiển thị các kết quả gợi ý dựa trên dữ liệu từ máy chủ.

**Infinity Scroll:** Tạo một danh sách dài của các mục. Sử dụng Ajax để tải dữ liệu từ máy chủ khi người dùng cuộn xuống cuối danh sách, giúp triển khai chức năng infinity scroll.

**Parallel Requests:** Sử dụng Promise hoặc Async/Await để thực hiện nhiều yêu cầu Ajax đồng thời (parallel requests) đến các API khác nhau. Xem cách xử lý kết quả khi tất cả các yêu cầu đã hoàn thành.

**Thực Hiện Thao Tác CRUD Đồng Thời:** Tạo một trang quản lý sản phẩm hoặc danh bạ. Sử dụng Ajax để thực hiện các thao tác CRUD (tạo, đọc, cập nhật, xoá) đồng thời trên nhiều mục.

**Real-time Chat App:** Xây dựng ứng dụng chat thời gian thực sử dụng WebSockets để truyền dữ liệu. Sử dụng Ajax để gửi và nhận tin nhắn văn bản giữa các người dùng.

**Gửi và Nhận Dữ Liệu Lớn:** Tạo một chức năng tải và tải lên file lớn. Sử dụng Ajax để chia nhỏ dữ liệu thành các chunks nhỏ và gửi chúng đến máy chủ. Hiển thị tiến độ tải lên trên giao diện người dùng.

Những bài tập này sẽ giúp bạn phát triển kỹ năng xử lý yêu cầu không đồng bộ trong ứng dụng web, từ các tác vụ cơ bản đến những chức năng phức tạp và thời gian thực. Hãy thử thực hiện chúng và tùy chỉnh chức năng và giao diện theo ý tưởng của bạn để tạo ra các ứng dụng web độc đáo và mạnh mẽ.



## CƠ SỞ DỮ LIỆU VÀ SQL, MYSQL TRONG LẬP TRÌNH WEB

### NGUYÊN LÝ CƠ BẢN CỦA CƠ SỞ DỮ LIỆU

**Nguyên lý cơ bản của cơ sở dữ liệu (Database Principles):** là các nguyên tắc và quy tắc căn bản trong thiết kế, quản lý và sử dụng cơ sở dữ liệu. Dưới đây là một số nguyên lý quan trọng.

**Nguyên Tắc ACID:** Nguyên tắc ACID (Atomicity, Consistency, Isolation, Durability) là một tập hợp các nguyên tắc đảm bảo tính toàn vẹn và nhất quán của dữ liệu trong cơ sở dữ liệu. Điều này đảm bảo rằng các giao dịch (transactions) sẽ được xử lý một cách an toàn và đáng tin cậy.

**Nguyên Tắc Đẻ Dữ Liệu Không Trùng Lặp (Normalization):** Nguyên tắc này để cập đến việc thiết kế cơ sở dữ liệu sao cho dữ liệu không bị trùng lặp. Điều này giúp giảm thiểu lãng phí lưu trữ, cải thiện hiệu suất và đảm bảo tính nhất quán của dữ liệu.

**Nguyên Tắc Thiết Kế Mô Hình Dữ Liệu (Data Modeling):** Để hiểu được cách dữ liệu được lưu trữ và tương tác trong cơ sở dữ liệu, nguyên tắc này đề xuất cách tạo mô hình dữ liệu. Mô hình này sẽ mô tả cách các đối tượng, thực thể và quan hệ giữa chúng được biểu diễn trong cơ sở dữ liệu.

**Nguyên Tắc Bảo Mật Dữ Liệu:** Bảo mật dữ liệu là quan trọng để đảm bảo rằng chỉ những người được phép có quyền truy cập và sửa đổi dữ liệu trong cơ sở dữ liệu.



Các nguyên tắc bảo mật bao gồm kiểm tra danh tính, quản lý quyền truy cập và mã hóa dữ liệu.

**Nguyên Tắc Sao Lưu và Khôi Phục (Backup and Recovery):** Để đảm bảo dữ liệu an toàn, nguyên tắc này đề xuất cách thực hiện sao lưu định kỳ của cơ sở dữ liệu và cách khôi phục dữ liệu trong trường hợp dữ liệu bị mất hoặc hỏng.

**Nguyên Tắc Hiệu Suất và Tối Ưu Hóa (Performance and Optimization):** Cải thiện hiệu suất của cơ sở dữ liệu là quan trọng. Nguyên tắc này đề xuất cách tối ưu hóa truy vấn cơ sở dữ liệu, sử dụng chỉ mục (indexing), và quản lý tài nguyên cơ sở dữ liệu một cách hiệu quả.

**Nguyên Tắc Bảo Quản Lịch Sử Dữ Liệu (Data History Preservation):** Trong một số trường hợp, việc bảo quản lịch sử dữ liệu (như thay đổi dữ liệu theo thời gian) có ý nghĩa quan trọng. Nguyên tắc này đề xuất cách lưu trữ và truy vấn dữ liệu lịch sử.

**Nguyên Tắc Liên Kết Dữ Liệu (Data Linking):** Khi cần kết nối dữ liệu từ nhiều nguồn hoặc cơ sở dữ liệu khác nhau, nguyên tắc này đề xuất cách thực hiện liên kết dữ liệu để có cái nhìn toàn diện hơn về thông tin.

Các nguyên tắc này là quan trọng trong quản lý cơ sở dữ liệu và thiết kế hệ thống thông tin hiệu quả. Nắm vững chúng giúp đảm bảo tính toàn vẹn, bảo mật và hiệu suất của cơ sở dữ liệu.



## 1. Hệ Quản Trị Cơ Sở Dữ Liệu Sử Dụng SQL

Hệ quản trị cơ sở dữ liệu (DBMS - Database Management System) sử dụng SQL (Structured Query Language) cho việc tương tác với cơ sở dữ liệu. Dưới đây là một số hệ quản trị cơ sở dữ liệu phổ biến sử dụng SQL:



### 1.1. MySQL

**Trang Chủ:** MySQL

**Link tải:** <https://www.mysql.com/>

#### Tính Năng:

- Một hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến.
- Hỗ trợ các tính năng ACID (Atomicity, Consistency, Isolation, Durability).
- Dễ dàng tích hợp với các ngôn ngữ lập trình như PHP, Python, và Node.js.

### 1.2. PostgreSQL

**Trang Chủ:** PostgreSQL

**Link tải:** <https://www.postgresql.org/>

#### Tính Năng:

- Hệ quản trị cơ sở dữ liệu mã nguồn mở với các tính năng mạnh mẽ và tiên tiến.
- Hỗ trợ JSON và các kiểu dữ liệu phức tạp.
- Có các tính năng bảo mật nâng cao.

## 1.3. Microsoft SQL Server

**Trang Chủ:** SQL Server

**Link tải:** <https://www.microsoft.com/en-us/sql-server/>

#### Tính Năng:

- Hệ quản trị cơ sở dữ liệu của Microsoft với các phiên bản dành cho các nền tảng khác nhau.
- Hỗ trợ tính năng Business Intelligence và tích hợp tốt với các sản phẩm Microsoft khác.

### 1.4. Oracle Database

**Trang Chủ:** Oracle Database

**Link tải:** <https://www.oracle.com/database/>

#### Tính Năng:

- Một trong những hệ quản trị cơ sở dữ liệu thương mại phổ biến nhất thế giới.
- Hỗ trợ các tính năng cao cấp như Partitioning, Real Application Clusters, và Data Guard.

### 1.5. SQLite

**Trang Chủ:** SQLite

**Link tải:** <https://www.sqlite.org/index.html>

#### Tính Năng:

- Là một hệ quản trị cơ sở dữ liệu nhẹ, không cần máy chủ riêng biệt.
- Dễ dàng tích hợp vào các ứng dụng độc lập và thiết bị di động.

## 1.6. MariaDB

**Trang Chủ:** MariaDB

Link tải: <https://mariadb.org/>

**Tính Năng:**

- Fork của MySQL sau khi được mua lại bởi Oracle.
- Hỗ trợ các tính năng tương tự như MySQL với sự mở rộng và cải thiện.

Các hệ quản trị cơ sở dữ liệu này đều sử dụng SQL để tương tác với dữ liệu, như tạo bảng, chèn, cập nhật và truy vấn dữ liệu. Lựa chọn hệ quản trị cơ sở dữ liệu phụ thuộc vào yêu cầu cụ thể của dự án và sở thích của nhà phát triển.

## 2. Hệ Quản Trị Cơ Sở Dữ Liệu Sử Dụng MySQL

MySQL là một hệ quản trị cơ sở dữ liệu mã nguồn mở phổ biến và được sử dụng rộng rãi trên toàn thế giới. Dưới đây là một số hệ quản trị cơ sở dữ liệu (DBMS) phổ biến mà bạn có thể sử dụng MySQL:

### 2.1. MariaDB

**Trang Chủ:** MariaDB

MariaDB là một fork của MySQL, được tạo ra khi Oracle mua lại MySQL. Nó giữ nguyên sức mạnh và linh hoạt của MySQL.

### 2.2. Percona Server for MySQL

**Trang Chủ:** Percona Server for MySQL

Percona Server là một phiên bản được tối ưu hóa và mở rộng của MySQL, được thiết kế để cải thiện hiệu suất và mở rộng.

## 2.3. Amazon Aurora

**Trang Chủ:** Amazon Aurora

Link tải: <https://aws.amazon.com/rds/aurora/>

Amazon Aurora là một dịch vụ quản lý cơ sở dữ liệu được xây dựng trên nền tảng MySQL. Nó được tối ưu hóa cho các môi trường cloud và cung cấp khả năng mở rộng tốt.

## 2.4. Google Cloud SQL

**Trang Chủ:** Google Cloud SQL

Google Cloud SQL là dịch vụ quản lý cơ sở dữ liệu MySQL được cung cấp bởi Google Cloud. Nó tự động quản lý sao lưu, di chuyển và phục hồi dữ liệu.

### 2.5. Microsoft Azure Database for MySQL

**Trang Chủ:**

Azure Database for MySQL

Azure Database for MySQL là dịch vụ quản lý cơ sở dữ liệu MySQL được Microsoft cung cấp trên nền tảng Azure.

### 2.6. DigitalOcean Managed Databases

**Trang Chủ:**

DigitalOcean Managed Databases

DigitalOcean cung cấp dịch vụ quản lý cơ sở dữ liệu MySQL trên các máy chủ của họ, giúp bạn dễ dàng triển khai và quản lý cơ sở dữ liệu.

Tùy thuộc vào yêu cầu và ngân sách của bạn, bạn có thể lựa chọn giữa việc tự triển khai và quản lý MySQL trên máy chủ riêng hoặc sử dụng các dịch vụ quản lý cơ sở dữ liệu trên các nền tảng đám mây.

## NGÔN NGỮ TRUY VẤN SQL

SQL (Structured Query Language) là một ngôn ngữ tiêu chuẩn được sử dụng để truy vấn và tương tác với cơ sở dữ liệu quan hệ. MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở phổ biến, sử dụng SQL để thao tác dữ liệu.

Dưới đây là một số lý thuyết cơ bản và giải thích chi tiết về SQL và MySQL:

| Cấp Độ     | Kiến Thức và Cú Pháp SQL, MySQL  |
|------------|--|
| Cơ Bản     | <ul style="list-style-type: none"> <li>- <b>SELECT</b> (lựa chọn dữ liệu từ cơ sở dữ liệu)</li> <li>- <b>WHERE</b> (điều kiện lựa chọn)</li> </ul>     |
|            | <ul style="list-style-type: none"> <li>- <b>INSERT INTO</b> (chèn dữ liệu vào bảng)</li> <li>- <b>UPDATE</b> (cập nhật dữ liệu trong bảng)</li> </ul>  |
|            | <ul style="list-style-type: none"> <li>- <b>DELETE FROM</b> (xóa dữ liệu từ bảng)</li> </ul>   |
| Nâng Cao   | <ul style="list-style-type: none"> <li>- <b>JOIN</b> (kết hợp dữ liệu từ nhiều bảng)</li> <li>- <b>ORDER BY</b> (sắp xếp kết quả truy vấn)</li> </ul>  |
|            | <ul style="list-style-type: none"> <li>- <b>GROUP BY</b> (nhóm dữ liệu)</li> <li>- <b>HAVING</b> (điều kiện nhóm)</li> </ul>                           |
|            | <ul style="list-style-type: none"> <li>- <b>DISTINCT</b> (lọc bỏ giá trị trùng lặp)</li> </ul>   |
| Chuyên Sâu | <ul style="list-style-type: none"> <li>- <b>SUBQUERIES</b> (truy vấn con)</li> <li>- <b>VIEWS</b> (xem)</li> <li>- <b>INDEXES</b> (chỉ mục)</li> </ul> |
|            | <ul style="list-style-type: none"> <li>- <b>TRANSACTIONS</b> (giao dịch)</li> <li>- <b>STORED PROCEDURES</b> (thủ tục lưu trữ)</li> </ul>              |
|            | <ul style="list-style-type: none"> <li>- <b>FULL-TEXT SEARCH</b> (tìm kiếm văn bản đầy đủ)</li> <li>- <b>TRIGGERS</b> (trigger sự kiện)</li> </ul>     |

| Ví Dụ   |
|---|
| <pre>SELECT * FROM table_name WHERE column_name = value; SELECT column1, column2 FROM table_name;</pre>   |
| <pre>INSERT INTO table_name (column1, column2) VALUES (value1, value2); UPDATE table_name SET column_name = new_value WHERE condition;</pre>  |
| <pre>DELETE FROM table_name WHERE condition;</pre>  |
| <pre>SELECT * FROM table1 INNER JOIN table2 ON table1.column_name = table2.column_name; SELECT * FROM table_name ORDER BY column_name ASC/DESC;</pre>   |
| <pre>SELECT column_name, COUNT(*) FROM table_name GROUP BY column_name HAVING COUNT(*) &gt; 1;</pre>  |
| <pre>SELECT DISTINCT column_name FROM table_name;</pre>   |
| <pre>SELECT * FROM table_name WHERE column_name IN (SELECT column_name FROM another_table); CREATE VIEW view_name AS SELECT column_name FROM table_name WHERE condition; CREATE INDEX index_name ON table_name (column_name);</pre> |
| <pre>START TRANSACTION; SQL_statements; COMMIT; CREATE PROCEDURE procedure_name(parameter_list) AS SQL_statements;</pre>  |
| <pre>SELECT * FROM table_name WHERE MATCH(column_name) AGAINST('search_text'); CREATE TRIGGER trigger_name BEFORE/AFTER INSERT/UPDATE/DELETE ON table_name FOR EACH ROW SQL_statements;</pre>                                       |

## 1. SQL - Ngôn Ngữ Truy Vấn Cơ Bản

### 1.1. Truy Vấn Dữ Liệu (SELECT)

Truy vấn dữ liệu từ một bảng:

► Cú pháp:

`SELECT column1, column2, ...`

`FROM table_name`

`WHERE condition;`

► Ví dụ:

`SELECT * FROM customers WHERE country = 'USA';`

### 1.2. Chèn Dữ Liệu (INSERT)

Chèn dữ liệu vào một bảng:

► Cú pháp:

`INSERT INTO table_name (column1, column2, ...)`

`VALUES (value1, value2, ...);`

► Ví dụ:

`INSERT INTO customers (name, email) VALUES ('John Doe', 'john@example.com');`

### 1.3. Cập Nhật Dữ Liệu (UPDATE)

Cập nhật dữ liệu trong một bảng:

► Cú pháp:

`UPDATE table_name`

`SET column1 = value1, column2 = value2, ...`

`WHERE condition;`

► Ví dụ:

`UPDATE customers SET city = 'New York' WHERE id = 1;`

### 1.4. Xóa Dữ Liệu (DELETE)

Xóa dữ liệu từ một bảng:

► Cú pháp:

`DELETE FROM table_name WHERE condition;`

► Ví dụ:

`DELETE FROM customers WHERE id = 1;`

### 1.5. Tạo Bảng (CREATE TABLE)

Tạo một bảng mới trong cơ sở dữ liệu:

► Cú pháp:

`CREATE TABLE table_name (`

`column1 datatype constraints,`

`column2 datatype constraints,`

`...`

`);`

► Ví dụ:

`CREATE TABLE customers (`  
`id INT PRIMARY KEY,`  
`name VARCHAR(255) NOT NULL,`  
`email VARCHAR(255),`  
`country VARCHAR(50)`  
`);`

### 1.6. Xóa Bảng (DROP TABLE)

Xóa một bảng khỏi cơ sở dữ liệu:

► Cú pháp:

`DROP TABLE table_name;`

► Ví dụ:

`DROP TABLE customers;`

## 2. MySQL - Hệ Quản Trị Cơ Sở Dữ Liệu

MySQL là một hệ quản trị cơ sở dữ liệu quan hệ mã nguồn mở phổ biến. Đây là một số điểm lý thuyết và giải thích chi tiết về MySQL:

### 2.1. Kết Nối đến MySQL Server

Để kết nối đến một máy chủ MySQL từ dòng lệnh, bạn có thể sử dụng lệnh sau:

`mysql -u username -p`

Sau đó, bạn sẽ được yêu cầu nhập mật khẩu để đăng nhập vào MySQL Server.

### 2.2. Tạo Cơ Sở Dữ Liệu

Để tạo một cơ sở dữ liệu mới, bạn có thể sử dụng lệnh SQL CREATE DATABASE:

▶ Cú pháp: `CREATE DATABASE mydatabase;`

### 2.3. Chọn Cơ Sở Dữ Liệu

Sử dụng lệnh USE để chọn cơ sở dữ liệu mà bạn muốn làm việc:

▶ Cú pháp: `USE mydatabase;`

### 2.4. Tạo Bảng

Để tạo một bảng mới, bạn có thể sử dụng lệnh CREATE TABLE, giống như ví dụ SQL ở trên.

### 2.5. Gán Quyền Truy Cập

Bạn có thể sử dụng lệnh GRANT để gán quyền truy cập cho người dùng đến cơ sở dữ liệu hoặc bảng cụ thể:

▶ Cú pháp: `GRANT ALL PRIVILEGES ON mydatabase.* TO 'username'@'localhost';`

## 2.6. Xem Tất Cả Bảng Trong Cơ Sở Dữ Liệu

Để xem tất cả các bảng trong cơ sở dữ liệu, bạn có thể sử dụng lệnh SHOW TABLES:

▶ Cú pháp: `SHOW TABLES;`

### 2.7. Thoát khỏi MySQL

Để thoát khỏi MySQL Command Line, bạn có thể sử dụng lệnh QUIT hoặc EXIT hoặc nhấn Ctrl + D.

Những lệnh và khái niệm này cung cấp một cơ sở vững chắc để bắt đầu làm việc với SQL và MySQL. Để hiểu sâu hơn và trở thành một chuyên gia MySQL, việc tham khảo tài liệu chính thức và thực hành thường xuyên là quan trọng.

## KẾT NỐI VÀ THAO TÁC DỮ LIỆU TRONG ỨNG DỤNG WEB

Dưới đây là hướng dẫn kết nối và thao tác dữ liệu trong ứng dụng web sử dụng SQL Server và MySQL, bao gồm cả lý thuyết, cú pháp và ví dụ tương ứng cho PHP, Django và ASP.NET (3 ngôn ngữ lập trình chính trong tập 2).

### 1. PHP và SQL Server/MySQL

#### 1.1. Lý Thuyết

| Kết Nối với SQL Server   | Kết Nối với MySQL  |
|--|--|
| • Sử dụng extension <code>sqisrv</code> hoặc <code>pdo_sqisrv</code> .                           | • Sử dụng extension <code>mysqli</code> hoặc <code>pdo_mysql</code> .                            |
| • Cung cấp thông tin như tên máy chủ, tên cơ sở dữ liệu, tên người dùng, và mật khẩu để kết nối. | • Cung cấp thông tin như tên máy chủ, tên cơ sở dữ liệu, tên người dùng, và mật khẩu để kết nối. |

## 1.2. Cú Pháp PHP và SQL Server

```
// Kết nối với SQL Server
$serverName = "localhost";
$connectionOptions = array(
    "Database" => "database_name",
    "Uid" => "username",
    "PWD" => "password"
);
$conn = sqlsrv_connect($serverName, $connectionOptions);

// Kết nối với MySQL
$conn = new mysqli("localhost", "username", "password", "database_name");
if ($conn->connect_error) {
    die("Connection failed: " . $conn->connect_error);
}
```

## 1.3. Cú Pháp PHP và MySQL

```
...
$result = $conn->query("SELECT * FROM table_name");
while($row = $result->fetch_assoc()) {
    // Xử lý dữ liệu
}

// Chèn dữ liệu
$sql = "INSERT INTO table_name (column1, column2) VALUES ('value1', 'value2')";
if ($conn->query($sql) === TRUE) {
    echo "New record created successfully";
} else {
    echo "Error: " . $sql . "<br>" . $conn->error;
}
```

## 2. Django (Python) và SQL Server/MySQL

### 2.1. Lý Thuyết

- Sử dụng **ORM (Object-Relational Mapping)** của **Django** để tương tác với cơ sở dữ liệu.
- Cấu hình kết nối trong file **settings.py**.

## 2.2. Cú Pháp Django và SQL Server/MySQL

```
// Kết nối với SQL Server
DATABASES = {
    'default': {
        'ENGINE': 'sql_server.pyodbc',
        'NAME': 'database_name',
        'USER': 'username',
        'PASSWORD': 'password',
        'HOST': 'localhost',
        'PORT': '',
        'OPTIONS': {
            'driver': 'ODBC Driver 17 for SQL Server',
        },
    },
}

// Kết nối với MySQL
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.mysql',
        'NAME': 'database_name',
        'USER': 'username',
        'PASSWORD': 'password',
        'HOST': 'localhost',
        'PORT': '3306',
    }
}
```

## 3. ASP.NET và SQL Server

### 3.1. Lý Thuyết

ASP.NET cung cấp các lớp trong **System.Data.SqlClient** để kết nối và tương tác với SQL Server.

### 3.2. Cú Pháp ASP.NET và SQL Server

```

using System.Data.SqlClient;

// Kết nối với SQL Server
string connectionString = "Data Source=localhost;Initial Catalog=database_name;User
ID=username;Password=password";
using (SqlConnection connection = new SqlConnection(connectionString))
{
    connection.Open();

    // Truy vấn dữ liệu
    using (SqlCommand command = new SqlCommand("SELECT * FROM table_name", connection))
    {
        SqlDataReader reader = command.ExecuteReader();
        while (reader.Read())
        {
            // Xử lý dữ liệu
        }
    }

    // Chèn dữ liệu
    using (SqlCommand command = new SqlCommand("INSERT INTO table_name (column1, column2)
VALUES (@param1, @param2)", connection))
    {
        command.Parameters.AddWithValue("@param1", value1);
        command.Parameters.AddWithValue("@param2", value2);
        int rowsAffected = command.ExecuteNonQuery();
    }
}

```

Lưu ý rằng các giá trị như 'localhost', 'database\_name', 'username', 'password', 'table\_name', và các truy vấn SQL cần được thay thế bằng giá trị tương ứng trong ứng dụng của bạn. Đồng thời, luôn luôn hãy sử dụng các câu lệnh chuẩn SQL hoặc các phương pháp an toàn để tránh tấn công SQL Injection.

### 4. Ngôn ngữ truy vấn sử dụng SQL, MySQL nâng cao

Các truy vấn SQL nâng cao giúp bạn tìm kiếm và xử lý dữ liệu một cách linh hoạt và hiệu quả hơn. Dưới đây là một số truy vấn SQL nâng cao chi tiết kèm ví dụ cụ thể:

### 4.1. JOINS (Liên kết Bảng):

ASP.NET cung cấp các lớp trong `System.Data.SqlClient` để kết nối và tương tác với SQL Server.

#### INNER JOIN

Mã code: SQL

```

SELECT orders.order_id, customers.customer_name
FROM orders
INNER JOIN customers
ON orders.customer_id = customers.customer_id;

```

#### LEFT JOIN

Mã code: SQL

```

SELECT customers.customer_name, orders.order_id
FROM customers
LEFT JOIN orders
ON customers.customer_id = orders.customer_id;

```

### 4.2. Subqueries (Truy vấn Nhỏ)

#### Truy vấn Nhỏ trong SELECT

Mã code: SQL

```

SELECT product_name, (SELECT AVG(product_price) FROM products) AS avg_price
FROM products;

```

## Truy vấn Nhỏ trong WHERE

Mã code: SQL

```
SELECT customer_name
FROM customers
WHERE customer_id IN (SELECT customer_id FROM orders);
```

## 4.3. UNION (Kết Hợp Kết Quả Các Truy Vấn)

Mã code: SQL

```
SELECT product_name FROM table1
UNION
SELECT product_name FROM table2;
```

## 4.4. GROUP BY và HAVING

Mã code: SQL

```
SELECT department, COUNT(*) as total_employees
FROM employees
GROUP BY department
HAVING total_employees > 10;
```



## 4.5. Window Functions (Các Hàm Cửa Sổ)

Mã code: SQL

```
SELECT product_name, product_price,
ROW_NUMBER() OVER (PARTITION BY category ORDER BY product_price) AS rank
FROM products;
```

## 4.6. CASE Statements (Câu Lệnh CASE)

Mã code: SQL

```
SELECT product_name,
CASE
WHEN product_price > 100 THEN 'Expensive'
WHEN product_price > 50 THEN 'Moderate'
ELSE 'Affordable'
END AS price_category
FROM products;
```

## 4.7. Triggers (Kích Hoạt)

Mã code: SQL

```
CREATE TRIGGER after_insert_employee
AFTER INSERT ON employees
FOR EACH ROW
BEGIN
INSERT INTO audit_log (event, event_time)
VALUES ('New employee added', NOW());
END;
```

## 4.8. Stored Procedures (Thủ Tục Lưu Trữ)

| Mã code: **SQL**

```
DELIMITER //

CREATE PROCEDURE GetEmployeeInfo (IN employee_id INT)
BEGIN
    SELECT * FROM employees WHERE id = employee_id;
END //

DELIMITER ;
```

Những truy vấn và công cụ nâng cao này cho phép bạn tương tác với cơ sở dữ liệu một cách linh hoạt và tiện lợi, giúp bạn truy vấn và xử lý dữ liệu một cách hiệu quả hơn trong các ứng dụng SQL của bạn. Hãy thực hành và tìm hiểu thêm để nắm vững cách sử dụng chúng trong các tình huống thực tế.

## BÀI TẬP THỰC HÀNH SQL VÀ MYSQL

### 1. Bài tập cơ bản và nâng cao

Bài tập thực hành chi tiết từ cơ bản đến nâng cao về từng loại **SQL server** và **MySQL** trong lập trình ứng dụng web từ cơ bản đến nâng cao và lời giải. Chủ đề quản lý sản phẩm, đảm bảo sử dụng và khai thác hết cú pháp và câu lệnh của **SQL** và **MYSQL** từ kết nối đến truy vấn theo từng dạng bài tập **tạo mới**, **thêm mới**, **sửa, xoá, update, join , alter, store procedure**.

#### 1.1 Dạng Bài Tập 1: Tạo Bảng và Thêm Dữ Liệu

##### 1.1.1 | SQL Server

**Tạo Bảng Products trong SQL Server:** Tạo bảng **Products** với các cột: ProductID (PK), ProductName, Price, StockQuantity.



| Lời Giải SQL Server:

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName NVARCHAR(255),
    Price DECIMAL(10, 2),
    StockQuantity INT
);
```

**Thêm Dữ Liệu vào Bảng Products trong SQL Server:** Chèn ít nhất 5 sản phẩm vào bảng **Products**.

| Lời Giải SQL Server:

```
INSERT INTO Products (ProductID, ProductName, Price, StockQuantity)
VALUES (1, N'Product 1', 10.99, 100),
       (2, N'Product 2', 15.99, 50),
       (3, N'Product 3', 8.49, 75),
       (4, N'Product 4', 23.95, 30),
       (5, N'Product 5', 19.99, 90);
```

#### 1.1.2 | MySQL

**Tạo Bảng Products:** Tạo bảng **Products** với các cột: ProductID (PK), ProductName, Price, StockQuantity.

| Lời Giải MySQL:

```
CREATE TABLE Products (
    ProductID INT PRIMARY KEY,
    ProductName VARCHAR(255),
    Price DECIMAL(10, 2),
    StockQuantity INT
);
```



**Thêm Dữ Liệu vào Bảng Products trong MySQL:** Chèn ít nhất 5 sản phẩm vào bảng Products. ProductID (PK), ProductName, Price, StockQuantity.

| Lời Giải MySQL:

```
INSERT INTO Products (ProductID, ProductName, Price, StockQuantity)
VALUES (1, 'Product 1', 18.99, 100),
       (2, 'Product 2', 15.99, 50),
       (3, 'Product 3', 8.49, 75),
       (4, 'Product 4', 23.95, 30),
       (5, 'Product 5', 19.99, 98);
```

## 1.2. Dạng Bài Tập 2: Truy Vấn Dữ Liệu và Cập Nhật

### 1.2.1 | SQL Server

**Truy Vấn Dữ Liệu từ Bảng Products:** Viết truy vấn SQL để lấy ra danh sách tất cả sản phẩm từ bảng Products.

| Lời Giải SQL Server:

```
SELECT * FROM Products;
```

**Cập Nhật Dữ Liệu trong Bảng Products:** Cập nhật giá của sản phẩm có ProductID là 1 thành 12.99.

| Lời Giải SQL Server:

```
UPDATE Products
SET Price = 12.99
WHERE ProductID = 1;
```

### 1.2.2 | MySQL

**Truy Vấn Dữ Liệu từ Bảng Products:** Viết truy vấn SQL để lấy ra danh sách tất cả sản phẩm từ bảng Products.

| Lời Giải MySQL:

```
SELECT * FROM Products;
```

**Cập Nhật Dữ Liệu trong Bảng Products:** Cập nhật giá của sản phẩm có ProductID là 1 thành 12.99.

| Lời Giải MySQL:

```
UPDATE Products
SET Price = 12.99
WHERE ProductID = 1;
```

## 1.3. Dạng Bài Tập 3: Xoá và Thao Tác Liên Kết Bảng

### 1.3.1 | SQL Server

**Xoá Sản Phẩm:** Xoá sản phẩm có ProductID là 3.

| Lời Giải SQL Server:

```
DELETE FROM Products
WHERE ProductID = 3;
```

**Liên Kết Bảng và Truy Vấn Kết Hợp:** Viết truy vấn SQL để lấy ra danh sách các đơn đặt hàng, bao gồm thông tin về sản phẩm từ bảng Orders và Products.

## I Lời Giải SQL Server:

```
SELECT O.OrderID, P.ProductName, P.Price, O.Quantity, O.OrderDate
FROM Orders O
JOIN Products P ON O.ProductID = P.ProductID;
```

## 1.3.2 MySQL

**Xoá Sản Phẩm:** Xoá sản phẩm có ProductID là 3.

## I Lời Giải MySQL:

```
DELETE FROM Products
WHERE ProductID = 3;
```

**Liên Kết Bảng và Truy Vấn Kết Hợp:** Viết truy vấn SQL để lấy ra danh sách các đơn đặt hàng, bao gồm thông tin về sản phẩm từ bảng Orders và Products.

## I Lời Giải SQL Server:

```
SELECT O.OrderID, P.ProductName, P.Price, O.Quantity, O.OrderDate
FROM Orders O
JOIN Products P ON O.ProductID = P.ProductID;
```

## 1.4. Dạng Bài Tập 4: Stored Procedures và ALTER TABLE

## 1.4.1 SQL Server

**Stored Procedure:** Viết một stored procedure để lấy ra danh sách tất cả sản phẩm từ bảng Products.



## I Lời Giải SQL Server:

```
CREATE PROCEDURE GetProducts
AS
BEGIN
    SELECT * FROM Products;
END;
```

**Thêm Cột vào Bảng Products:** Thêm cột Manufacturer kiểu NVARCHAR(255) vào bảng Products.

## I Lời Giải SQL Server:

```
ALTER TABLE Products
ADD Manufacturer NVARCHAR(255);
```

## 1.4.2 MySQL

**Stored Procedure:** Viết một stored procedure để lấy ra danh sách tất cả sản phẩm từ bảng Products.

## I Lời Giải MySQL:

```
DELIMITER //
CREATE PROCEDURE GetProducts()
BEGIN
    SELECT * FROM Products;
END //
DELIMITER ;
```



Thêm Cột vào Bảng Products: Thêm cột Manufacturer kiểu VARCHAR(255) vào bảng Products.

| Lời Giải MySQL:

```
ALTER TABLE Products  
ADD COLUMN Manufacturer VARCHAR(255);
```

Đây là một loạt các bài tập từ cơ bản đến nâng cao với SQL Server và MySQL trong lập trình ứng dụng web. Bạn có thể sử dụng các lời giải này để thực hành và nâng cao kỹ năng SQL của mình.

## 2. Bài tập tự thực hành chuyên sâu

### 2.1. Tạo Bảng và Thêm Dữ Liệu

Tạo một bảng trong MySQL để lưu trữ thông tin về sản phẩm (ID, tên sản phẩm, giá, danh mục). Thêm một số dữ liệu vào bảng.

### 2.2. Truy Vấn Dữ Liệu

Viết các truy vấn SQL để truy xuất thông tin về sản phẩm. Ví dụ: Tất cả sản phẩm trong một danh mục cụ thể, sản phẩm có giá cao nhất, số lượng sản phẩm theo danh mục, v.v.

### 2.3. Kết Hợp Dữ Liệu Từ Nhiều Bảng

Tạo một bảng mới để lưu trữ thông tin về đơn hàng (ID đơn hàng, ID sản phẩm, số lượng, ngày đặt hàng). Viết các truy vấn SQL kết hợp dữ liệu từ cả hai bảng để lấy thông tin về đơn hàng, bao gồm tên sản phẩm và giá.

### 2.4. Cập Nhật và Xóa Dữ Liệu

Viết các truy vấn SQL để cập nhật thông tin của một sản phẩm hoặc xoá một sản phẩm khỏi bảng.

### 2.5. Thao Tác Trên Dữ Liệu JSON

Lưu trữ dữ liệu dưới dạng JSON trong một cột của bảng MySQL. Viết các truy vấn SQL để truy cập và cập nhật thông tin trong các trường JSON.

### 2.6. Viết Store Procedure

Viết một stored procedure trong MySQL để thực hiện một nhiệm vụ cụ thể, ví dụ: Tính tổng giá trị đơn hàng, lấy danh sách sản phẩm theo danh mục.

### 2.7. Tối Ưu Hóa Truy Vấn

Xác định và tối ưu hóa các truy vấn SQL để đảm bảo hiệu suất tốt, bao gồm việc sử dụng chỉ mục, viết truy vấn sử dụng JOIN thích hợp, và tối ưu hóa các truy vấn phức tạp.

### 2.8. Sử Dụng Triggers

Tạo một trigger trong MySQL để tự động cập nhật một trường hoặc thực hiện một hành động khác khi có sự thay đổi trên bảng.

### Bài Tập 1: Quản lý Sản Phẩm

Đề bài:

- Tạo bảng "products" có các trường: product\_id, product\_name, price, quantity.
- Thêm ít nhất 5 sản phẩm vào bảng.
- Lấy tên và giá của các sản phẩm có giá lớn hơn 50.

## | Lời Giải: Mã SQL

- ▶ Tạo bảng "products" có các trường: product\_id, product\_name, price, quantity

```
CREATE TABLE products (
    product_id INT PRIMARY KEY,
    product_name VARCHAR(255),
    price DECIMAL(10, 2),
    quantity INT
);
```

- ▶ Thêm ít nhất 5 sản phẩm vào bảng

```
INSERT INTO products (product_id, product_name, price, quantity)
VALUES (1, 'Laptop', 600, 10),
       (2, 'Smartphone', 400, 20),
       (3, 'Tablet', 200, 15),
       (4, 'Headphones', 50, 30),
       (5, 'Printer', 150, 8);
```

- ▶ Lấy tên và giá của các sản phẩm có giá lớn hơn 50

```
SELECT product_name, price FROM products WHERE price > 50;
```

## Bài Tập 2: Quản lý Đơn Hàng

## Đề bài:

- Tạo bảng "orders" có các trường: order\_id, customer\_id, product\_id, quantity, order\_date.
- Thêm ít nhất 3 đơn hàng vào bảng.
- Lấy thông tin đơn hàng (order\_id, customer\_id, product\_name, quantity) kèm theo tên sản phẩm.

## | Lời Giải: Mã SQL

- ▶ Tạo bảng "orders" có các trường: order\_id, customer\_id, product\_id, quantity, order\_date

```
CREATE TABLE orders (
    order_id INT PRIMARY KEY,
    customer_id INT,
    product_id INT,
    quantity INT,
    order_date DATE
);
```



- ▶ Thêm ít nhất 3 đơn hàng vào bảng

```
INSERT INTO orders (order_id, customer_id, product_id, quantity, order_date)
VALUES (1, 101, 1, 2, '2023-01-15'),
       (2, 102, 3, 5, '2023-01-16'),
       (3, 103, 2, 3, '2023-01-17');
```

- Lấy thông tin đơn hàng (order\_id, customer\_id, product\_name, quantity) kèm theo tên sản phẩm

```
SELECT o.order_id, o.customer_id, p.product_name, o.quantity
FROM orders o
INNER JOIN products p ON o.product_id = p.product_id;
```

### Bài Tập 3: Quản lý Khách Hàng

Đề bài:

- Tạo bảng "customers" có các trường: customer\_id, customer\_name, email, phone.
- Thêm ít nhất 3 khách hàng vào bảng.
- Lấy thông tin khách hàng (customer\_id, customer\_name) kèm theo số đơn hàng mà họ đã đặt.

#### I Lời Giải: Mã SQL

- Tạo bảng "customers" có các trường: customer\_id, customer\_name, email, phone

```
CREATE TABLE customers (
    customer_id INT PRIMARY KEY,
    customer_name VARCHAR(255),
    email VARCHAR(255),
    phone VARCHAR(20)
);
```

- Thêm ít nhất 3 khách hàng vào bảng

```
INSERT INTO customers (customer_id, customer_name, email, phone)
VALUES (101, 'Alice Johnson', 'alice@example.com', '123-456-7890'),
       (102, 'Bob Smith', 'bob@example.com', '987-654-3210'),
       (103, 'Eva Davis', 'eva@example.com', '111-222-3333');
```

- Lấy thông tin khách hàng (customer\_id, customer\_name) kèm theo số đơn hàng mà họ đã đặt

```
SELECT c.customer_id, c.customer_name, COUNT(o.order_id) AS total_orders
FROM customers c
LEFT JOIN orders o ON c.customer_id = o.customer_id
GROUP BY c.customer_id;
```

Những bài tập này sẽ giúp bạn rèn luyện kỹ năng SQL và làm quen với việc làm việc với cơ sở dữ liệu MySQL từ các thao tác cơ bản đến các chức năng phức tạp và tối ưu hóa hiệu suất. Hãy thử thực hiện chúng trên một môi trường thử nghiệm hoặc trên một máy chủ cơ sở dữ liệu thực tế để có trải nghiệm thực tế hơn.

## PHẦN 2: LẬP TRÌNH ỨNG DỤNG



### CHƯƠNG VI FRAMEWORKS VÀ THƯ VIỆN PHỔ BIẾN

#### GIỚI THIỆU VỀ REACT, ANGULAR, VÀ VUE.JS

##### 1. React

###### 1.1. Khái Niệm

React là một thư viện JavaScript cho việc xây dựng giao diện người dùng (UI) tương tác. Nó sử dụng cơ chế “component-based architecture”, trong đó giao diện người dùng được chia thành các thành phần (components) riêng lẻ. Các components có thể tái sử dụng, giúp tạo ra ứng dụng linh hoạt và dễ bảo trì.

###### ► **Bí quyết căn bản**

```
import React from 'react';

class MyComponent extends React.Component {
  render() {
    return <div>Hello, World!</div>;
  }
}

// Sử dụng component trong ứng dụng
ReactDOM.render(<MyComponent />, document.getElementById('root'));
```

## 1.2. Ví dụ thực tế

- Một ví dụ về việc sử dụng React để tạo một danh sách sản phẩm

```
import React from 'react';

class ProductList extends React.Component {
  render() {
    const products = ['Product 1', 'Product 2', 'Product 3'];
    return (
      <ul>
        {products.map(product => (
          <li key={product}>{product}</li>
        ))}
      </ul>
    );
  }
}

ReactDOM.render(<ProductList />, document.getElementById('root'));
```

## 2. Angular

- Khái niệm

Angular là một framework phát triển bởi Google, chuyên về việc xây dựng ứng dụng web đầy đủ với các thành phần như routing, forms, và HTTP modules. Angular sử dụng TypeScript, một siêu tập lệnh của JavaScript, giúp phát hiện lỗi và tăng tính dễ bảo trì của mã nguồn.



130

## 2.2. Cú pháp cơ bản

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-root',
  template: '<div>Hello, World!</div>',
})
export class AppComponent { }
```

## 2.3. Ví dụ thực tế

Một ví dụ về việc sử dụng Angular để tạo một danh sách sản phẩm:

```
import { Component } from '@angular/core';

@Component({
  selector: 'app-product-list',
  template:
    <ul>
      <li *ngFor="let product of products">{{ product }}</li>
    </ul>
})
export class ProductListComponent {
  products = ['Product 1', 'Product 2', 'Product 3'];
}
```

131

### 3. Vue.js

#### 3.1. Khái niệm

Vue.js là một framework nguồn mở được thiết kế để xây dựng giao diện người dùng và dễ dàng tích hợp vào các dự án hiện tại. Nó sử dụng cú pháp gần giống HTML thông thường để định nghĩa giao diện người dùng và sử dụng các directives để tương tác với DOM.

#### 3.2. Cú pháp cơ bản

```
<template>
  <div>Hello, World!</div>
</template>

<script>
export default {
  name: 'MyComponent',
};
</script>
```



132

#### 3.3. Ví dụ thực tế

Một ví dụ về việc sử dụng Vue.js để tạo một danh sách sản phẩm:

```
<template>
  <ul>
    <li v-for="product in products" :key="product">{{ product }}</li>
  </ul>
</template>

<script>
export default {
  data() {
    return {
      products: ['Product 1', 'Product 2', 'Product 3'],
    };
  },
};
</script>
```

## SỬ DỤNG THƯ VIỆN JQUERY VÀ Lodash

#### 1. Khái niệm

##### 1.1. jQuery

jQuery là một thư viện JavaScript phổ biến được thiết kế để giúp việc tương tác với các phần tử HTML, xử lý sự kiện, thay đổi nội dung trang web và thao tác AJAX trở nên đơn giản. jQuery cung cấp cú pháp ngắn gọn và dễ đọc, giúp lập trình viên tập trung vào logic kinh doanh thay vì các thao tác DOM phức tạp.

##### 1.2. Lodash

Lodash là một thư viện JavaScript tiện ích (utility library) mạnh mẽ, cung cấp rất nhiều hàm giúp thao tác và xử lý dữ liệu dễ dàng hơn. Lodash giúp giảm độ phức tạp của việc viết mã, tối ưu hóa hiệu suất và cung cấp các hàm tiện ích không có sẵn trong JavaScript cơ bản.

```

// Chọn phần tử HTML và thao tác trên nó
$("#myElement").addClass("highlight");

// Bắt sự kiện và xử lý nó
$("#myButton").click(function(){
    alert("Button clicked!");
});

// Thao tác AJAX đơn giản
$.ajax({
    url: "example.com/api/data",
    method: "GET",
    success: function(data){
        console.log(data);
    }
});

```

## 2.2. Lodash

```

// Sử dụng hàm Lodash
var numbers = [1, 2, 3, 4, 5];

// Lấy tổng của mảng số
var sum = _.sum(numbers);
console.log("Sum of numbers: " + sum);

// Lọc các phần tử trong mảng
var evens = _.filter(numbers, function(num){
    return num % 2 === 0;
});
console.log("Even numbers: " + evens);

```

```

<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>API Data Display</title>
    <script src="https://code.jquery.com/jquery-3.6.0.min.js"></script>
    <script src="https://cdn.jsdelivr.net/npm/lodash@4.17.21/lodash.min.js"></script>
</head>
<body>
    <h1>API Data Display</h1>
    <ul id="dataList"></ul>

    <script>
        // Sử dụng jQuery để lấy dữ liệu từ API
        $(document).ready(function(){
            $.ajax({
                url: "https://jsonplaceholder.typicode.com/posts",
                method: "GET",
                success: function(data){
                    // Sử dụng Lodash để lọc và hiển thị dữ liệu
                    var filteredData = _.take(data, 5); // Lấy 5 bài đăng đầu tiên
                    _.forEach(filteredData, function(post){
                        $("#dataList").append("<li>" + post.title + "</li>");
                    });
                }
            });
        });
    </script>
</body>
</html>

```

**Phân tích ví dụ thực tế:** Trong ví dụ này, chúng ta sử dụng jQuery để thực hiện một yêu cầu GET đến một API endpoint (trong trường hợp này, chúng ta sử dụng JSONPlaceholder, một API giả mạo cho mục đích thử nghiệm) và nhận dữ liệu từ server. Sau đó, chúng ta sử dụng Lodash để lọc và hiển thị 5 bài đăng đầu tiên từ dữ liệu nhận được. Điều này giúp chúng ta hiểu rõ cách jQuery và Lodash có thể được sử dụng cùng nhau để tạo ra các ứng dụng web mạnh mẽ và dễ dàng quản lý dữ liệu.

## 4. Django

### ► Khái niệm

**Django:** Là một framework phát triển web Python được thiết kế để xây dựng ứng dụng web nhanh chóng và bảo mật.

### ► Đặc điểm

**ORM (Object-Relational Mapping):** Tương tác với cơ sở dữ liệu bằng Python thay vì SQL truyền thống.

**Admin Interface:** Hệ thống quản lý giao diện admin được tạo tự động từ các models.

**Batteries-Included:** Cung cấp nhiều tính năng sẵn có như xác thực người dùng, quản lý form, và xử lý tệp tin.

### ► Ví dụ

```
from django.http import HttpResponse

def hello(request):
    return HttpResponse("Hello, Django!")
```

## 5. Ruby on Rails

### ► Khái niệm

**Rails:** Là một framework phát triển web Ruby mạnh mẽ và yêu thích, được thiết kế để tối thiểu hóa việc lặp lại và giúp lập trình viên xây dựng ứng dụng web nhanh chóng.

### , Đặc điểm

**ORM (Object-Relational Mapping):** Sử dụng các quy ước để giảm bớt việc cấu hình.

**Don't Repeat Yourself (DRY):** Hạn chế việc lặp lại mã nguồn thông qua việc sử dụng các kỹ thuật như ActiveRecord.

**RESTful Routes:** Hỗ trợ việc xây dựng các routes RESTful dễ dàng.

### , Ví dụ

```
class UsersController < ApplicationController
  def index
    @users = User.all
  end
end
```

## KIẾN THỨC CƠ BẢN VỀ FRAMEWORKS VÀ THƯ VIỆN

### 1. Khái Niệm Cơ Bản

#### 1.1. Thư viện (Library):

Một thư viện là một tập hợp các hàm, module, và công cụ giúp lập trình viên thực hiện các nhiệm vụ cụ thể một cách dễ dàng hơn. Thư viện thường tập trung vào một lĩnh vực cụ thể như xử lý DOM, thao tác dữ liệu, hoặc tương tác với API. Lập trình viên có thể sử dụng các hàm từ thư viện để giảm bớt công sức cần thiết để viết mã.



## 1.2. Framework:

Một framework là một cấu trúc cơ bản được thiết kế để giúp xây dựng các ứng dụng hoặc dự án một cách nhanh chóng và hiệu quả. Framework cung cấp một kiến trúc chuẩn, các quy tắc và các công cụ giúp lập trình viên xây dựng ứng dụng một cách có tổ chức và dễ bảo trì. Các framework thường đi kèm với các quy tắc và hướng dẫn sử dụng để giúp lập trình viên phát triển ứng dụng một cách nhất quán.

## 2. Sự Khác Biệt Giữa Thư Viện và Framework

### ▶ **Khả năng tùy chọn**

**Thư viện:** Lập trình viên có tự do lựa chọn và sử dụng các hàm từ thư viện theo ý muốn mà không bị ràng buộc bởi một kiến trúc cụ thể.

**Framework:** Lập trình viên phải tuân theo kiến trúc và quy tắc của framework, điều này giúp đảm bảo tính nhất quán trong ứng dụng.

### ▶ **Kiến trúc**

**Thư viện:** Không yêu cầu một kiến trúc cụ thể, chỉ cần gọi các hàm cần thiết khi cần sử dụng.

**Framework:** Được thiết kế theo một kiến trúc chuẩn, đưa ra cấu trúc và quy tắc cho việc phát triển ứng dụng.

### ▶ **Quy mô dự án**

**Thư viện:** Thích hợp cho các dự án nhỏ hoặc các nhiệm vụ cụ thể như xử lý DOM hoặc thao tác dữ liệu.

**Framework:** Thích hợp cho các dự án lớn và phức tạp, cung cấp các giải pháp và hỗ trợ cho nhiều khía cạnh của ứng dụng.

## BÀI TẬP THỰC HÀNH

### 1. Bài Tập Cơ Bản: Hiển Thị Danh Sách Bài Viết

**React và Axios:**

#### 1.1. Yêu cầu

- Tạo một ứng dụng React mới bằng Create React App.
- Sử dụng Axios để gửi yêu cầu GET đến API: <https://jsonplaceholder.typicode.com/posts> để lấy danh sách bài viết.
- Hiển thị danh sách bài viết trên giao diện người dùng.

#### 1.2. Mã nguồn

```
import React, { useState, useEffect } from 'react';
import axios from 'axios';

function App() {
  const [posts, setPosts] = useState([]);

  useEffect(() => {
    axios.get('https://jsonplaceholder.typicode.com/posts')
      .then(response => {
        setPosts(response.data);
      })
      .catch(error => {
        console.error(error);
      });
  }, []);

  return (
    <div>
      <h1>Danh Sách Bài Viết</h1>
      <ul>
        {posts.map(post => (
          <li key={post.id}>{post.title}</li>
        ))}
      </ul>
    </div>
  );
}

export default App;
```

## 2. Bài Tập Trung Bình: Quản Lý Người Dùng

Django và Django REST Framework:

### 2.1. Yêu cầu

- Tạo một ứng dụng Django mới với Django REST Framework.
- Định nghĩa model cho Người dùng với các trường như Tên, Email, Mật khẩu.
- Tạo API endpoints cho việc xem, tạo, cập nhật, và xóa người dùng.

### 2.2. Mã nguồn

```
# models.py
from django.db import models

class User(models.Model):
    name = models.CharField(max_length=100)
    email = models.EmailField(unique=True)
    password = models.CharField(max_length=100)

# serializers.py
from rest_framework import serializers
from .models import User

class UserSerializer(serializers.ModelSerializer):
    class Meta:
        model = User
        fields = '__all__'

# views.py
from rest_framework import viewsets
from .models import User
from .serializers import UserSerializer

class UserViewSet(viewsets.ModelViewSet):
    queryset = User.objects.all()
    serializer_class = UserSerializer
```

140

## 3. Bài Tập Nâng Cao: Ứng Dụng Quản Lý Sản Phẩm

Angular và Express.js

### 3.1. Yêu cầu:

- Tạo một ứng dụng Angular với một danh sách sản phẩm.
- Sử dụng Express.js để tạo một API cho việc xem và quản lý sản phẩm (CRUD operations).

### 3.2. Mã nguồn:

```
// product.model.ts
export interface Product {
    id: number;
    name: string;
    price: number;
}

// product.service.ts
import { Injectable } from '@angular/core';
import { HttpClient } from '@angular/common/http';
import { Observable } from 'rxjs';
import { Product } from './product.model';

@Injectable({
    providedIn: 'root'
})
export class ProductService {
    private apiUrl = 'http://localhost:3000/products';

    constructor(private http: HttpClient) {}

    getProducts(): Observable<Product[]> {
        return this.http.get<Product[]>(this.apiUrl);
    }
}
```

141

```
// product-list.component.ts
import { Component, OnInit } from '@angular/core';
import { ProductService } from './product.service';
import { Product } from './product.model';

@Component({
  selector: 'app-product-list',
  templateUrl: './product-list.component.html',
  styleUrls: ['./product-list.component.css']
})
export class ProductListComponent implements OnInit {
  products: Product[] = [];

  constructor(private productService: ProductService) {}

  ngOnInit(): void {
    this.productService.getProducts().subscribe(data => {
      this.products = data;
    });
  }
}
```

Nhớ rằng, để chạy các ví dụ trên, bạn cần cài đặt các **dependencies như React, Angular CLI, Django, Django REST Framework, Express.js, và thư viện kết nối HTTP tương ứng**. Đồng thời, cần cấu hình **CORS (Cross-Origin Resource Sharing)** cho các ứng dụng khi chúng giao tiếp qua các domain khác nhau.



## AUTHENTICATION VÀ SECURITY

### XÁC THỰC NGƯỜI DÙNG VÀ TÀI KHOẢN

Xác thực người dùng và quản lý tài khoản là một phần quan trọng trong phát triển ứng dụng website. Dưới đây là một hướng dẫn cơ bản về cách xác thực người dùng và quản lý tài khoản trong lập trình website:

#### 1. Xác Thực Người Dùng

##### ► Xác Thực Tên Người Dùng và Mật Khẩu

- Người dùng nhập tên người dùng và mật khẩu vào form đăng nhập.
- Mật khẩu thường được băm (hash) trước khi lưu trữ trong cơ sở dữ liệu để đảm bảo an toàn.
- Server kiểm tra thông tin đăng nhập với cơ sở dữ liệu và cho phép truy cập nếu thông tin chính xác.

##### ► Xác Thực Bằng Email hoặc Số Điện Thoại

- Người dùng đăng ký với email hoặc số điện thoại của họ.
- Một mã xác nhận hoặc liên kết được gửi đến email hoặc số điện thoại.
- Người dùng nhập mã xác nhận hoặc nhấn vào liên kết để xác nhận danh tính.

##### ► Xác Thực Bằng Dịch Vụ Bên Ngoài (OAuth, Google, Facebook, v.v)

- Cho phép người dùng đăng nhập bằng tài khoản từ các dịch vụ như Google, Facebook, hoặc Twitter.

- Sử dụng OAuth hoặc các thư viện xác thực bên ngoài để thiết lập kết nối và xác thực.

## 2. Quản Lý Tài Khoản

### ► Đăng Ký Tài Khoản

- Người dùng nhập thông tin cần thiết vào form đăng ký (tên, email, mật khẩu, v.v.).
- Thông tin đăng ký được lưu trữ trong cơ sở dữ liệu.

### ► Đặt Lại Mật Khẩu

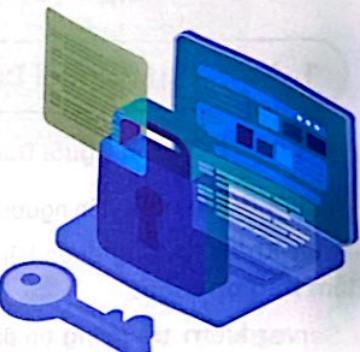
- Người dùng yêu cầu đặt lại mật khẩu nếu quên mật khẩu cũ.
- Một liên kết hoặc mã xác nhận được gửi đến email hoặc số điện thoại để thiết lập mật khẩu mới.

### ► Đăng Xuất (Logout)

- Người dùng nhấn nút "Đăng Xuất" để kết thúc phiên làm việc và đăng xuất khỏi tài khoản.

### ► Xóa Tài Khoản

- Người dùng có thể yêu cầu xóa tài khoản nếu không muốn sử dụng dịch vụ nữa.
- Tài khoản và dữ liệu liên quan được xóa khỏi cơ sở dữ liệu.



## 3. Bảo Mật và Bảo Vệ Dữ Liệu

- Sử dụng HTTPS để bảo mật giao tiếp giữa máy khách và máy chủ.
- Sử dụng mã hóa mạnh để lưu trữ mật khẩu người dùng trong cơ sở dữ liệu.
- Tránh hiển thị thông tin nhạy cảm như mật khẩu trong các gói dữ liệu gửi đi.

**Lưu ý:** Khi triển khai xác thực người dùng và quản lý tài khoản, luôn luôn tìm hiểu và tuân thủ các hướng dẫn bảo mật và chuẩn mã hóa để đảm bảo an toàn cho người dùng. Sử dụng thư viện và framework an toàn để giảm thiểu lỗ hổng bảo mật.

# BẢO MẬT ỨNG DỤNG WEB: XSS, CSRF, SQL INJECTION

## 1. XSS (Cross-Site Scripting)

### 1.1. Lý Thuyết

XSS là kỹ thuật tấn công khi kẻ tấn công chèn mã JavaScript độc hại vào trang web, được thực thi trên trình duyệt của người dùng.

### 1.2. Cú Pháp

```
<script>alert('XSS Attack');</script>
```

### 1.3. Mã Code Người Dùng

```
// Khi người dùng nhập dữ liệu vào trang web và nó không được kiểm tra hoặc làm sạch
let userInput = "<script>alert('XSS Attack');</script>";
document.getElementById("comment").innerHTML = userInput;
```

### 1.4. Phân tích

Mã JavaScript độc hại sẽ được chạy trên trình duyệt của người dùng khi trang web hiển thị nội dung không an toàn.

### 1.5. Ngăn Chặn XSS

- Sử dụng HTML escaping hoặc encode dữ liệu người dùng trước khi hiển thị.
- Sử dụng Content Security Policy (CSP) để giới hạn nguồn tải tài nguyên và ngăn chặn XSS.



## 2. CSRF (Cross-Site Request Forgery)

### 2.1. Lý Thuyết

**CSRF:** Là kỹ thuật tấn công khi kẻ tấn công lừa người dùng đăng nhập vào trang web và sau đó thực hiện các hành động không mong muốn trên trang web mà người dùng đã đăng nhập.

### 2.2. Cú Pháp

```

Mã Code Người Dùng:
// Khi người dùng đăng nhập vào trang web và duy trì phiên đăng nhập
// Hình ảnh động không hiển thị, chỉ là ví dụ minh họa cách tấn công
let img = new Image();
img.src = "https://bank.com/transfer?
toAccount=AttackerAccount&amount=1000&csrfToken=VICTIM_CSRF_TOKEN";
```

### 2.3. Phân Tích

- Khi người dùng đăng nhập vào trang web và mở một trang web khác có chứa mã tấn công, yêu cầu gửi đến trang web đã đăng nhập sẽ được thực hiện.

### 2.4. Ngăn Chặn CSRF

- Sử dụng token CSRF (Double Submit Cookies, Synchronizer Token Pattern) để xác minh mỗi yêu cầu gửi từ người dùng có nguồn gốc hợp lệ.
- Sử dụng SameSite cookie attribute để giảm thiểu rủi ro CSRF.

## 3. SQL Injection

### ► Lý Thuyết

**SQL Injection:** Là kỹ thuật tấn công khi kẻ tấn công chèn câu lệnh SQL độc hại vào các trường dữ liệu đầu vào, thực hiện các truy vấn không mong muốn trong cơ sở dữ liệu.

### Cú Pháp

SQL:

```
SELECT * FROM users WHERE username='admin' AND password='12345'
```

### ► Mã Code Người Dùng

```
// Khi người dùng nhập dữ liệu vào trang web và nó không được kiểm tra trước làm sạch
let userInput = '' OR '1'='1';
let query = "SELECT * FROM users WHERE username=" + userInput + " AND
password='12345'";
// Thực hiện truy vấn SQL sử dụng query
```

### ► Phân tích

- Câu lệnh SQL độc hại (' OR '1'='1') khiến cho điều kiện truy vấn trở thành luôn đúng, trả về tất cả dữ liệu từ bảng users.

### ► Ngăn Chặn SQL Injection

- Sử dụng prepared statements hoặc parameterized queries để ngăn chặn việc chèn câu lệnh SQL vào dữ liệu đầu vào.
- Kiểm tra và làm sạch dữ liệu đầu vào trước khi sử dụng chúng trong truy vấn SQL.

## SỬ DỤNG HTTPS VÀ BẢO MẬT DỮ LIỆU TRONG LẬP TRÌNH WEBSITE

### 1. Phân Tích

#### 1.1. HTTPS (SSL/TLS Encryption)

**Lý Thuyết:** HTTPS sử dụng SSL/TLS để mã hóa dữ liệu truyền tải giữa trình duyệt người dùng và máy chủ web.

**Cú Pháp:** Cài đặt chứng chỉ SSL/TLS, kết nối qua cổng 443.

**Ví Dụ Thực Tế:** Cấu hình máy chủ web (Apache hoặc Nginx) để hỗ trợ HTTPS.

**Sử dụng Let's Encrypt:** Để lấy chứng chỉ SSL/TLS miễn phí. Cài đặt chuyển hướng từ HTTP sang HTTPS để đảm bảo tất cả yêu cầu đều sử dụng HTTPS.

## 1.2. Bảo Mật Dữ Liệu

**Lý Thuyết:** Bảo mật dữ liệu bao gồm việc mã hóa, xác thực, và ngăn chặn tấn công để bảo vệ thông tin người dùng và dữ liệu ứng dụng.

**Cú Pháp:** Sử dụng mã hóa dữ liệu, kiểm tra người dùng, quản lý mật khẩu, ngăn chặn CSRF, SQL Injection, và XSS Attacks.

## 1.3. Mã Hóa Dữ Liệu

- Sử dụng HTTPS để mã hóa dữ liệu truyền tải giữa máy chủ và trình duyệt.
- Mã hóa dữ liệu trong cơ sở dữ liệu, ví dụ: sử dụng bcrypt để lưu trữ mật khẩu người dùng.

## 1.4. Kiểm Tra Người Dùng và Mật Khẩu

- Sử dụng xác thực hai yếu tố (2FA) để bảo vệ tài khoản người dùng.
- Sử dụng hashed passwords và salt để lưu trữ mật khẩu người dùng một cách an toàn.

## 1.5. Ngăn Chặn Tấn Công CSRF, SQL Injection và XSS

- Sử dụng token CSRF để ngăn chặn CSRF Attacks.
- Sử dụng prepared statements hoặc ORM để ngăn chặn SQL Injection Attacks.
- Sử dụng HTML escaping và Content Security Policy (CSP) để ngăn chặn XSS Attacks.

## 2. Ví Dụ Thực Tế

### 2.1. Cấu Hình HTTPS trong Node.js với Express

- ▶ **Cài Đặt Dependencies:** npm install express https fs
- ▶ **Cấu Hình Máy Chủ HTTPS**

```
const express = require('express');
const https = require('https');
const fs = require('fs');

const app = express();

// ...Cấu hình ứng dụng Express...

const httpsOptions = {
  key: fs.readFileSync('path/to/private-key.pem'),
  cert: fs.readFileSync('path/to/certificate.pem')
};

https.createServer(httpsOptions, app).listen(443, () => {
  console.log('Server is running on port 443 (HTTPS)');
});
```

## 2.2. Bảo Mật Dữ Liệu và Kiểm Tra Mật Khẩu trong Node.js

### ▶ Sử Dụng Bcrypt Để Hash Mật Khẩu

```
const bcrypt = require('bcrypt');
const saltRounds = 10;

const plaintextPassword = 'securepassword';
bcrypt.hash(plaintextPassword, saltRounds, (err, hash) => {
  if (err) throw err;
  // Lưu trữ 'hash' trong cơ sở dữ liệu
});
```

### ▶ Xác Thực Mật Khẩu

```

const hashedPasswordFromDB = '...'; // Lấy 'hash' từ cơ sở dữ liệu
const loginPassword = 'userinputpassword';

bcrypt.compare(loginPassword, hashedPasswordFromDB, (err, result) => {
  if (err) throw err;
  if (result) {
    // Mật khẩu đúng, cho phép người dùng đăng nhập
  } else {
    // Mật khẩu sai, thông báo lỗi
  }
});

```

» **Lưu ý:** Đây chỉ là các ví dụ cơ bản và thực tế. Bảo mật ứng dụng yêu cầu kiến thức sâu hơn về mã hóa, xác thực, và các biện pháp bảo mật khác. Luôn tìm hiểu và áp dụng các tiêu chuẩn an toàn khi phát triển ứng dụng web.

## BÀI TẬP THỰC HÀNH

### 1. Bài Tập 1: Xác Thực Người Dùng

#### 1.1. Tạo Trang Đăng Nhập và Đăng Ký

**Lời Giải:** Tạo hai trang HTML: login.html và register.html với các trường nhập dữ liệu tương ứng.

#### 1.2. Xử Lý Đăng Nhập và Đăng Ký

**Lời Giải:** Sử dụng Node.js và Express để xử lý yêu cầu đăng nhập và đăng ký. Lưu thông tin người dùng vào cơ sở dữ liệu (ví dụ: MongoDB) sau khi đã mã hóa mật khẩu.



```

// Import các module cần thiết
const express = require('express');
const bcrypt = require('bcrypt');
const session = require('express-session');

const app = express();

// Middleware để xử lý dữ liệu POST
app.use(express.urlencoded({ extended: true }));

// Middleware session
app.use(session({
  secret: 'secret-key',
  resave: false,
  saveUninitialized: true,
  cookie: { secure: true, maxAge: 60000 } // Thời gian hết hạn của
  // phiên làm việc (60 giây)
}));

// Cơ sở dữ liệu người dùng (giả sử)
const users = [];

// Trang đăng ký
app.get('/register', (req, res) => {
  res.send('<h1>Register Page</h1><form method="POST" action="/register"><input type="text" name="username" placeholder="Username" required><br><input type="password" name="password" placeholder="Password" required><br><button type="submit">Register</button></form>');
});

// Xử lý đăng ký
app.post('/register', (req, res) => {
  const { username, password } = req.body;
  const hashedPassword = bcrypt.hashSync(password, 10);
  users.push({ username, password: hashedPassword });
  res.send('<h1>Registration Successful!</h1>');
});

```

```
// Trang đăng nhập
app.get('/login', (req, res) => {
  res.send(`<h1>Login Page</h1><form method="POST" action="/login"><input type="text" name="username" placeholder="Username" required><br><input type="password" name="password" placeholder="Password" required><br><button type="submit">Login</button></form>`);
});

// Xử lý đăng nhập
app.post('/login', (req, res) => {
  const { username, password } = req.body;
  const user = users.find(user => user.username === username);
  if (user && bcrypt.compareSync(password, user.password)) {
    req.session.user = user;
    res.send(`<h1>Login Successful!</h1>`);
  } else {
    res.send(`<h1>Login Failed!</h1>`);
  }
});

// Start server
app.listen(3000, () => {
  console.log('Server is running on port 3000');
});
```

## 2. Bài Tập 2: Bảo Mật Dữ Liệu

### 2.1. Sử Dụng HTTPS

**Lời Giải:** Cấu hình máy chủ Node.js để hỗ trợ HTTPS bằng cách sử dụng chứng chỉ SSL/TLS. Sử dụng module https trong Node.js.

### 2.2. Băm Mật Khẩu

**Lời Giải:** Sử dụng bcrypt để băm mật khẩu người dùng trước khi lưu vào cơ sở dữ liệu.

```
const bcrypt = require('bcrypt');
const saltRounds = 10;
bcrypt.hash(userPassword, saltRounds, (err, hash) => {
  // Lưu 'hash' vào cơ sở dữ liệu
});
```

## › Lời giải bài tập 2:

```
// ... (Code của Bài Tập 1 ở đây) ...

// Bảo mật yêu cầu HTTP
app.use((req, res, next) => {
  if (!req.secure) {
    res.redirect(`https://${req.headers.host}${req.url}`);
  } else {
    next();
  }
});

// Băm mật khẩu
const hashedPassword = bcrypt.hashSync('userpassword', 10);

// ... (Tiếp tục code của Bài Tập 1 ở đây) ...
```

## 3. Bài Tập 3: Bảo Mật Các Yêu Cầu HTTP

### 3.1. Ngăn Chặn Tấn Công CSRF

**Lời Giải:** Sử dụng token CSRF trong yêu cầu POST và kiểm tra token này trước khi xử lý yêu cầu.

```

const csrf = require('csurf');
const csrfProtection = csrf({ cookie: true });
app.use(csrfProtection);
// ...
app.post('/process', csrfProtection, (req, res) => {
  // Kiểm tra token CSRF trước khi xử lý yêu cầu
});

```

### 3.2. Ngăn Chặn Tấn Công XSS

**Lời Giải:** Sử dụng thư viện như sanitize-html để làm sạch dữ liệu người dùng trước khi hiển thị.

```

const sanitizeHtml = require('sanitize-html');
const cleanedInput = sanitizeHtml(userInput, {
  allowedTags: [],
  allowedAttributes: {}
});

```

▶ Lời giải bài tập 3:

```

// ... (Code của Bài Tập 2 ở đây) ...

// Ngăn chặn tấn công CSRF
const csrf = require('csurf');
const csrfProtection = csrf({ cookie: true });
app.use(csrfProtection);

// Ngăn chặn tấn công XSS
const sanitizeHtml = require('sanitize-html');

```

```

app.post('/process', csrfProtection, (req, res) => {
  // Kiểm tra token CSRF
  if (req.csrfToken() !== req.body._csrf) {
    res.send('<h1>CSRF Attack Detected!</h1>');
    return;
  }

  // Làm sạch dữ liệu người dùng trước khi hiển thị
  const cleanedInput = sanitizeHtml(req.body.userInput, {
    allowedTags: [],
    allowedAttributes: {}
  });

  // Xử lý dữ liệu
  // ...
});

```

### 4. Bài Tập 4: Quản Lý Phiên Làm Việc

#### 4.1. Phiên Làm Việc An Toàn

**Lời Giải:** Sử dụng module express-session để quản lý phiên làm việc và thiết lập thời gian hết hạn cho phiên làm việc.

```

const session = require('express-session');
app.use(session({
  secret: 'secret-key',
  resave: false,
  saveUninitialized: true,
  cookie: { secure: true, maxAge: 60000 } // Thời gian hết hạn của
  // phiên làm việc (60 giây)
}));

```

## 4.2. Xác Thực Hai Yếu Tố (2FA)

**Lời Giải:** Sử dụng thư viện như speakeasy để tạo và xác thực mã OTP (One-Time Password) cho xác thực hai yếu tố.

```
const speakeasy = require('speakeasy');
const secret = speakeeasy.generateSecret({ length: 20 });
// Gửi 'secret.base32' đến người dùng và lưu vào cơ sở dữ liệu
const token = req.body.token;
const verified = speakeeasy.totp.verify({
  secret: storedSecret, // Lấy từ cơ sở dữ liệu
  encoding: 'base32',
  token: token,
});
```

▶ **Lời giải bài tập 4:**

```
// ... (Code của Bài Tập 3 ở đây) ...

// Phiên làm việc an toàn
app.get('/dashboard', (req, res) => {
  if (req.session.user) {
    res.send('<h1>Welcome to Dashboard!</h1>');
  } else {
    res.redirect('/login');
  }
});
// Xác thực hai yếu tố (2FA)
const speakeasy = require('speakeasy');

app.get('/enable-2fa', (req, res) => {
  const secret = speakeeasy.generateSecret({ length: 20 });
  // Lưu 'secret.base32' vào cơ sở dữ liệu cho người dùng
  res.send('<h1>2FA Secret: ${secret.base32}</h1>');
});
```

```
app.post('/verify-2fa', (req, res) => {
  const { token, secret } = req.body;
  const verified = speakeeasy.totp.verify({
    secret: secret,
    encoding: 'base32',
    token: token,
  });
  if (verified) {
    res.send('<h1>2FA Verification Successful!</h1>');
  } else {
    res.send('<h1>2FA Verification Failed!</h1>');
  }
});
// ... (Tiếp tục code của Bài Tập 3 ở đây) ...
```

## 4.3. Gửi Thông Báo Đối Với Hoạt Động Đáng Ngờ

**Lời Giải:** Sử dụng email hoặc thông báo push để gửi thông báo đến người dùng có hoạt động đáng ngờ trên tài khoản của họ.

Những bài tập này giúp bạn áp dụng kiến thức về xác thực và bảo mật trong một ứng dụng web thực tế. Đảm bảo rằng bạn hiểu rõ về từng giải pháp bảo mật và biết cách tích hợp chúng vào ứng dụng của mình. Bạn nên tự điều chỉnh mã nguồn để phù hợp với những dự án thực tế.

## CHƯƠNG VIII

### RESPONSIVE WEB DESIGN VÀ MOBILE DEVELOPMENT

#### THIẾT KẾ ĐÁP ỨNG CHO CÁC THIẾT BỊ DI ĐỘNG VÀ MÀN HÌNH NHỎ

##### ► Lý thuyết

##### 1.1. Viewport Meta Tag

Mã code HTML:

```
<meta name="viewport" content="width=device-width,  
initial-scale=1">
```

##### 1.2. CSS Media Queries

Mã code CSS:

```
@media only screen and (max-width: 768px) {  
    /* CSS for screens smaller than 768px */  
}  
  
@media only screen and (min-width: 769px) {  
    /* CSS for screens larger than or equal to 769px */  
}
```

##### 1.3. Flexible Grid Layout

Sử dụng Flexbox hoặc CSS Grid để tạo bố cục linh hoạt.

#### 1. Responsive Images và Multimedia:

Mã code CSS:

```
img, video {  
    max-width: 100%;  
    height: auto;  
}
```

Nguồn Ví Dụ:

Mã code HTML:

```
<!DOCTYPE html>  
html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1">  
    <title>Responsive Website</title>  
    <link rel="stylesheet" href="styles.css">  
</head>  
<body>  
    <header>  
        <h1>Responsive Header</h1>  
    </header>  
    <main>  
        <section class="content">  
            <h2>Main Content</h2>  
            <p>This is the main content of the website.</p>  
        </section>  
        <aside class="sidebar">  
            <h2>Sidebar</h2>  
            <p>Additional information goes here.</p>  
        </aside>  
    </main>  
    <footer>  
        <p>© 2023 Responsive Website</p>  
    </footer>  
</body>  
</html>
```

Mã code CSS:

```

/* styles.css */
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
}

header, main, footer {
    padding: 20px;
}

@media only screen and (max-width: 768px) {
    header {
        background-color: #333;
        color: white;
        text-align: center;
    }

    .sidebar {
        display: none;
    }
}

@media only screen and (min-width: 769px) {
    header {
        background-color: #F4F4F4;
        color: #333;
    }

    .sidebar {
        display: block;
        float: right;
        width: 30%;
    }
}

```

## BÀI TẬP CƠ BẢN VÀ NÂNG CAO

### 1. Bài tập

#### 1.1. Bài Tập 1: Responsive Navigation Bar

Thiết kế một thanh navigation bar sử dụng Flexbox hoặc CSS Grid để chuyển đổi giữa trạng thái mở và đóng khi kích thước màn hình thay đổi.



#### 1.2. Bài Tập 2: Responsive Image Gallery

Tạo một trang chứa một bộ sưu tập hình ảnh. Sử dụng CSS Grid để hiển thị hình ảnh một cách responsive trên mọi thiết bị.

#### 1.3. Bài Tập 3: Flexible Form Layout

Thiết kế một biểu mẫu đơn đẹp với các trường nhập liệu. Sử dụng Flexbox hoặc CSS Grid để tạo bố cục linh hoạt cho biểu mẫu khi kích thước màn hình thay đổi.

### 2. Lời Giải Bài Tập Thực Hành:

#### 2.1. Bài Tập 1: Responsive Navigation Bar

Mã code HTML:

```

<!-- HTML -->
<header>
    <h1>Responsive Website</h1>
    <nav>
        <div class="menu-icon">☰</div>
        <ul class="menu">
            <li><a href="#">Home</a></li>
            <li><a href="#">About</a></li>
            <li><a href="#">Services</a></li>
            <li><a href="#">Contact</a></li>
        </ul>
    </nav>
</header>

```

```
/* CSS */
.menu-icon {
  display: none;
  cursor: pointer;
}

.menu {
  list-style-type: none;
  display: flex;
  justify-content: space-around;
}

@media only screen and (max-width: 768px) {
  .menu {
    display: none;
  }

  .menu-icon {
    display: block;
  }

  .menu.active {
    display: flex;
  }
}
```



162

## 12. Bài Tập 2: Responsive Image Gallery

Mã code HTML:

```
<!-- HTML -->
<main>
  <section class="image-gallery">
    
    
    
    <!-- More images... -->
  </section>
</main>
```

Mã code CSS:

```
/* CSS */
.image-gallery {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(200px, 1fr));
  gap: 10px;
  justify-content: center;
}

img {
  width: 100%;
  height: auto;
}
```



163

```
<!-- HTML -->
<main>
  <form class="form">
    <div class="form-group">
      <label for="name">Name:</label>
      <input type="text" id="name" name="name"
required>
    </div>
    <div class="form-group">
      <label for="email">Email:</label>
      <input type="email" id="email" name="email"
required>
    </div>
    <div class="form-group">
      <label for="message">Message:</label>
      <textarea id="message" name="message" required>
</textarea>
    </div>
    <button type="submit">Submit</button>
  </form>
</main>
```



164

```
/* CSS */
.form {
  display: grid;
  grid-template-columns: 1fr;
  gap: 10px;
  max-width: 400px;
  margin: 0 auto;
}

.form-group {
  display: flex;
  flex-direction: column;
}

label {
  margin-bottom: 5px;
}

input, textarea {
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 5px;
}

button {
  padding: 10px 20px;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 5px;
  cursor: pointer;
}

button:hover {
  background-color: #0056b3;
}
```

Những ví dụ và bài tập này sẽ giúp bạn hiểu rõ về thiết kế đáp ứng từ cơ bản đến chi tiết chuyên sâu và cách áp dụng nó vào dự án của mình. Đừng ngần ngại thử nghiệm và tinh chỉnh mã nguồn để hiểu rõ hơn về cách nó hoạt động.

## SỬ DỤNG BOOTSTRAP VÀ MEDIA QUERIES

### ► Khái niệm

**Bootstrap:** Bootstrap là một framework CSS và JavaScript phổ biến dùng để thiết kế giao diện web responsive và hiện đại. Nó cung cấp các lớp CSS, component và plugins giúp tạo ra giao diện thân thiện với người dùng một cách nhanh chóng.

**Media Queries:** Media Queries là một công cụ trong CSS cho phép bạn định nghĩa các luật CSS dựa trên các điều kiện nhất định như kích thước màn hình, thiết bị và phương thức in ấn. Điều này giúp tối ưu hóa trải nghiệm người dùng trên nhiều thiết bị khác nhau.

### ► Cấu Trúc Câu Lệnh và Ví Dụ Thực Tế

| Cấp độ   | Kiến Thức và Cú Pháp Bootstrap   | Ví Dụ   |
|----------|--|---|
| Cơ bản   | <ul style="list-style-type: none"> <li>- Grid System (Hệ thống lưới)</li> <li>- Typography (Kiểu chữ)</li> <li>- Colors (Màu sắc)</li> <li>- Buttons (Nút)</li> </ul>    | <pre>&lt;div class="container"&gt;&lt;div class="row"&gt;&lt;div class="col-md-6"&gt;Content&lt;/div&gt;&lt;/div&gt;&lt;/div&gt;</pre>                  |
|          | <ul style="list-style-type: none"> <li>- Forms (Biểu mẫu)</li> <li>- Alerts (Thông báo)</li> <li>- Navbar (Thanh điều hướng)</li> <li>- Modals (Cửa sổ popup)</li> </ul> | <pre>&lt;form&gt;&lt;input type="text" class="form-control"&gt;&lt;button class="btn btn-primary"&gt;Submit&lt;/button&gt;&lt;/form&gt;</pre>           |
| Nâng cao | <ul style="list-style-type: none"> <li>- Carousel (Hình ảnh trượt)</li> <li>- Tabs (Thẻ chuyển đổi nội dung)</li> </ul>  | <pre>&lt;div id="carouselExample" class="carousel slide" data-bs-ride="carousel"&gt;&lt;div class="carousel-inner"&gt;...&lt;/div&gt;&lt;/div&gt;</pre> |

| Cấp độ     | Kiến Thức và Cú Pháp Bootstrap  | Ví Dụ   |
|------------|---|---|
|            | <ul style="list-style-type: none"> <li>- Accordion (Thẻ mở rộng)</li> <li>- Collapse (Thu gọn nội dung)</li> </ul>  |   |
|            | <ul style="list-style-type: none"> <li>- Tooltip (Hộp chú thích)</li> <li>- Popover (Hộp thông báo)</li> <li>- Scrollspy (Theo dõi cuộc cuộc)</li> <li>- Spinners (Chỉ mục quay)</li> </ul> | <pre>&lt;button data-bs-toggle="tooltip" title="Tooltip Text"&gt;Hover Me&lt;/button&gt;</pre>  |
| Chuyên sâu | <ul style="list-style-type: none"> <li>- Responsive Utilities (Tiện ích phản hồi)</li> <li>- Custom CSS (Tùy chỉnh CSS)</li> <li>- Themes (Chủ đề)</li> </ul>                               | <pre>&lt;div class="d-none d-md-block"&gt;Visible on medium and larger screens&lt;/div&gt;</pre>  |
|            | <ul style="list-style-type: none"> <li>- Form Validation (Xác nhận biểu mẫu)</li> <li>- Utility Classes (Lớp tiện ích)</li> <li>- Flexbox (Hộp đàn hồi)</li> </ul>                          | <pre>&lt;input type="email" class="form-control is-invalid" required&gt;&lt;div class="invalid-feedback"&gt;Invalid Email&lt;/div&gt;</pre> |

### ► Sử Dụng Bootstrap:

#### Mã code HTML:

```

<!-- Link Bootstrap CSS từ CDN --&gt;
&lt;link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css"&gt;

<!-- Sử dụng container để căn giữa và responsive --&gt;
&lt;div class="container"&gt;
  &lt;!-- Sử dụng lớp CSS của Bootstrap --&gt;
  &lt;div class="alert alert-primary" role="alert"&gt;
    Đây là một thông báo Bootstrap!
  &lt;/div&gt;
&lt;/div&gt;

<!-- Link Bootstrap JavaScript và Popper.js từ CDN --&gt;
&lt;script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"&gt;
&lt;/script&gt;
</pre>

```

## ▶ Sử Dụng Media Queries

Mã code CSS:

```
/* Định nghĩa luật CSS cho màn hình có kích thước nhỏ hơn hoặc bằng 768px */
@media (max-width: 768px) {
    /* CSS cho màn hình nhỏ */
    body {
        background-color: lightblue;
    }
}

/* Định nghĩa luật CSS cho màn hình có kích thước từ 769px đến 992px */
@media (min-width: 769px) and (max-width: 992px) {
    /* CSS cho màn hình trung bình */
    body {
        background-color: lightgreen;
    }
}
```



## ▶ Bootstrap nâng cao

dưới đây là một số kỹ thuật Bootstrap nâng cao với các ví dụ chi tiết:

## 5.1. Thiết Lập Một Navbar Tùy Chỉnh

Mã code HTML:

```
<nav class="navbar navbar-expand-lg navbar-dark bg-primary">
    <a class="navbar-brand" href="#">Logo</a>
    <button class="navbar-toggler" type="button" data-
        toggle="collapse" data-target="#navbarNav" aria-
        controls="navbarNav" aria-expanded="false" aria-label="Toggle
        navigation">
        <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
        <ul class="navbar-nav ml-auto">
            <li class="nav-item active">
                <a class="nav-link" href="#">Home</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">About</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Services</a>
            </li>
            <li class="nav-item">
                <a class="nav-link" href="#">Contact</a>
            </li>
        </ul>
    </div>
</nav>
```

## 5.2. Tạo Carousel (Trình Chiếu Ảnh Tự Động Chuyển Động)

Mã code HTML:

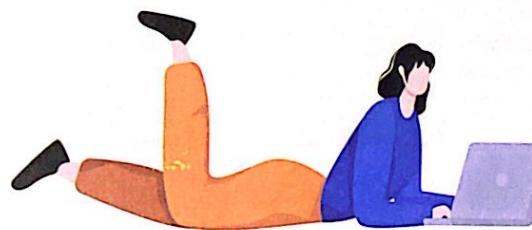
```
<div id="carouselExampleIndicators" class="carousel slide" data-ride="carousel">
  <ol class="carousel-indicators">
    <li data-target="#carouselExampleIndicators" data-slide-to="0" class="active"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="1"></li>
    <li data-target="#carouselExampleIndicators" data-slide-to="2"></li>
  </ol>
  <div class="carousel-inner">
    <div class="carousel-item active">
      
    </div>
    <div class="carousel-item">
      
    </div>
    <div class="carousel-item">
      
    </div>
  </div>
  <a class="carousel-control-prev" href="#carouselExampleIndicators" role="button" data-slide="prev">
    <span class="carousel-control-prev-icon" aria-hidden="true"></span>
    <span class="sr-only">Previous</span>
  </a>
  <a class="carousel-control-next" href="#carouselExampleIndicators" role="button" data-slide="next">
    <span class="carousel-control-next-icon" aria-hidden="true"></span>
    <span class="sr-only">Next</span>
  </a>
</div>
```

## 5.3. Tạo Modal (Cửa Sổ Hiển Thị Popup)

Mã code HTML:

```
<!-- Button trigger modal -->
<button type="button" class="btn btn-primary" data-toggle="modal" data-target="#myModal">
  Open Modal
</button>

<!-- Modal -->
<div class="modal fade" id="myModal" tabindex="-1" role="dialog" aria-labelledby="exampleModalLabel" aria-hidden="true">
  <div class="modal-dialog" role="document">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="exampleModalLabel">Modal Title</h5>
        <button type="button" class="close" data-dismiss="modal" aria-label="Close">
          <span aria-hidden="true">&times;</span>
        </button>
      </div>
      <div class="modal-body">
        Content goes here.
      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-dismiss="modal">Close</button>
        <button type="button" class="btn btn-primary">Save changes</button>
      </div>
    </div>
  </div>
</div>
```



```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bootstrap Example</title>
    <link rel="stylesheet"
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
    <style>
        /* CSS cho màn hình lớn */
        @media (min-width: 992px) {
            body {
                background-color: lightblue;
            }
        }

        /* CSS cho màn hình nhỏ */
        @media (max-width: 991px) {
            body {
                background-color: lightgreen;
            }
        }
    </style>
</head>

<body>
    <div class="container">
        <h1>Danh Sách Sản Phẩm</h1>
        <ul class="list-group">
            <li class="list-group-item">Sản phẩm 1</li>
            <li class="list-group-item">Sản phẩm 2</li>
            <li class="list-group-item">Sản phẩm 3</li>
        </ul>
    </div>

    <script
        src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js">
    </script>
</body>

</html>
```

Trong ví dụ này, trang web sử dụng **Bootstrap** để hiển thị danh sách sản phẩm trong một container. **CSS** được định nghĩa bằng **Media Queries** để thay đổi màu nền của trang tùy thuộc vào kích thước màn hình.

## 2. Bài tập chuyên sâu về Bootstrap:

### 2.1. Bài Tập 1: Tạo Trang Chủ Website Công Ty

Tạo một trang chủ cho một công ty với các phần sau:

- Header với menu điều hướng và logo.
- Phần giới thiệu công ty với hình ảnh và mô tả ngắn.
- Danh sách sản phẩm hoặc dịch vụ của công ty sử dụng Cards.
- Phần liên hệ với biểu mẫu liên hệ.

Lời giải

Mã code HTML:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Website Công Ty</title>
    <link rel="stylesheet"
        href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>

<body>
    <header class="bg-dark text-white p-3">
        <div class="container">
            <nav class="navbar navbar-expand-lg navbar-dark">
                <a class="navbar-brand" href="#">Logo</a>
                <button class="navbar-toggler" type="button" data-bs-
                    toggle="collapse"
                    data-bs-target="#navbarNav" aria-controls="navbarNav" aria-
                    expanded="false">
```

```
aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
</button>
<div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav ms-auto">
        <li class="nav-item">
            <a class="nav-link" href="#">Trang Chủ</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="#">Sản Phẩm</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="#">Dịch Vụ</a>
        </li>
        <li class="nav-item">
            <a class="nav-link" href="#">Liên Hệ</a>
        </li>
    </ul>
</div>
</nav>
</div>
</header>

<section class="py-5">
    <div class="container">
        <div class="row">
            <div class="col-md-6">
                
            </div>
            <div class="col-md-6">
                <h2>Giới Thiệu Công Ty XYZ</h2>
                <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit. Nullam in ante eu tellus convallis convallis. Nulla vestibulum purus ut felis commodo, et blandit lectus aliquet.</p>
            </div>
        </div>
    </div>
</section>
```

```

<section class="py-5 bg-light">
    <div class="container">
        <h2 class="text-center mb-4">Sản Phẩm và Dịch Vụ</h2>
        <div class="row">
            <div class="col-md-4">
                <div class="card">
                    
                    <div class="card-body">
                        <h5 class="card-title">Sản Phẩm 1</h5>
                        <p class="card-text">Mô tả sản phẩm 1.</p>
                    </div>
                </div>
            <div class="col-md-4">
                <div class="card">
                    
                    <div class="card-body">
                        <h5 class="card-title">Sản Phẩm 2</h5>
                        <p class="card-text">Mô tả sản phẩm 2.</p>
                    </div>
                </div>
            <div class="col-md-4">
                <div class="card">
                    
                    <div class="card-body">
                        <h5 class="card-title">Dịch Vụ 1</h5>
                        <p class="card-text">Mô tả dịch vụ 1.</p>
                    </div>
                </div>
            </div>
        </div>
    </div>
</section>

```

```

<section class="py-5">
    <div class="container">
        <h2 class="text-center mb-4">Liên Hệ</h2>
        <form>
            <div class="mb-3">
                <label for="name" class="form-label">Họ và Tên</label>
                <input type="text" class="form-control" id="name">
            </div>
            <div class="mb-3">
                <label for="email" class="form-label">Email</label>
                <input type="email" class="form-control" id="email">
            </div>
            <div class="mb-3">
                <label for="message" class="form-label">Nội Dung</label>
                <textarea class="form-control" id="message" rows="3"></textarea>
            </div>
            <button type="submit" class="btn btn-primary">Gửi Liên Hệ</button>
        </form>
    </div>
</section>

<script
src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js">
</script>
</body>
</html>

```



## 2.2. Bài Tập 2: Tạo Trang Danh Sách Sản Phẩm với Pagination

Tạo một trang danh sách sản phẩm với phân trang sử dụng Bootstrap Pagination.

Lời giải

Mã code HTML:

```
<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Danh Sách Sản Phẩm</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/css/bootstrap.min.css">
</head>

<body>
    <div class="container mt-5">
        <h2 class="mb-4">Danh Sách Sản Phẩm</h2>
        <ul class="list-group">
            <li class="list-group-item">Sản Phẩm 1</li>
            <li class="list-group-item">Sản Phẩm 2</li>
            <!-- Thêm danh sách sản phẩm -->
            <li class="list-group-item">Sản Phẩm 3</li>
            <li class="list-group-item">Sản Phẩm 4</li>
            <li class="list-group-item">Sản Phẩm 5</li>
            <!-- Thêm danh sách sản phẩm -->
        </ul>
        <!-- Phân trang -->
        <nav class="mt-4">
            <ul class="pagination justify-content-center">
                <li class="page-item disabled">
                    <a class="page-link" href="#" tabindex="-1" aria-disabled="true">Trang trước</a>
                </li>
                <li class="page-item active" aria-current="page">
                    <a class="page-link" href="#">1 <span class="sr-only">(current)</span></a>
                </li>
                <li class="page-item"><a class="page-link" href="#">2</a></li>
                <li class="page-item"><a class="page-link" href="#">3</a></li>
                <li class="page-item">
                    <a class="page-link" href="#">Trang sau</a>
                </li>
            </ul>
        </nav>
    </div>

    <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

Lưu ý: Trong ví dụ này, chỉ có hai trang sản phẩm được hiển thị. Bạn có thể mở rộng danh sách sản phẩm và cấu hình phân trang theo ý muốn. Hãy tham khảo tài liệu chính thức của Bootstrap để tìm hiểu thêm về cách sử dụng Pagination và các tùy chọn khác của Bootstrap.

## PHÁT TRIỂN ỨNG DỤNG DI ĐỘNG VỚI REACT NATIVE

### ► Giới Thiệu Về React Native

React Native là một framework phát triển ứng dụng di động được xây dựng trên nền tảng React, một thư viện JavaScript phổ biến để xây dựng giao diện người dùng. Sử dụng React Native, bạn có thể phát triển ứng dụng di động đa nền tảng với một lượng lớn mã nguồn chung giữa iOS và Android.

### ► Lý thuyết React Native

#### 1. Components và Props:

Trong React Native, bạn xây dựng các thành phần (components) để tạo giao diện người dùng. Các thành phần có thể nhận các thuộc tính (props) để truyền dữ liệu giữa chúng.

#### 2. State:

React Native sử dụng state để lưu trữ và quản lý trạng thái của thành phần. Khi state thay đổi, giao diện người dùng cũng được cập nhật tương ứng.

#### 3. React component:

React Native có các phương thức vòng đời (lifecycle methods) như componentDidMount, componentDidUpdate, và componentWillUnmount giúp bạn quản lý các thao tác khi thành phần được tạo, cập nhật hoặc hủy bỏ.

#### 4. Styles và Layout:

React Native sử dụng các kiểu dạng (styles) để định dạng giao diện người dùng. Các kiểu dạng được thiết kế để tương tự CSS, nhưng sử dụng JavaScript object.

## 2.5. Navigation:

Phát triển ứng dụng di động đòi hỏi quản lý điều hướng (navigation) giữa các màn hình. React Navigation là một thư viện phổ biến để quản lý điều hướng trong React Native.

### ► Cú Pháp Câu Lệnh React Native

#### 3.1. Tạo Một Thành Phần (Component):

*javascript*

```
import React from 'react';
import { View, Text, StyleSheet } from 'react-native';

const MyComponent = () => {
  return (
    <View style={styles.container}>
      <Text>Hello, React Native!</Text>
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
});

export default MyComponent;
```



#### 3.2. Sử Dụng Props:

*javascript*

```
const MyComponent = (props) => {
  return (
    <View style={styles.container}>
      <Text>Hello, {props.name}!</Text>
    </View>
  );
};
```

#### 3.3. Sử Dụng State:

*javascript*

```
import React, { useState } from 'react';
import { View, Button, Text, StyleSheet } from 'react-native';

const Counter = () => {
  const [count, setCount] = useState(0);

  const increaseCount = () => {
    setCount(count + 1);
  };

  return (
    <View style={styles.container}>
      <Text>Count: {count}</Text>
      <Button title="Increase Count" onPress={increaseCount} />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
});

export default Counter;
```



► Bài Tập Thực Hành với Lời Giải:

#### 4.1. Bài Tập 1: Tạo Ứng Dụng Hiển Thị Danh Sách Sản Phẩm

- ✓ Tạo một danh sách sản phẩm với tên và giá.
- ✓ Hiển thị danh sách sản phẩm trong ứng dụng React Native.
- ✓ Khi người dùng nhấn vào sản phẩm, hiển thị thông tin chi tiết của sản phẩm đó.

Lời giải

**Mã code: javascript**

```
// ProductList.js
import React from 'react';
import { View, Text, FlatList, TouchableOpacity, StyleSheet } from 'react-native';

const ProductList = ({ products, onProductPress }) => {
  return (
    <FlatList
      data={products}
      renderItem={({ item }) => (
        <TouchableOpacity onPress={() => onProductPress(item)}>
          <View style={styles.productItem}>
            <Text>{item.name}</Text>
            <Text>${item.price}</Text>
          </View>
        </TouchableOpacity>
      )}
      keyExtractor={(item) => item.id.toString()}
    />
  );
}

const styles = StyleSheet.create({
  productItem: {
    flexDirection: 'row',
    justifyContent: 'space-between',
    padding: 10,
    borderBottomWidth: 1,
    borderBottomColor: '#ccc',
  },
});

export default ProductList;
```

**Mã code: javascript**

```
// ProductDetail.js
import React from 'react';
import { View, Text } from 'react-native';

const ProductDetail = ({ route }) => {
  const { product } = route.params;

  return (
    <View>
      <Text>Name: {product.name}</Text>
      <Text>Price: ${product.price}</Text>
      <Text>Description: {product.description}</Text>
    </View>
  );
};

export default ProductDetail;
```

**Mã code: javascript**

```
// App.js
import React from 'react';
import { NavigationContainer } from '@react-navigation/native';
import { createStackNavigator } from '@react-navigation/stack';
import ProductList from './ProductList';
import ProductDetail from './ProductDetail';

const Stack = createStackNavigator();

const products = [
  { id: 1, name: 'Product 1', price: 100, description: 'Description of Product 1' },
  { id: 2, name: 'Product 2', price: 150, description: 'Description of Product 2' },
  // Add more products as needed
];


```

```

const App = () => {
  return (
    <NavigationContainer>
      <Stack.Navigator initialRouteName="ProductList">
        <Stack.Screen
          name="ProductList"
          component={ProductList}
          options={{ title: 'Product List' }}
          initialParams={{ products }}
        />
        <Stack.Screen
          name="ProductDetail"
          component={ProductDetail}
          options={{ title: 'Product Detail' }}
        />
      </Stack.Navigator>
    </NavigationContainer>
  );
};

export default App;

```

Trong ví dụ này, `ProductList` là một danh sách sản phẩm, và khi người dùng nhấn vào sản phẩm, họ sẽ được chuyển đến màn hình `ProductDetail` để xem chi tiết sản phẩm đó.

*Hy vọng rằng ví dụ và bài tập này giúp bạn hiểu rõ hơn về cách phát triển ứng dụng di động với React Native. Đừng ngần ngại thử nghiệm và tùy chỉnh mã nguồn để hiểu rõ hơn về cách nó hoạt động của nó nhé!*

## LẬP TRÌNH WEBSITE VỚI MOBILE DEVELOPMENT

### ► Khái Niệm:

Mobile Development là quá trình xây dựng ứng dụng hoặc trang web được tối ưu hóa để chạy trên các thiết bị di động như điện thoại thông minh và máy tính bảng. Nó liên quan đến việc sử dụng các ngôn ngữ lập trình và framework để tạo ra trải nghiệm người dùng tốt trên các thiết bị có kích thước màn hình khác nhau.

### ► Ngôn Ngữ Sử Dụng:

#### 2.1. HTML, CSS, và JavaScript:

HTML (HyperText Markup Language) định cấu trúc nội dung trang web.

CSS (Cascading Style Sheets) quản lý thiết kế và giao diện.

JavaScript cung cấp các chức năng tương tác và xử lý sự kiện trên trình duyệt.

#### 2.2. Frameworks và Thư Viện:

React Native: Cho phép xây dựng ứng dụng di động sử dụng React và JavaScript.

Flutter: Framework phát triển ứng dụng di động đa nền tảng sử dụng ngôn ngữ Dart.

Ionic: Sử dụng HTML, CSS và JavaScript để xây dựng ứng dụng di động và web.

### ► Cú Pháp Câu Lệnh và Ví Dụ Thực Tế:

#### 3.1. HTML:

Mã HTML:

```
<!DOCTYPE html>
<html>
<head>
    <title>My Mobile App</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Hello, Mobile World!</h1>
    <button onclick="showAlert()">Click Me</button>

    <script>
        function showAlert() {
            alert('Button Clicked!');
        }
    </script>
</body>
</html>
```

### 3.2. CSS:

Mã CSS:

```
body {
    font-family: Arial, sans-serif;
}

h1 {
    color: blue;
}

button {
    background-color: green;
    color: white;
    padding: 10px 20px;
    border: none;
    cursor: pointer;
}

button:hover {
    background-color: darkgreen;
}
```

### 3. JavaScript (React Native):

Mã code: jsx

```
import React, { useState } from 'react';
import { View, Text, Button, Alert } from 'react-native';

const App = () => {
    const [message, setMessage] = useState('Hello, Mobile World!');

    const showAlert = () => {
        Alert.alert('Button Clicked!');
    };

    return (
        <View style={{ flex: 1, justifyContent: 'center', alignItems: 'center' }}>
            <Text>{message}</Text>
            <Button title="Click Me" onPress={showAlert} />
        </View>
    );
};

export default App;
```

► Bài Tập Ứng Dụng và Lời Giải Chi Tiết:

#### 4.1. Bài Tập:

Xây dựng ứng dụng di động đơn giản hiển thị danh sách sản phẩm. Mỗi sản phẩm có tên, giá và mô tả.

#### 4.2. Lời Giải Chi Tiết:

jsx



```

import React from 'react';
import { View, Text, FlatList, StyleSheet } from 'react-native';

const products = [
  { id: '1', name: 'Product 1', price: '$10', description: 'Description 1' },
  { id: '2', name: 'Product 2', price: '$20', description: 'Description 2' },
  { id: '3', name: 'Product 3', price: '$30', description: 'Description 3' },
  // ...more products
];

const ProductScreen = () => {
  const renderItem = ({ item }) => (
    <View style={styles.item}>
      <Text style={styles.title}>{item.name}</Text>
      <Text>{item.price}</Text>
      <Text>{item.description}</Text>
    </View>
  );

  return (
    <View style={styles.container}>
      <FlatList
        data={products}
        renderItem={renderItem}
        keyExtractor={item => item.id}
      />
    </View>
  );
};

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 16,
    justifyContent: 'center',
  },
  item: {
    borderBottomWidth: 1,
    borderColor: '#ccc',
    padding: 8,
  },
  title: {
    fontSize: 18,
    fontWeight: 'bold',
  },
});

```

export default ProductScreen;

Trong ví dụ này, chúng ta sử dụng React Native để xây dựng một danh sách sản phẩm đơn giản và hiển thị chúng trên màn hình di động.

▶ Bài tập thực hành từ cơ bản đến chuyên sâu về lập trình website với Mobile

## Bài Tập Cơ Bản

### 5.1.1 | Xây dựng Trang Chào Mừng

- Tạo một trang chào mừng sử dụng HTML, CSS để thiết kế giao diện.
- Thêm hiệu ứng hover cho các nút và hình ảnh.

### 5.1.2 | Tạo Danh Sách Sản Phẩm

- Tạo một danh sách sản phẩm với thông tin như tên, giá, và mô tả sử dụng HTML và CSS.
- Sử dụng JavaScript để tạo chức năng hiển thị/ẩn thông tin sản phẩm khi người dùng nhấp vào.

### 5.1.3 | Ứng Dụng Quản Lý Công Việc

- Xây dựng một ứng dụng quản lý công việc (to-do list) với các chức năng thêm, sửa, xóa công việc.
- Sử dụng React Native hoặc Flutter để tạo giao diện đẹp và tương tác linh hoạt.

### 5.1.4 | Ứng Dụng Chat

- Phát triển một ứng dụng chat đơn giản giữa hai người dùng.
- Sử dụng WebSocket hoặc Firebase Realtime Database để cập nhật tin nhắn realtime.

### 5.1.5 | Ứng Dụng Thời Tiết

- Kết hợp với API dự báo thời tiết để hiển thị thông tin thời tiết dựa trên vị trí hiện tại của người dùng.
- Sử dụng các biểu đồ hoặc hình ảnh động để thể hiện dữ liệu thời tiết.

### 5.1.6 Ứng Dụng Đọc Tin Tức

Liên kết với API tin tức để hiển thị các tin tức theo chủ đề hoặc từ nguồn thẻ.

Cho phép người dùng tìm kiếm và lọc tin tức theo từ khóa hoặc thể loại.

Lời giải cho các bài tập ứng dụng trên

### 5.2.1 Xây dựng Trang Chào Mừng

```
HTML:
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Welcome Page</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <div class="container">
        <h1>Welcome to Our Website!</h1>
        <button class="hover-effect">Click Me</button>
    </div>
</body>
</html>

CSS (styles.css):
body {
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    margin: 0;
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
}

.container {
    text-align: center;
}

.hover-effect:hover {
    background-color: #4CAF50;
    color: white;
}
```

### 5.2.2 Tạo Danh Sách Sản Phẩm

```
HTML:
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Product List</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <ul class="product-list">
        <li class="product">Product 1 - $10</li>
        <li class="product">Product 2 - $20</li>
        <li class="product">Product 3 - $30</li>
    </ul>
</body>
</html>

CSS (styles.css):
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
    background-color: #f0f0f0;
}

.product-list {
    list-style-type: none;
    padding: 0;
}

.product {
    background-color: white;
    border: 1px solid #ccc;
    margin: 5px;
    padding: 10px;
    cursor: pointer;
}

.product:hover {
    background-color: #f0f0f0;
}
```

```
JavaScript (script.js):
function toggleProductDetails(element) {
    const productDetails = document.querySelector('.product-details');
    const productDescription = document.getElementById('product-description');

    productDescription.textContent = 'Details for ${element.textContent}';
    productDetails.style.display = 'block';
}
```

### 5.2.3 | Ứng Dụng Quản Lý Công Việc (React Native)

```
import React, { useState } from 'react';
import { View, TextInput, Button, FlatList, Text, StyleSheet } from 'react-native';

const App = () => {
    const [task, setTask] = useState('');
    const [tasks, setTasks] = useState([]);

    const addTask = () => {
        setTasks([...tasks, task]);
        setTask('');
    };

    return (
        <View style={styles.container}>
            <TextInput
                style={styles.input}
                placeholder="Enter a task"
                value={task}
                onChangeText={setTask}
            />
            <Button title="Add Task" onPress={addTask} />
            <FlatList
                data={tasks}
                renderItem={({ item }) => <Text style={styles.task}>{item}</Text>}
                keyExtractor={(item, index) => index.toString()}
            />
        </View>
    );
};

export default App;
```

```
const styles = StyleSheet.create({
    container: {
        flex: 1,
        padding: 20,
        backgroundColor: '#fff',
    },
    input: {
        height: 40,
        borderColor: 'gray',
        borderWidth: 1,
        marginBottom: 10,
        paddingHorizontal: 10,
    },
    task: {
        fontSize: 18,
        marginTop: 10,
    },
});

export default App;
```

### 5.2.4 | Ứng Dụng Chat (Firebase Realtime Database)

Để sử dụng Firebase Realtime Database trong ứng dụng React Native, bạn cần thiết lập Firebase và import thư viện firebase vào dự án của bạn. Sau đó, bạn có thể sử dụng các chức năng của Firebase để lưu và đồng bộ dữ liệu tin nhắn giữa các người dùng.

#### Firebase Realtime Database Thiết Lập

- Đầu tiên, bạn cần tạo một dự án Firebase trên trang web chính thức của Firebase.
- Thiết lập Firebase Realtime Database trong mục "Database" của dự án Firebase của bạn. Để bắt đầu, bạn có thể thiết lập quy tắc đơn giản để cho phép tất cả mọi người đọc và ghi dữ liệu.
- Sau đó, bạn cần nhúng Firebase SDK vào trang HTML của bạn.

## File HTML:

```
<!-- Thêm thư viện Firebase -->
<script src="https://www.gstatic.com/firebasejs/9.6.1.firebaseio-app.js"></script>
<script src="https://www.gstatic.com/firebasejs/9.6.1/firebase-database.js">
</script>

<!-- Cấu hình Firebase -->
<script>
  var firebaseConfig = {
    apiKey: "YOUR_API_KEY",
    authDomain: "YOUR_AUTH_DOMAIN",
    databaseURL: "YOUR_DATABASE_URL",
    projectId: "YOUR_PROJECT_ID",
    storageBucket: "YOUR_STORAGE_BUCKET",
    messagingSenderId: "YOUR_MESSAGING_SENDER_ID",
    appId: "YOUR_APP_ID"
  };

  // Khởi tạo Firebase
  firebase.initializeApp(firebaseConfig);

  // Lấy tham chiếu đến cơ sở dữ liệu
  var database = firebase.database();
</script>
```

## Gửi và Nhận Tin Nhắn:

## File javascript

```
// Lấy tham chiếu đến nút trong cơ sở dữ liệu Firebase
var messagesRef = database.ref('messages');

// Gửi tin nhắn
function sendMessage() {
  var message = document.getElementById('message').value;
  var username = document.getElementById('username').value;
```

```
// Gửi tin nhắn đến cơ sở dữ liệu
messagesRef.push({
  message: message,
  username: username,
  timestamp: firebase.database.ServerValue.TIMESTAMP
});

// Xóa nội dung tin nhắn sau khi gửi
document.getElementById('message').value = '';
}

// Lắng nghe sự kiện khi có tin nhắn mới được thêm vào cơ sở dữ liệu
messagesRef.on('child_added', function(data) {
  var message = data.val();
  displayMessage(message);
});

// Hiển thị tin nhắn trên giao diện
function displayMessage(message) {
  var messageDiv = document.createElement('div');
  messageDiv.textContent = message.username + ': ' + message.message;
  document.getElementById('messages-container').appendChild(messageDiv);
}
```

## Giao Diện HTML:

```
<!-- Giao diện HTML -->
<div id="messages-container"></div>
<input type="text" id="username" placeholder="Tên người dùng">
<input type="text" id="message" placeholder="Nhập tin nhắn">
<button onclick="sendMessage()">Gửi</button>
```

Trong ví dụ này, mỗi khi người dùng gửi tin nhắn, tin nhắn sẽ được thêm vào cơ sở dữ liệu Firebase và tự động hiển thị trên giao diện của tất cả người dùng đang kết nối.



```

// Đây chỉ là ví dụ cơ bản, bạn cần sử dụng một API thời tiết thực tế để lấy dữ liệu.
import React, { useState, useEffect } from 'react';
import { View, Text, StyleSheet } from 'react-native';

const WeatherApp = () => {
  const [weather, setWeather] = useState(null);

  useEffect(() => {
    // Gọi API thời tiết và cập nhật state weather.
    // Cần sử dụng một API thời tiết thực tế để lấy dữ liệu.
    // Ví dụ: fetch('API_URL').then(response => response.json()).then(data =>
    setWeather(data));
  }, []);

  return (
    <View style={styles.container}>
      {weather ? (
        <View>
          <Text style={styles.weatherText}>{weather.temperature}°C</Text>
          <Text style={styles.weatherText}>{weather.description}</Text>
        </View>
      ) : (
        <Text>Loading...</Text>
      )}
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    justifyContent: 'center',
    alignItems: 'center',
  },
  weatherText: {
    fontSize: 24,
    marginBottom: 18,
  },
});

export default WeatherApp;

```

196

```

// Đây chỉ là ví dụ cơ bản, bạn cần sử dụng một API tin tức thực tế để lấy dữ liệu.
import React, { useState, useEffect } from 'react';
import { View, Text, FlatList, StyleSheet } from 'react-native';

const NewsApp = () => {
  const [news, setNews] = useState([]);

  useEffect(() => {
    // Gọi API tin tức và cập nhật state news.
    // Cần sử dụng một API tin tức thực tế để lấy dữ liệu.
    // Ví dụ: fetch('API_URL').then(response => response.json()).then(data =>
    setNews(data.articles));
  }, []);

  return (
    <View style={styles.container}>
      <FlatList
        data={news}
        renderItem={({ item }) => (
          <View style={styles.newsItem}>
            <Text style={styles.newsTitle}>{item.title}</Text>
            <Text style={styles.newsDescription}>{item.description}</Text>
          </View>
        )}
        keyExtractor={(item, index) => index.toString()}
      />
    </View>
  );
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    padding: 20,
    backgroundColor: '#fff',
  },
  newsItem: {
    marginBottom: 20,
  },
  newsTitle: {
    fontSize: 18,
    fontWeight: 'bold',
  },
  newsDescription: {
    marginTop: 10,
  },
});

export default NewsApp;

```

197

**Lưu ý:** Trong các ứng dụng React Native, bạn cần sử dụng các thư viện như axios hoặc fetch để gọi API từ server. Đồng thời, đảm bảo rằng bạn đã thiết lập và cấu hình Firebase Realtime Database đúng cách để sử dụng trong ứng dụng chat.

- Khi lập trình website cho thiết bị di động, có một số lưu ý quan trọng mà bạn nên xem xét để đảm bảo rằng trang web của bạn hoạt động mượt mà và tương thích trên các thiết bị di động khác nhau:

### Responsive Design

Thiết kế giao diện sao cho nó phản ánh tốt trên các thiết bị di động khác nhau. Sử dụng media queries để điều chỉnh giao diện dựa trên kích thước màn hình.

### Tối Ưu Hóa Hình Ảnh và Đa Phương Tiện

Sử dụng hình ảnh có độ phân giải thấp hơn để giảm dung lượng tải xuống. Cân nhắc việc sử dụng định dạng hình ảnh dạng WebP để giảm kích thước tệp.

### Giảm Tiêu Hao Tài Nguyên

Tránh sử dụng quá nhiều hiệu ứng đồ họa hoặc video phức tạp để tránh làm giảm hiệu suất trang web.

### Tối Ưu Hóa CSS và JavaScript

Gộp và nén các tệp CSS và JavaScript để giảm số lượng yêu cầu và dung lượng tải xuống.

### Kiểm Tra Trên Nhiều Trình Duyệt và Thiết Bị

Kiểm tra trang web của bạn trên các trình duyệt di động khác nhau (Chrome, Safari, Firefox) và trên các thiết bị với các kích thước màn hình khác nhau.

### Kiểm Thủ Tốc Độ Tải Trang

Sử dụng công cụ như Google PageSpeed Insights để đánh giá tốc độ tải trang và tối ưu hóa dựa trên gợi ý.

### Hạn Chế Sử Dụng Pop-up và Quảng Cáo

Nếu bạn cần sử dụng pop-up hoặc quảng cáo, đảm bảo chúng không gây cản trở cho trải nghiệm người dùng trên thiết bị di động.

### Xử Lý Cẩn Thận Trên Các Thiết Bị Có Kích Thước Nhỏ

Đảm bảo rằng các phần tử như nút và hình ảnh đủ lớn để được nhấp hoặc nhìn rõ trên các thiết bị di động với màn hình nhỏ.



### Tối Ưu Hóa Vùng Nhấp và Tương Tác

Xác định kích thước vùng nhấp cho các phần tử tương tác như nút và liên kết để chúng dễ nhấp trên màn hình cảm ứng.

### Sử Dụng Progressive Web Apps (PWA) Nếu Có Thể

Nếu ứng dụng của bạn cung cấp các tính năng offline hoặc muốn được truy cập từ màn hình nền, xem xét việc phát triển thành một ứng dụng Progressive Web App.

### Chăm Sóc UX/UI

Thiết kế giao diện người dùng (UI) và trải nghiệm người dùng (UX) sao cho chúng phản ánh mong muốn và thói quen của người dùng di động.

### Kiểm Thủ Cẩn Thận Trên Thiết Bị Di Động Thực

Kiểm tra trang web của bạn trên các thiết bị di động thực để đảm bảo rằng trang web hoạt động đúng trên các điều kiện thực tế.

Nhớ rằng, việc chú ý đến trải nghiệm người dùng và hiệu suất sẽ giúp trang web của bạn thu hút và giữ chân người dùng trên các thiết bị di động.



# GIẢI THÍCH TỪ KHÓA

|                     |   |
|---------------------|---|
| <b>Sublime Text</b> | Là một trình soạn thảo mã nguồn phổ biến được đánh giá cao về hiệu suất, tính nhất quán và khả năng tùy chỉnh. Nó là lựa chọn tốt cho lập trình viên và những người viết mã nguồn cần một công cụ nhẹ nhàng, linh hoạt và mạnh mẽ.  |
| <b>Atom</b>         | Là một trình soạn thảo mã nguồn mở được phát triển bởi GitHub. Nó là một công cụ dành cho lập trình viên, cho phép họ viết, chỉnh sửa và quản lý mã nguồn một cách dễ dàng và linh hoạt.  |
| <b>Flexbox</b>      | Hoặc còn gọi là CSS Flexible Box Layout, là một kỹ thuật thiết kế giao diện trong CSS cho phép bạn tạo ra cấu trúc linh hoạt và dễ dàng căn chỉnh các phần tử trên một trục hoặc nhiều trục một cách hiệu quả.  |
| <b>SQL</b>          | Là viết tắt của Structured Query Language (Ngôn Ngữ Truy Vấn Cấu Trúc) và là một ngôn ngữ lập trình chuyên dụng được thiết kế để quản lý và truy vấn cơ sở dữ liệu quan hệ. SQL được sử dụng rộng rãi trong việc tạo, đọc, cập nhật và xóa dữ liệu từ cơ sở dữ liệu quan hệ, thường là các hệ thống quản lý cơ sở dữ liệu như MySQL, PostgreSQL, SQLite, SQL Server, Oracle và nhiều hơn nữa. |
| <b>MySQL</b>        | Là một hệ thống quản lý cơ sở dữ liệu quan hệ mã nguồn mở (RDBMS) phổ biến, được phát triển, hỗ trợ và duy trì bởi Oracle Corporation. Nó là một phần của bộ sản phẩm MySQL, bao gồm các công cụ, thư viện và hệ thống quản lý cơ sở dữ liệu, được sử dụng rộng rãi cho các ứng dụng web và doanh nghiệp.   |

|                 |   |
|-----------------|---|
| <b>React</b>    | Là một thư viện JavaScript phổ biến được sử dụng để xây dựng các ứng dụng web hiện đại. Được phát triển bởi Facebook, React giúp phát triển giao diện người dùng tương tác một cách dễ dàng và linh hoạt.   |
| <b>Git</b>      | Là một hệ thống quản lý phiên bản phân tán (distributed version control system - DVCS) được thiết kế để theo dõi các thay đổi trong mã nguồn khi nhiều người cùng làm việc trên dự án phát triển phần mềm.  |
| <b>HTTP</b>     | Là viết tắt của Hypertext Transfer Protocol (Giao thức truyền tải siêu văn bản). Nó là một giao thức dùng để truyền tải dữ liệu qua Internet. Giao thức này là cơ sở cho việc truy cập các trang web và truyền tải dữ liệu giữa máy tính của bạn và máy chủ web (server).   |
| <b>UX</b>       | Là viết tắt của User Experience, tức là Trải nghiệm Người dùng. Nó liên quan đến cách mà người dùng tương tác với và cảm nhận về một sản phẩm hoặc dịch vụ khi họ sử dụng nó. UX không chỉ bao gồm các khía cạnh thiết kế giao diện (UI - User Interface), mà còn bao hàm cảm xúc, thái độ, và trải nghiệm tổng thể của người dùng khi họ tương tác với sản phẩm hoặc dịch vụ đó. |
| <b>WebStorm</b> | Là một IDE (Integrated Development Environment) dành cho lập trình viên phát triển ứng dụng web. Được phát triển bởi JetBrains, WebStorm được thiết kế đặc biệt để hỗ trợ lập trình frontend và backend cho các ứng dụng web, trang web và ứng dụng web di động.  |
| <b>JIRA</b>     | Là một công cụ quản lý dự án và theo dõi công việc được phát triển bởi Atlassian. Nó được thiết kế đặc biệt cho các nhóm phát triển phần mềm và hỗ trợ quản lý các yêu cầu, lỗi, nhiệm vụ, và các công việc phát triển.   |

## Trello

Là một công cụ quản lý dự án dựa trên hệ thống thẻ và bảng. Người dùng có thể tạo các thẻ để đại diện cho các công việc hoặc nhiệm vụ, và sắp xếp chúng vào các bảng. Các thẻ có thể được di chuyển giữa các danh sách (list) trên bảng để thể hiện tiến độ công việc từ trạng thái này sang trạng thái khác. Trello thường được sử dụng cho các dự án nhỏ đến trung bình và được chú trọng vào việc quản lý công việc theo hình thức thẻ kéo và thả (drag-and-drop).

## Atom

Là một công cụ quản lý dự án trực tuyến được thiết kế để giúp các nhóm tổ chức công việc, theo dõi tiến độ và tương tác với nhau trong môi trường làm việc chung. Asana cho phép người dùng tạo công việc, lập kế hoạch, gán người thực hiện, đặt hạn chót và theo dõi tiến độ công việc thông qua các bảng, danh sách, và lịch. Asana cũng hỗ trợ tích hợp với các ứng dụng và dịch vụ khác để tối ưu hóa quy trình làm việc.

# LỜI CẢM ƠN !!!

Chúng tôi tin rằng việc học lập trình web không chỉ là hành trình ngắn hạn mà còn là một cuộc phiêu lưu kéo dài, mở ra trước bạn những cánh cửa mới của kiến thức và sự sáng tạo. Trong tập 1 của cuốn sách “**Lập Trình Web từ Cơ Bản đến Chuyên Sâu**”, chúng tôi đã nỗ lực truyền đạt những kiến thức cơ bản nhưng thiết yếu, giúp bạn xây dựng nền tảng vững chắc để tiến xa hơn trong lĩnh vực này.

Qua từng trang sách, chúng tôi hy vọng bạn đã không chỉ nắm bắt được cú pháp và kỹ thuật, mà còn hiểu được sâu sắc ý nghĩa của việc tạo ra những trang web và ứng dụng web hấp dẫn. Lập trình web không chỉ là việc code một chuỗi ký tự, mà còn là nghệ thuật biến ý tưởng thành hiện thực, là cách kết nối người dùng, giải quyết vấn đề và mang lại trải nghiệm khó quên.

Trước khi chúng tôi kết thúc cuốn sách này, chúng tôi muốn gửi lời cảm ơn chân thành tới các bạn đã ủng hộ và đồng hành cùng chúng tôi. Sự hỗ trợ và động viên của các bạn là nguồn động lực to lớn giúp chúng tôi hoàn thiện cuốn sách này. Chúng tôi cũng muốn gửi lời cảm ơn đặc biệt tới những người đã dành thời gian và công sức để đọc cuốn sách này, hy vọng rằng nó đã đáp ứng được mong đợi của bạn và giúp bạn có thêm kiến thức và kỹ năng trong lĩnh vực lập trình web.

Nếu bạn đã tìm thấy giá trị trong cuốn sách này, chúng tôi mong muốn nhìn thấy bạn tiếp tục hành trình của mình trong lĩnh vực lập trình web. Đừng ngừng học hỏi, thách thức bản thân và khám phá những khả năng mới. Lập trình web là một lĩnh vực không ngừng phát triển, và chúng tôi tin rằng bạn có thể góp phần vào sự đổi mới và sáng tạo trong ngành này. Các bạn tiếp tục tìm đọc tập số 2 để hiểu sâu và nắm vững các bài tập thực hành phục vụ học tập và công việc một cách thuận lợi nhất.

Cuối cùng, chúng tôi chân thành cảm ơn bạn đã đồng hành cùng chúng tôi trong cuốn sách này. Hy vọng chúng tôi sẽ gặp lại bạn trong các tập tiếp theo của series “**Lập Trình Web từ Cơ Bản đến Chuyên Sâu**”. Chúc bạn thành công trên hành trình lập trình và trong mọi nỗ lực của cuộc sống.

Trân trọng và cảm ơn bạn !!!

Đào Xuân Hiệp

