TRƯỜNG ĐẠI HỌC TRÀ VINH

KHOA KỸ THUẬT VÀ CÔNG NGHỆ

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN HỌC**

**THIẾT KẾ WEBSITE BÁN GIÀY THỜI TRANG**

|  |  |
| --- | --- |
| **Giáo viên hướng dẫn:**  Nguyễn Ngọc Đan Thanh | **Sinh viên thực hiện:**  Mã số sinh viên: 110122079  Họ tên: Trầm Tấn Khá  Lớp: DA22TTD |

***Trà Vinh, tháng 4 năm 2024***

**TRƯỜNG ĐẠI HỌC TRÀ VINH**

**KHOA KỸ THUẬT VÀ CÔNG NGHỆ**

**BỘ MÔN CÔNG NGHỆ THÔNG TIN**



**ĐỒ ÁN MÔN HỌC**

**THIẾT KẾ WEBSITE BÁN GIÀY THỜI TRANG**

|  |  |
| --- | --- |
| **Giáo viên hướng dẫn:**  Nguyễn Ngọc Đan Thanh | **Sinh viên thực hiện:**  Mã số sinh viên: 110122079  Họ tên: Trầm Tấn Khá  Lớp: DA22TTD |

***Trà Vinh, tháng 4 năm 2024***

**MỤC LỤC**

[CHƯƠNG 1. Tổng quan về nội dung nghiên cứu 1](#_Toc164618487)

[1.1. Quy trình phát triển website 1](#_Toc164618488)

[1.1.1. Lên kế hoạch và thu thập yêu cầu: 1](#_Toc164618489)

[1.1.2. Thiết kế giao diện: 1](#_Toc164618490)

[1.1.3. Phát triển website: 1](#_Toc164618491)

[1.1.4. Thử nghiệm và triển khai: 1](#_Toc164618492)

[1.1.5. Bảo trì và cập nhật: 1](#_Toc164618493)

[1.1.6. Ngoài ra, quy trình phát triển website có thể bao gồm thêm các bước sau: 2](#_Toc164618494)

[1.2. Ngôn ngữ HTML 2](#_Toc164618495)

[1.2.1. Khái niệm về HTML 2](#_Toc164618496)

[1.2.2. HTML cơ bản 2](#_Toc164618497)

[1.2.3. Phần tử HTML 3](#_Toc164618498)

[1.2.4. Thuộc tính HTML 5](#_Toc164618499)

[1.2.5. Tiêu đề trong HTML 7](#_Toc164618500)

[1.2.6. Đoạn văn bản trong HTML 8](#_Toc164618501)

[1.2.7. Định dạng văn bản trong HTML 9](#_Toc164618502)

[1.2.8. Nhận xét trong HTML: 10](#_Toc164618503)

[1.2.9. Liên kết trong HTML 11](#_Toc164618504)

[1.2.10. Hình ảnh trong HTML 14](#_Toc164618505)

[1.2.11. Bảng trong HTML: 15](#_Toc164618506)

[1.2.12. Danh sách trong HTML 22](#_Toc164618507)

[1.2.13. Phần tử khối và phần tử nội tuyến: 24](#_Toc164618508)

[1.2.14. Khung nội tuyến trong HTML 26](#_Toc164618509)

[1.2.15. Đường dẫn tệp trong HTML 27](#_Toc164618510)

[1.2.16. Biểu mẫu trong HTML 28](#_Toc164618511)

[1.3. Bảng định kiểu CSS 35](#_Toc164618512)

[1.3.1. CSS nội tuyến (Inline CSS) 35](#_Toc164618513)

[1.3.2. CSS nội bộ (Internal CSS) 35](#_Toc164618514)

[1.3.3. CSS bên ngoài (External CSS) 36](#_Toc164618515)

[1.4. Ngôn ngữ Javascript 36](#_Toc164618516)

[1.4.1. Giới thiệu về Javascript 36](#_Toc164618517)

[1.4.2. Thêm Javascript vào trang 37](#_Toc164618518)

[1.4.3. Javascript cơ bản 37](#_Toc164618519)

[1.4.4. Variables(Biến) 38](#_Toc164618520)

[1.4.5. Data types(Loại dữ liệu) 39](#_Toc164618521)

[1.4.6. Comparison Operators(Toán tử so sánh) 41](#_Toc164618522)

[1.4.7. Mathematical operators (Toán tử toán học) 41](#_Toc164618523)

[1.4.8. if/else statements(câu lệnh if/else) 42](#_Toc164618524)

[1.4.9. Loops(vòng lặp) 42](#_Toc164618525)

[1.4.10. Functions(chức năng) 43](#_Toc164618526)

[1.4.11. Variable Scope and the var Keyword(Phạm vi biến và từ khóa var) 44](#_Toc164618527)

[1.4.12. THE BROWSER OBJECT (ĐỐI TƯỢNG TRÌNH DUYỆT) 45](#_Toc164618528)

[1.4.13. Events 45](#_Toc164618529)

[1.4.14. DOM 47](#_Toc164618530)

[1.5. Kết chương 60](#_Toc164618531)

[CHƯƠNG 2. Phân tích thiết kế hệ thống 61](#_Toc164618532)

[2.1. Mô tả bài toán 61](#_Toc164618533)

[2.2. Đặc tả các yêu cầu 61](#_Toc164618534)

[2.2.1. Yêu cầu chức năng 61](#_Toc164618535)

[2.2.2. Yêu cầu phi chức năng 61](#_Toc164618536)

[2.3. Thiết kế dữ liệu 61](#_Toc164618537)

[2.3.1. Mô hình dữ liệu 61](#_Toc164618538)

[2.3.2. Danh sách các thực thể và mối kết hợp 61](#_Toc164618539)

[2.3.3. Chi tiết thực thể và mối kết hợp 61](#_Toc164618540)

[2.4. Thiết kế xử lý 61](#_Toc164618541)

[2.4.1. Sơ đồ DFD mức ngữ cảnh 61](#_Toc164618542)

[2.4.2. Sơ đồ DFD mức 1 62](#_Toc164618543)

[2.4.3. Sơ đồ DFD mức 2 (Theo từng chức năng) 62](#_Toc164618544)

[2.5. Thiết kế giao diện 62](#_Toc164618545)

[2.5.1. Sơ đồ hệ thống 62](#_Toc164618546)

[2.5.2. Giao diện trang chủ người dùng 63](#_Toc164618547)

[2.5.2.1 Giao diện chức năng … 63](#_Toc164618548)

[2.5.2.2 Giao diện chức năng … 64](#_Toc164618549)

[2.5.2.3 Giao diện chức năng … 64](#_Toc164618550)

[2.5.3. Giao diện trang chủ quản trị 65](#_Toc164618551)

[2.5.3.1 Giao diện chức năng … 65](#_Toc164618552)

[2.5.3.2 Giao diện chức năng … 65](#_Toc164618553)

[2.5.3.3 Giao diện chức năng … 65](#_Toc164618554)

[2.5.4. Kết chương 65](#_Toc164618555)

[CHƯƠNG 3. Cài đặt thực nghiệm 66](#_Toc164618556)

[3.1. Bộ dữ liệu thử nghiệm 66](#_Toc164618557)

[3.2. Kết quả thực nghiệm 66](#_Toc164618558)

[3.2.1. Chức năng … 66](#_Toc164618559)

[3.2.2. Chức năng … 66](#_Toc164618560)

[3.2.3. Chức năng 66](#_Toc164618561)

[3.3. Kết chương 66](#_Toc164618562)

[CHƯƠNG 4. Kết luận và hướng phát triển 67](#_Toc164618563)

[4.1. Kết luận 67](#_Toc164618564)

[4.2. Hướng phát triển 67](#_Toc164618565)

**DANH MỤC HÌNH ẢNH**

[Hình 2.1 Sơ đồ hệ thống 3](#_Toc164154428)

[Hình 2.2 Giao diện trang chủ của người dùng 4](#_Toc164154429)

[Hình 2.3 Giao diện chức năng … 4](#_Toc164154430)

[Hình 2.4 Hình … 5](#_Toc164154431)

**DANH MỤC BẢNG BIỂU**

[Bảng 2.1 Danh sách các thực thể và mối kết hợp 2](#_Toc164154346)

[Bảng 2.2 Chi tiết thực thể … 2](#_Toc164154347)

# Tổng quan về nội dung nghiên cứu

## Quy trình phát triển website

### Lên kế hoạch và thu thập yêu cầu:

* Phân tích nhu cầu của khác hàng
* Phân tích xu hướng thị trường
* Xác định mục tiêu kinh danh, đối tượng người dùng
* Xác định phạm vi dự án, ngân sách và tài nguyên cần thiết

### Thiết kế giao diện:

* Tạo ra mẫu thiết kế thực tế với sự chú trọng vào thẩm mỹ và trải nghiệm người dùng.
* Tạo bản mẫu giao diện wireframes: Xác định cấu trúc trang web, bố trí các phần chính như header, footer,..vv.

### Phát triển website:

* Sử dụng các công nghệ phù hợp để phát triển website, ví dụ như HTML, CSS, JavaScript cho phần frontend và các framework như Node.js hoặc PHP cho phần beckend, và MySQL.
* Xây dựng hệ thống quản lý nội dung(CMS): Đối với website bán hàng, việc sử dụng CMS như WordPress, Magento, WooCommerce có thể giúp quản lý sản phẩm và đơn hàng dễ dàng hơn.

### Thử nghiệm và triển khai:

* Kiểm tra từng tính năng của website như tìm kiếm, đặt hàng, thanh toán để chúng hoạt động đúng cách
* Đảm bảo website hoạt động mượt mà trên các trình duyệt và thiết bị khác nhau
* Triển khai website trên một máy chủ ổn định và cấu hình đúng cách để đảm bảo hiệu suất và bảo mật.

### Bảo trì và cập nhật:

* Liên tục duy trì và cập nhật website
* Bảo trì hệ thống , sửa lỗi website

### Ngoài ra, quy trình phát triển website có thể bao gồm thêm các bước sau:

* Tích hợp các công thanh toán trực tuyến để người dùng có thể thanh toán một cách tiện dụng và an toàn
* Xây dựng chiên lượt Marketing để thu hút và giữ chân khách hàng.

## Ngôn ngữ HTML

### Khái niệm về HTML

- HTML là viết tắt của HyperText Markup Language: là ngôn ngữ đánh dấu được sử dụng để tạo cấu trúc và định dạng nội dung trên trang web. HTML cung cấp một cách để mô tả cấu trúc của một trang web để định nghĩa các phần tử như tiêu đề, đoạn văn, hình ảnh, liên kết,v.v.

### HTML cơ bản

- Một tài liệu HTML phải có phần khai báo loại tài liệu bắt đầu bằng <!DOCTYPE html>.

- Phần chính của tài liệu HTML được đặt giữa thẻ <html> và </html>.

- Phần nội dung được hiển thị trên trình duyệt được đặt trong thẻ <body> và </body>.

- Ví dụ:

<!DOCTYPE html>

<html>

<body>

<h1>My First Heading</h1>

<p>My first paragraph.</p>

</body>

</html>

\*Khai báo <!DOCTYPE>:

- Khai báo <!DOCTYPE> thể hiện loại tài liệu và giúp trình duyệt hiển thị trang web một cách chính xác.

- Nó chỉ được khai báo một lần, ở đầu trang (trước bất kỳ thẻ HTML nào).

- Khai báo <!DOCTYPE> không phân biệt chữ hoa chữ thường.

- Khai báo <!DOCTYPE> cho HTML5 là:

<!DOCTYPE html>

### Phần tử HTML

* Thẻ mở (opening tag): Bao gồm tên của phần tử được đặt trong dấu ngoặc nhọn `<>`.

- Ví dụ: `<h1>`, `<p>`, `<img>`.

* Nội dung (content): Nội dung được đặt giữa thẻ mở và thẻ đóng.

- Ví dụ: `<h1>Tiêu đề chính</h1>`, `<p>Đây là một đoạn văn bản.</p>`.

* Thẻ đóng (closing tag): Bao gồm tên của phần tử được đặt trong dấu ngoặc nhọn, có thêm dấu gạch chéo `/` trước tên.
* \*\*Thẻ `<p>`\*\*: Để hiển thị một đoạn văn bản.

<p>This is a paragraph.</p>

* . \*\*Thẻ `<a>`\*\*: Để tạo một liên kết đến một trang khác.

<a href="https://www.example.com">Visit Example Website</a>

* \*\*Thẻ `<img>`\*\*: Để hiển thị một hình ảnh trên trang web.

<img src="image.jpg" alt="Description of image">

* \*\*Thẻ `<ul>` và `<li>`\*\*: Để tạo ra một danh sách không có thứ tự.

<ul>

<li>Item 1</li>

<li>Item 2</li>

<li>Item 3</li>

</ul>

* \*\*Thẻ `<ol>` và `<li>`\*\*: Để tạo ra một danh sách có thứ tự.

<ol>

<li>First item</li>

<li>Second item</li>

<li>Third item</li>

</ol>

```

* \*\*Thẻ `<table>`, `<tr>` và `<td>`\*\*: Để tạo ra một bảng.

<table>

<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>

* \*\*Thẻ `<form>`\*\*: Để tạo ra một biểu mẫu gửi dữ liệu.

<form action="/submit-form" method="post">

<label for="username">Username:</label>

<input type="text" id="username" name="username"><br><br>

<label for="password">Password:</label>

<input type="password" id="password" name="password"><br><br>

<input type="submit" value="Submit">

</form>: `</h1>`, `</p>`.

* \*\*Thẻ `<h1>`, `<h2>`, ..., `<h6>`\*\*: Để xác định các tiêu đề với mức độ quan trọng giảm dần.

<h1>This is a Heading 1</h1>

<h2>This is a Heading 2</h2>

<h3>This is a Heading 3</h3>

* \*\*Thẻ `<div>`\*\*: Để nhóm các phần tử HTML lại với nhau và tạo ra các phần vùng trên trang web.

<div>

<p>This paragraph is inside a div.</p>

<img src="image.jpg" alt="Description of image">

</div>

* \*\*Thẻ `<span>`\*\*: Để định dạng một phần nhỏ của văn bản hoặc một phần nhỏ của một phần tử HTML.

<p>This is <span style="color: blue;">blue</span> text.</p>

* \*\*Thẻ `<button>`\*\*: Để tạo ra một nút bấm.

<button type="button">Click Me!</button>

* \*\*Thẻ `<input>`\*\*: Được sử dụng trong các biểu mẫu để nhận dữ liệu từ người dùng.

<input type="text" id="username" name="username">

* .\*\*Thẻ `<textarea>`\*\*: Để tạo ra một vùng văn bản đa dòng cho người dùng nhập liệu.

<textarea id="message" name="message" rows="4" cols="50"></textarea>

* \*\*Thẻ `<iframe>`\*\*: Để nhúng một trang web khác vào trong trang web hiện tại.

<iframe src="https://www.example.com" width="600" height="400"></iframe>

* \*\*Thẻ `<audio>`\*\*: Để nhúng một tệp âm thanh vào trong trang web.

<audio controls>

<source src="audio.mp3" type="audio/mpeg">

Your browser does not support the audio element.

</audio>

### Thuộc tính HTML

Trong HTML, các thuộc tính (attributes) được sử dụng để cung cấp thêm thông tin hoặc định cấu hình cho các phần tử. Các thuộc tính được đặt bên trong thẻ mở của phần tử.

Cấu trúc cơ bản của một thuộc tính:

<tên\_phần\_tử tên\_thuộc\_tính="giá\_trị\_thuộc\_tính">Nội dung</tên\_phần\_tử>

Một số ví dụ về các thuộc tính phổ biến:

* `src`: Chỉ định nguồn (URL) của một tài nguyên, như hình ảnh hoặc video.

<img src="image.jpg" alt="Ảnh minh họa">

<video src="video.mp4" controls></video>

* `href`: Chỉ định đường dẫn (URL) của một liên kết.

<a href="https://www.example.com">Liên kết đến trang web khác</a>

* `alt`: Cung cấp văn bản thay thế cho hình ảnh khi không thể hiển thị.

<img src="image.jpg" alt="Mô tả hình ảnh">

* `class` và `id`: Được sử dụng để xác định và phân loại các phần tử.

<div class="container">Nội dung</div>

<h1 id="main-title">Tiêu đề chính</h1>

* `style`: Cho phép định nghĩa các kiểu CSS trực tiếp trong HTML.

<p style="color: red; font-size: 16px;">Đoạn văn bản có kiểu định dạng</p>

* `title`: Cung cấp thông tin bổ sung về phần tử khi người dùng di chuột qua.

<a href="https://www.example.com" title="Liên kết đến trang web chính">Liên kết</a>

* `lang`: Chỉ định ngôn ngữ của nội dung.

<html lang="en">

<body>

<p>This is an English paragraph.</p>

</body>

</html>

### Tiêu đề trong HTML

Trong HTML, các tiêu đề được định nghĩa bằng các phần tử `<h1>` đến `<h6>`, với `<h1>` là tiêu đề cấp cao nhất và `<h6>` là tiêu đề cấp thấp nhất.

Cấu trúc cơ bản của các phần tử tiêu đề:

<h1>Tiêu đề cấp 1</h1>

<h2>Tiêu đề cấp 2</h2>

<h3>Tiêu đề cấp 3</h3>

<h4>Tiêu đề cấp 4</h4>

<h5>Tiêu đề cấp 5</h5>

<h6>Tiêu đề cấp 6</h6>

Các tiêu đề trong HTML có các đặc điểm sau:

* \*\*Phân cấp\*\*: Các phần tử tiêu đề được sử dụng để tạo ra cấu trúc phân cấp của nội dung. `<h1>` là tiêu đề cấp cao nhất, thường dùng cho tiêu đề chính của trang web hoặc phần. Các tiêu đề cấp thấp hơn (`<h2>`, `<h3>`, v.v.) được sử dụng để tạo các tiêu đề phụ và chia nhỏ nội dung.
* \*\*Kích thước và định dạng\*\*: Mặc định, các phần tử tiêu đề sẽ được trình duyệt hiển thị với kích thước và định dạng khác nhau, phù hợp với cấp độ của chúng. Tuy nhiên, bạn có thể tùy chỉnh định dạng bằng cách sử dụng CSS.
* \*\*Tính ngữ nghĩa\*\*: Các phần tử tiêu đề giúp tạo ra cấu trúc ngữ nghĩa cho nội dung, cho phép trình duyệt và công cụ tìm kiếm hiểu rõ hơn về cấu trúc và nội dung của trang web.

Ví dụ về cách sử dụng các tiêu đề trong HTML:

<h1>Tiêu đề chính của trang web</h1>

<p>Đoạn văn bản giới thiệu về nội dung chính của trang web.</p>

<h2>Tiêu đề phần 1</h2>

<p>Nội dung của phần 1.</p>

<h3>Tiêu đề phần con 1.1</h3>

<p>Nội dung của phần con 1.1.</p>

<h3>Tiêu đề phần con 1.2</h3>

<p>Nội dung của phần con 1.2.</p>

<h2>Tiêu đề phần 2</h2>

<p>Nội dung của phần 2.</p>

### Đoạn văn bản trong HTML

Trong HTML, đoạn văn bản được định nghĩa bằng phần tử `<p>`. Đây là phần tử cơ bản để tạo ra các đoạn văn bản trên trang web.

Cấu trúc cơ bản của một đoạn văn bản trong HTML:

<p>Đây là một đoạn văn bản.</p>

Một số đặc điểm của đoạn văn bản trong HTML:

* \*\*Ngắt dòng\*\*: Khi bạn muốn tạo một dòng mới trong một đoạn văn bản, bạn có thể sử dụng phần tử `<br>` (line break).

<p>Đây là dòng thứ nhất.<br>

Đây là dòng thứ hai.</p>

* \*\*Căn lề\*\*: Bạn có thể sử dụng các thuộc tính CSS như `text-align` để căn lề đoạn văn bản.

<p style="text-align: center;">Đây là đoạn văn bản được căn giữa.</p>

* 3. \*\*Định dạng văn bản\*\*: Bạn có thể sử dụng các phần tử định dạng văn bản như `<b>`, `<i>`, `<u>`, `<strong>`, `<em>` để tạo các hiệu ứng như in đậm, in nghiêng, gạch chân, v.v.

<p>Đây là <b>văn bản in đậm</b> và <i>văn bản in nghiêng</i>.</p>

* \*\*Khoảng cách\*\*: Các đoạn văn bản trong HTML thường được tách biệt bằng một khoảng cách (margin) hoặc một dòng trống. Bạn có thể điều chỉnh khoảng cách này bằng CSS.

<p>Đoạn văn bản 1.</p>

<p>Đoạn văn bản 2.</p>

### Định dạng văn bản trong HTML

Trong HTML, có nhiều cách để định dạng văn bản, bao gồm:

* \*\*Các phần tử cơ bản\*\*:

- `<p>`: Định nghĩa một đoạn văn bản.

- `<br>`: Tạo một dòng mới (line break).

- `<hr>`: Tạo một đường phân cách ngang (horizontal rule).

* \*\*Các phần tử định dạng văn bản\*\*:

- `<b>`: In đậm (bold).

- `<i>`: In nghiêng (italic).

- `<u>`: Gạch chân (underline).

- `<strong>`: Tạo văn bản quan trọng.

- `<em>`: Tạo văn bản nhấn mạnh.

- `<small>`: Tạo văn bản nhỏ hơn.

- `<sub>` và `<sup>`: Tạo chỉ số dưới và chỉ số trên.

* \*\*Các phần tử định dạng tiêu đề\*\*:

- `<h1>` đến `<h6>`: Các phần tử tiêu đề từ cấp 1 đến cấp 6.

* \*\*Các phần tử định dạng danh sách\*\*:

- `<ul>`: Danh sách không thứ tự (unordered list).

- `<ol>`: Danh sách có thứ tự (ordered list).

- `<li>`: Mục trong danh sách (list item).

* \*\*Các phần tử định dạng trích dẫn\*\*:

- `<blockquote>`: Định nghĩa một đoạn trích dẫn dài.

- `<q>`: Định nghĩa một trích dẫn ngắn.

- `<cite>`: Định nghĩa tên của một tác phẩm.

* \*\*Các phần tử định dạng khác\*\*:

- `<code>`: Định nghĩa đoạn mã nguồn.

- `<pre>`: Định nghĩa văn bản được định dạng sẵn (pre-formatted text).

- `<abbr>`: Định nghĩa một viết tắt.

- `<address>`: Định nghĩa thông tin liên hệ.

Ví dụ về cách sử dụng các phần tử định dạng văn bản:

<p>Đây là một đoạn văn bản <b>in đậm</b> và <i>in nghiêng</i>.</p>

<p><strong>Chú ý:</strong> Đây là một thông báo quan trọng.</p>

<p><small>Đây là văn bản nhỏ hơn.</small></p>

<p>Công thức hóa học: H<sub>2</sub>O</p>

<p>Albert Einstein nói: <q>Tưởng tượng quan trọng hơn kiến thức.</q></p>

<blockquote>

<p>Suy nghĩ là những gì làm nên con người.</p>

<cite>- René Descartes</cite>

</blockquote

### Nhận xét trong HTML

Trong HTML, bạn có thể thêm các nhận xét vào mã nguồn của trang web. Nhận xét là những đoạn văn bản không được trình duyệt web hiển thị ra ngoài, nhưng chúng có thể giúp bạn ghi chú, giải thích hoặc cung cấp thông tin bổ sung về mã HTML.

Cú pháp để thêm nhận xét trong HTML:

<!-- Đây là một nhận xét -->

Ví dụ:

<!-- Đây là phần tiêu đề của trang web -->

<h1>Chào mừng đến với trang web của tôi</h1>

<!-- Đây là phần nội dung chính -->

<p>Nội dung chính của trang web sẽ được hiển thị ở đây.</p>

<!-- Đây là phần chân trang -->

<footer>

<p>&copy; 2023 Trang web của tôi. Bản quyền đã được bảo vệ.</p>

</footer>

Các lưu ý khi sử dụng nhận xét trong HTML:

* \*\*Tính rõ ràng\*\*: Hãy viết nhận xét rõ ràng, ngắn gọn và mô tả chính xác mục đích của đoạn mã.
* \*\*Tránh sử dụng quá nhiều\*\*: Không nên lạm dụng nhận xét, vì chúng có thể làm tăng kích thước tệp HTML và ảnh hưởng đến hiệu suất trang web.
* \*\*Không hiển thị trên trình duyệt\*\*: Nhận xét sẽ không được trình duyệt web hiển thị ra ngoài, chúng chỉ dùng để ghi chú và giải thích mã HTML.
* \*\*Không lồng nhận xét\*\*: Bạn không thể lồng nhận xét vào bên trong nhận xét khác.

### Liên kết trong HTML

Trong HTML, liên kết (link) được định nghĩa bằng phần tử `<a>` (anchor). Liên kết cho phép người dùng di chuyển từ trang này sang trang khác hoặc từ vị trí này sang vị trí khác trong cùng một trang.

Cấu trúc cơ bản của một liên kết trong HTML:

<a href="URL">Văn bản liên kết</a>

Trong đó:

- `href` là thuộc tính chứa địa chỉ URL (Uniform Resource Locator) mà liên kết sẽ trỏ đến.

- "Văn bản liên kết" là nội dung hiển thị trên trang web, thường là một từ hoặc cụm từ mô tả liên kết.

Ví dụ:

<a href="https://www.example.com">Đây là một liên kết</a>

Khi người dùng nhấp vào "Đây là một liên kết", trình duyệt sẽ chuyển hướng đến trang web `https://www.example.com`.

Một số loại liên kết phổ biến:

* \*\*Liên kết đến trang web khác\*\*: Sử dụng URL bắt đầu bằng `http://` hoặc `https://`.
* \*\*Liên kết đến trang web trong cùng trang\*\*: Sử dụng URL bắt đầu bằng `#` và theo sau là ID của phần tử.
* \*\*Liên kết đến email\*\*: Sử dụng `mailto:` theo sau là địa chỉ email.
* \*\*Liên kết đến tệp tải xuống\*\*: Sử dụng URL của tệp cần tải xuống.

Ngoài ra, bạn cũng có thể thêm các thuộc tính khác vào phần tử `<a>` để tùy chỉnh hành vi của liên kết, chẳng hạn như:

- `target="\_blank"`: Mở liên kết trong một tab/cửa sổ mới.

- `title="Mô tả liên kết"`: Hiển thị mô tả khi di chuột qua liên kết.

### Hình ảnh trong HTML

Trong HTML, hình ảnh được đưa vào trang web bằng cách sử dụng phần tử `<img>`. Đây là một phần tử không có thẻ đóng, nghĩa là nó chỉ có một thẻ mở.

Cấu trúc cơ bản của một hình ảnh trong HTML:

<img src="đường\_dẫn\_hình\_ảnh" alt="mô\_tả\_hình\_ảnh">

Trong đó:

- `src` là thuộc tính chứa đường dẫn (URL) đến hình ảnh.

- `alt` là thuộc tính chứa mô tả văn bản của hình ảnh, được hiển thị khi hình ảnh không thể tải được.

Ví dụ:

<img src="logo.png" alt="Logo của công ty">

Khi trang web được tải, trình duyệt sẽ hiển thị hình ảnh `logo.png` và nếu hình ảnh không thể tải được, thì sẽ hiển thị mô tả "Logo của công ty".

Một số lưu ý khi sử dụng hình ảnh trong HTML:

* \*\*Định dạng hình ảnh\*\*: Các định dạng hình ảnh phổ biến là JPEG, PNG, GIF và SVG. Hãy chọn định dạng phù hợp với nhu cầu của bạn.
* \*\*Kích thước hình ảnh\*\*: Bạn nên chỉ định kích thước của hình ảnh bằng cách sử dụng thuộc tính `width` và `height`. Điều này giúp trang web tải nhanh hơn.
* \*\*Tối ưu hóa hình ảnh\*\*: Hãy nén và tối ưu hóa kích thước file hình ảnh để giảm thời gian tải trang.
* 4. \*\*Sử dụng `alt` một cách hợp lý\*\*: Mô tả `alt` nên ngắn gọn và chính xác, giúp người dùng hiểu được nội dung của hình ảnh.
* \*\*Liên kết hình ảnh\*\*: Bạn có thể liên kết hình ảnh đến một URL bằng cách đặt phần tử `<img>` bên trong phần tử `<a>`.

### Bảng trong HTML

Trong HTML, bảng được định nghĩa bằng phần tử `<table>`. Bảng được sử dụng để hiển thị dữ liệu theo hàng và cột, giúp tổ chức thông tin một cách rõ ràng và dễ đọc.

Cấu trúc cơ bản của một bảng trong HTML:

<table>

<tr>

<th>Tiêu đề 1</th>

<th>Tiêu đề 2</th>

<th>Tiêu đề 3</th>

</tr>

<tr>

<td>Dữ liệu 1</td>

<td>Dữ liệu 2</td>

<td>Dữ liệu 3</td>

</tr>

<tr>

<td>Dữ liệu 4</td>

<td>Dữ liệu 5</td>

<td>Dữ liệu 6</td>

</tr>

</table>

```

Trong đó:

* `<table>`: Định nghĩa một bảng.
* `<tr>`: Định nghĩa một hàng trong bảng.
* `<th>`: Định nghĩa một ô tiêu đề trong bảng.
* `<td>`: Định nghĩa một ô dữ liệu trong bảng.

Ví dụ này sẽ tạo ra một bảng với 3 cột (Tiêu đề 1, Tiêu đề 2, Tiêu đề 3) và 3 hàng (Dữ liệu 1-3, Dữ liệu 4-6).

Bạn cũng có thể thêm các thuộc tính khác để tùy chỉnh bảng, chẳng hạn như:

* `border`: Thiết lập đường viền của bảng.
* `cellpadding`: Thiết lập khoảng cách giữa nội dung và đường viền của ô.
* `cellspacing`: Thiết lập khoảng cách giữa các ô.
* `colspan`: Cho phép một ô trải rộng nhiều cột.
* `rowspan`: Cho phép một ô trải rộng nhiều hàng.

ví dụ chi tiết về cách sử dụng bảng trong HTML:

<!DOCTYPE html>

<html>

<head>

<title>Bảng trong HTML</title>

<style>

table {

border-collapse: collapse;

width: 100%;

}

th, td {

padding: 8px;

text-align: left;

border-bottom: 1px solid #ddd;

}

th {

background-color: #f2f2f2;

}

</style>

</head>

<body>

<h1>Bảng Thông Tin Sinh Viên</h1>

<table>

<tr>

<th>Họ và Tên</th>

<th>Tuổi</th>

<th>Điểm Trung Bình</th>

<th>Lớp</th>

</tr>

<tr>

<td>Nguyễn Văn A</td>

<td>20</td>

<td>8.5</td>

<td>CNTT01</td>

</tr>

<tr>

<td>Trần Thị B</td>

<td>21</td>

<td>7.8</td>

<td>CNTT01</td>

</tr>

<tr>

<td>Lê Công C</td>

<td>19</td>

<td>9.2</td>

<td>CNTT02</td>

</tr>

<tr>

<td>Phạm Thị D</td>

<td>22</td>

<td>7.0</td>

<td>CNTT02</td>

</tr>

</table>

</body>

</html>

Trong ví dụ này, chúng ta có:

* 1. Định nghĩa bảng với phần tử `<table>`.
* 2. Định nghĩa các hàng với phần tử `<tr>`.
* 3. Định nghĩa các ô tiêu đề với phần tử `<th>`.
* 4. Định nghĩa các ô dữ liệu với phần tử `<td>`.
* 5. Sử dụng CSS để tùy chỉnh định dạng của bảng, chẳng hạn như:

- `border-collapse: collapse;`: Hợp nhất các đường viền của bảng.

- `width: 100%;`: Thiết lập chiều rộng của bảng bằng 100% chiều rộng của trang.

- `padding: 8px;`: Thiết lập khoảng cách giữa nội dung và đường viền của ô.

- `text-align: left;`: Căn chỉnh nội dung của ô sang trái.

- `border-bottom: 1px solid #ddd;`: Thêm đường viền dưới mỗi hàng.

- `background-color: #f2f2f2;`: Thiết lập màu nền cho các ô tiêu đề.

Khi mở trang web này, bạn sẽ thấy một bảng hiển thị thông tin về các sinh viên, bao gồm họ và tên, tuổi, điểm trung bình và lớp.

### Danh sách trong HTML

Trong HTML, có 3 loại danh sách chính:

* Danh sách không thứ tự (Unordered List - `<ul>`):

- Được sử dụng khi không cần quan tâm đến thứ tự của các mục.

- Các mục được đánh dấu bằng các ký hiệu như dấu chấm, hình tròn, hình vuông, v.v.

- Được định nghĩa bằng thẻ `<ul>` và các mục được định nghĩa bằng thẻ `<li>`.

Ví dụ:

<ul>

<li>Mục 1</li>

<li>Mục 2</li>

<li>Mục 3</li>

</ul>

* Danh sách có thứ tự (Ordered List - `<ol>`):

- Được sử dụng khi cần quan tâm đến thứ tự của các mục.

- Các mục được đánh số thứ tự.

- Được định nghĩa bằng thẻ `<ol>` và các mục được định nghĩa bằng thẻ `<li>`.

Ví dụ:

<ol>

<li>Bước 1</li>

<li>Bước 2</li>

<li>Bước 3</li>

</ol>

* Danh sách mô tả (Description List - `<dl>`):

- Được sử dụng để tạo ra một danh sách các thuật ngữ và định nghĩa tương ứng.

- Được định nghĩa bằng thẻ `<dl>`, các thuật ngữ bằng thẻ `<dt>` và các định nghĩa bằng thẻ `<dd>`.

Ví dụ:

<dl>

<dt>HTML</dt>

<dd>Hypertext Markup Language</dd>

<dt>CSS</dt>

<dd>Cascading Style Sheets</dd>

<dt>JavaScript</dt>

<dd>A programming language for web pages</dd>

</dl>

Ngoài ra, bạn cũng có thể lồng các loại danh sách vào nhau để tạo ra các cấu trúc phức tạp hơn. Ví dụ:

<ul>

<li>Trái cây

<ul>

<li>Táo</li>

<li>Cam</li>

<li>Chuối</li>

</ul>

</li>

<li>Rau củ

<ol>

<li>Cà rốt</li>

<li>Bắp cải</li>

<li>Khoai tây</li>

</ol>

</li>

</ul>

### Phần tử khối và phần tử nội tuyến

Trong HTML, các phần tử được chia thành hai loại chính: phần tử khối (block-level elements) và phần tử nội tuyến (inline elements).

* Phần tử khối (Block-level elements):

- Chiếm toàn bộ chiều rộng của trang web (100% chiều rộng của cha).

- Bắt đầu trên một dòng mới và kết thúc trên một dòng mới.

- Có thể chứa các phần tử khác, bao gồm cả phần tử nội tuyến.

- Một số ví dụ về phần tử khối: `<div>`, `<p>`, `<h1>` đến `<h6>`, `<ul>`, `<ol>`, `<li>`, `<table>`, `<form>`, `<header>`, `<footer>`, `<section>`, `<article>`, v.v.

* Phần tử nội tuyến (Inline elements):

- Chỉ chiếm không gian cần thiết để hiển thị nội dung.

- Không bắt đầu trên một dòng mới và không kết thúc trên một dòng mới.

- Chỉ có thể chứa văn bản hoặc các phần tử nội tuyến khác.

- Không thể chứa các phần tử khối.

- Một số ví dụ về phần tử nội tuyến: `<a>`, `<span>`, `<img>`, `<strong>`, `<em>`, `<br>`, `<button>`, `<input>`, `<label>`, `<select>`, `<textarea>`, v.v.

Sự khác biệt giữa hai loại phần tử này ảnh hưởng đến cách chúng được hiển thị trên trang web. Phần tử khối thường được sử dụng để tạo ra cấu trúc và bố cục của trang, trong khi phần tử nội tuyến được sử dụng để định dạng và tạo hiệu ứng cho nội dung.

Ví dụ:

<div>

<h1>Tiêu đề chính</h1>

<p>Đây là một đoạn văn bản. <span>Một phần của đoạn văn bản</span> được định dạng bằng phần tử nội tuyến.</p>

<ul>

<li>Mục 1</li>

<li>Mục 2</li>

<li>Mục 3</li>

</ul>

</div>

Trong ví dụ trên, `<div>`, `<h1>`, `<p>`, `<ul>` và `<li>` là các phần tử khối, trong khi `<span>` là một phần tử nội tuyến.

### Khung nội tuyến trong HTML

Trong HTML, khung nội tuyến (inline-level elements) là một loại phần tử HTML được sử dụng để định dạng và tạo hiệu ứng cho nội dung bên trong các phần tử khối (block-level elements).

Các đặc điểm của khung nội tuyến:

* Chiếm không gian cần thiết để hiển thị nội dung:

- Không chiếm toàn bộ chiều rộng của trang web.

- Chỉ chiếm không gian cần thiết để hiển thị nội dung bên trong.

* Không bắt đầu trên một dòng mới và không kết thúc trên một dòng mới:

- Các phần tử nội tuyến được hiển thị trên cùng một dòng với các phần tử khác.

- Không tạo ra dòng mới khi được sử dụng.

* Chỉ có thể chứa văn bản hoặc các phần tử nội tuyến khác:

- Không thể chứa các phần tử khối.

- Chỉ có thể chứa các phần tử nội tuyến khác hoặc văn bản.

Một số ví dụ về các phần tử nội tuyến:

- `<a>`: Liên kết siêu văn bản

- `<span>`: Định dạng văn bản

- `<img>`: Hình ảnh

- `<strong>`: Văn bản in đậm

- `<em>`: Văn bản in nghiêng

- `<button>`: Nút bấm

- `<input>`: Trường nhập liệu

- `<label>`: Nhãn cho các phần tử form

Ví dụ:

<p>Đây là một <span style="color:red;">đoạn văn bản</span> với một <a href="#">liên kết</a> và một <img src="image.jpg" alt="Hình ảnh"> bên trong.</p>

Trong ví dụ trên, `<span>`, `<a>` và `<img>` là các phần tử nội tuyến được sử dụng để định dạng và tạo hiệu ứng cho nội dung bên trong phần tử khối `<p>`.

### Đường dẫn tệp trong HTML

Trong HTML, đường dẫn tệp (file path) được sử dụng để xác định vị trí của một tệp tin trên máy tính hoặc trên mạng Internet. Có 3 loại đường dẫn tệp chính:

* 1. Đường dẫn tuyệt đối (Absolute path):

- Đường dẫn tuyệt đối bắt đầu từ thư mục gốc (root directory) của máy tính hoặc trang web.

- Ví dụ: `https://example.com/images/logo.jpg`

* 2. Đường dẫn tương đối (Relative path):

- Đường dẫn tương đối được xác định dựa trên vị trí hiện tại của tệp tin.

- Ví dụ: `images/logo.jpg` (tệp tin nằm trong thư mục "images" cùng cấp với tệp HTML)

* 3. Đường dẫn tương đối với thư mục cha (Parent directory):

- Đường dẫn tương đối với thư mục cha được sử dụng để truy cập tệp tin nằm trong thư mục cha.

- Ký hiệu `..` được sử dụng để chỉ thư mục cha.

- Ví dụ: `../images/logo.jpg` (tệp tin nằm trong thư mục "images" ở cấp trên)

Ví dụ sử dụng đường dẫn tệp trong HTML:

<!-- Đường dẫn tuyệt đối -->

<img src="https://example.com/images/logo.jpg" alt="Logo">

<!-- Đường dẫn tương đối -->

<img src="images/logo.jpg" alt="Logo">

<!-- Đường dẫn tương đối với thư mục cha -->

<img src="../images/logo.jpg" alt="Logo">

Lưu ý:

- Đường dẫn tuyệt đối thường được sử dụng khi tệp tin nằm trên một trang web khác.

- Đường dẫn tương đối thường được sử dụng khi tệp tin nằm trong cùng một trang web.

- Đường dẫn tương đối với thư mục cha được sử dụng khi tệp tin nằm trong thư mục cha của thư mục hiện tại.

### Biểu mẫu trong HTML

Biểu mẫu (form) trong HTML là một phần tử quan trọng để tạo ra các giao diện người dùng tương tác, cho phép người dùng nhập và gửi dữ liệu đến máy chủ. Dưới đây là một số thông tin về biểu mẫu trong HTML:

* 1. Phần tử `<form>`:

- `action`: Chỉ định URL nơi dữ liệu biểu mẫu sẽ được gửi đến.

- Ví dụ: `<form action="/submit-form" method="post">`

- `method`: Chỉ định phương thức HTTP để gửi dữ liệu (GET hoặc POST).

- Ví dụ: `<form action="/submit-form" method="post">`

- `enctype`: Chỉ định cách mã hóa dữ liệu khi gửi (ví dụ: `application/x-www-form-urlencoded`, `multipart/form-data`).

- Ví dụ: `<form action="/submit-form" method="post" enctype="multipart/form-data">`

- `name`: Cung cấp tên cho biểu mẫu.

- Ví dụ: `<form action="/submit-form" method="post" name="contact-form">`

- `target`: Chỉ định nơi hiển thị kết quả sau khi gửi biểu mẫu (ví dụ: `\_blank`, `\_self`).

- Ví dụ: `<form action="/submit-form" method="post" target="\_blank">`

* 2. Phần tử `<input>`:

- `type`: Chỉ định loại trường nhập liệu (ví dụ: `text`, `password`, `email`, `number`, `date`, `checkbox`, `radio`).

- Ví dụ: `<input type="text" name="username">`, `<input type="password" name="password">`

- `name`: Cung cấp tên cho trường nhập liệu.

- Ví dụ: `<input type="text" name="username">`

- `value`: Chỉ định giá trị mặc định cho trường nhập liệu.

- Ví dụ: `<input type="text" name="username" value="JohnDoe">`

- `placeholder`: Hiển thị văn bản hướng dẫn trong trường nhập liệu.

- Ví dụ: `<input type="text" name="username" placeholder="Enter your username">`

- `required`: Yêu cầu người dùng nhập giá trị vào trường.

- Ví dụ: `<input type="text" name="username" required>`

- `minlength` và `maxlength`: Chỉ định độ dài tối thiểu và tối đa của giá trị nhập.

- Ví dụ: `<input type="text" name="username" minlength="3" maxlength="20">`

- `min` và `max`: Chỉ định giá trị tối thiểu và tối đa cho trường số.

- Ví dụ: `<input type="number" name="age" min="18" max="100">`

* 3. Phần tử `<textarea>`:

- `name`: Cung cấp tên cho vùng văn bản.

- Ví dụ: `<textarea name="message"></textarea>`

- `rows` và `cols`: Chỉ định kích thước của vùng văn bản.

- Ví dụ: `<textarea name="message" rows="5" cols="30"></textarea>`

## Bảng định kiểu CSS

### CSS nội tuyến (Inline CSS)

CSS nội tuyến (inline CSS) là cách áp dụng các quy tắc CSS trực tiếp vào các phần tử HTML bằng cách sử dụng thuộc tính `style` của phần tử. Đây là cách đơn giản nhất để áp dụng CSS, nhưng không được khuyến khích sử dụng rộng rãi vì nó không tách biệt được nội dung và trình bày.

Ví dụ về cách sử dụng CSS nội tuyến trong HTML:

<h1 style="color: blue; font-size: 24px;">Tiêu đề</h1>

<p style="text-align: justify; line-height: 1.5;">Đây là một đoạn văn bản. Chúng ta đang sử dụng CSS nội tuyến để định dạng nó.</p>

<button style="background-color: green; color: white; padding: 10px 20px;">Nhấn vào đây</button>

### CSS nội bộ (Internal CSS)

CSS nội bộ (internal CSS) là cách sử dụng các quy tắc CSS được định nghĩa trong phần `<style>` của tài liệu HTML. Điều này cho phép bạn áp dụng các quy tắc CSS cho các phần tử HTML trong cùng một tài liệu.

Ví dụ về cách sử dụng CSS nội bộ trong HTML:

<!DOCTYPE html>

<html>

<head>

<title>Ví dụ về CSS nội bộ</title>

<style>

h1 {

color: blue;

font-size: 24px;

}

p {

text-align: justify;

line-height: 1.5;

}

button {

background-color: green;

color: white;

padding: 10px 20px;

}

</style>

</head>

<body>

<h1>Tiêu đề</h1>

<p>Đây là một đoạn văn bản. Chúng ta đang sử dụng CSS nội bộ để định dạng nó.</p>

<button>Nhấn vào đây</button>

</body>

</html>

### CSS bên ngoài (External CSS)

CSS bên ngoài (external CSS) là cách sử dụng các quy tắc CSS được định nghĩa trong một tệp CSS riêng biệt, sau đó được liên kết với tài liệu HTML thông qua thẻ `<link>`.

Ví dụ về cách sử dụng CSS bên ngoài trong HTML:

Đầu tiên, tạo một tệp CSS riêng biệt, ví dụ như `styles.css`, với các quy tắc CSS:

h1 {

color: blue;

font-size: 24px;

}

p {

text-align: justify;

line-height: 1.5;

}

button {

background-color: green;

color: white;

padding: 10px 20px;

}

Sau đó, trong tài liệu HTML, sử dụng thẻ `<link>` để liên kết tệp CSS:

<!DOCTYPE html>

<html>

<head>

<title>Ví dụ về CSS bên ngoài</title>

<link rel="stylesheet" href="styles.css">

</head>

<body>

<h1>Tiêu đề</h1>

<p>Đây là một đoạn văn bản. Chúng ta đang sử dụng CSS bên ngoài để định dạng nó.</p>

<button>Nhấn vào đây</button>

</body>

</html>

## Ngôn ngữ Javascript

### Giới thiệu về Javascript

JavaScript là một ngôn ngữ lập trình kịch bản (scripting language) được phát triển vào năm 1995 bởi Brendan Eich, một nhà phát triển tại Netscape. Nó được thiết kế để chạy trên trình duyệt web, cho phép tạo ra các trang web động, tương tác và ứng dụng web.

Một số đặc điểm chính của JavaScript:

* Ngôn ngữ lập trình kịch bản: JavaScript được sử dụng để thêm tính năng động và tương tác vào các trang web. Nó chạy trên trình duyệt web, cho phép thực hiện các tác vụ như xử lý sự kiện, thay đổi nội dung trang web, tạo hiệu ứng, gửi yêu cầu AJAX, v.v.
* Ngôn ngữ lập trình đa năng: Ngoài việc sử dụng trên trình duyệt web, JavaScript còn có thể được sử dụng trên máy chủ (Node.js), trong ứng dụng di động (React Native), trong trò chơi (Unity), và nhiều ứng dụng khác.
* Ngôn ngữ lập trình động: JavaScript là ngôn ngữ lập trình động, có nghĩa là các biến có thể được khai báo mà không cần chỉ định kiểu dữ liệu cụ thể. Điều này giúp code trở nên linh hoạt và dễ dàng thích ứng với các yêu cầu thay đổi.
* Ngôn ngữ lập trình đơn giản: Mặc dù JavaScript có nhiều tính năng nâng cao, nhưng nó vẫn được coi là một ngôn ngữ lập trình tương đối đơn giản, dễ học và sử dụng, đặc biệt đối với những người mới bắt đầu lập trình web.
* Hỗ trợ lập trình hướng đối tượng: Mặc dù JavaScript không phải là một ngôn ngữ lập trình hướng đối tượng thuần túy, nhưng nó cung cấp các tính năng hướng đối tượng như lớp, đối tượng, kế thừa, v.v.

JavaScript đóng vai trò rất quan trọng trong việc tạo ra các trang web và ứng dụng web hiện đại, cung cấp khả năng tương tác, hiệu ứng và tính năng động cho người dùng. Nó là một trong những ngôn ngữ lập trình phổ biến và được sử dụng rộng rãi nhất trên thế giới.

### Thêm Javascript vào trang

Để thêm JavaScript vào một trang web, có hai cách chính:

* Sử dụng thẻ <script>:

- Bạn có thể nhúng mã JavaScript trực tiếp vào trang HTML bằng cách sử dụng thẻ <script>.

- Ví dụ:

<script>

// Mã JavaScript ở đây

console.log("Hello, World!");

</script>

* Sử dụng tệp JavaScript riêng biệt:

- Bạn có thể tạo một tệp JavaScript riêng biệt (thường có phần mở rộng .js) và nhúng nó vào trang HTML.

- Ví dụ:

<script src="path/to/your-script.js"></script>

Một số lưu ý khi thêm JavaScript vào trang:

* Vị trí của thẻ <script>: Thông thường, các thẻ <script> được đặt ở cuối phần <body> của trang HTML để tránh ảnh hưởng đến việc tải trang.
* Tải JavaScript bất đồng bộ: Bạn có thể thêm thuộc tính "async" hoặc "defer" vào thẻ <script> để tải JavaScript bất đồng bộ, tăng tốc độ tải trang.
* Sử dụng các thư viện và framework: Bạn có thể sử dụng các thư viện và framework JavaScript như jQuery, React, Angular, Vue.js, v.v. bằng cách nhúng chúng vào trang.
* Quản lý các tệp JavaScript: Khi dự án lớn, bạn nên tổ chức các tệp JavaScript vào các thư mục riêng biệt để dễ quản lý và bảo trì.

Bằng cách sử dụng các kỹ thuật trên, bạn có thể dễ dàng thêm JavaScript vào trang web của mình, tạo ra các trang web động, tương tác và ứng dụng web hiện đại.

### Javascript cơ bản

Tổng quan về các chủ đề cơ bản trong JavaScript:

* Biến và kiểu dữ liệu:

- Khai báo biến sử dụng từ khóa var, let hoặc const.

- Các kiểu dữ liệu cơ bản: number, string, boolean, undefined, null, object, symbol.

* Toán tử và biểu thức:

- Các toán tử số học, gán, so sánh, logic.

- Sử dụng các biểu thức để thực hiện các phép tính.

* Câu lệnh điều khiển:

- Câu lệnh if-else, switch.

- Vòng lặp for, while, do-while.

* Hàm:

- Khai báo và gọi hàm.

- Tham số và giá trị trả về.

- Hàm mũi tên (arrow function).

* Mảng:

- Khai báo và truy cập các phần tử trong mảng.

- Các phương thức thông dụng như push, pop, slice, concat.

* Đối tượng:

- Khai báo và truy cập các thuộc tính và phương thức của đối tượng.

- Sử dụng từ khóa this.

* Sự kiện:

- Lắng nghe và xử lý các sự kiện trên trang web như click, keypress, submit.

* DOM (Document Object Model):

- Truy cập và thao tác các phần tử HTML bằng JavaScript.

- Sử dụng các phương thức như getElementById, querySelector, appendChild.

* Bất đồng bộ và Promises:

- Xử lý các tác vụ bất đồng bộ như AJAX, setTimeout.

- Sử dụng Promises để quản lý luồng thực thi.

* Modules:

- Chia nhỏ code thành các module riêng biệt.

- Sử dụng import và export để tái sử dụng code.

### Variables(Biến)

Biến (variable) là một khái niệm cơ bản trong lập trình, bao gồm cả JavaScript. Biến được sử dụng để lưu trữ và thao tác dữ liệu trong chương trình.

Khai báo biến trong JavaScript:

* Sử dụng từ khóa `var`, `let` hoặc `const` để khai báo biến.
* Ví dụ:

var name = "John";

let age = 30;

const PI = 3.14159;

Các quy tắc đặt tên biến:

* Tên biến phải bắt đầu bằng một chữ cái, dấu gạch dưới (`\_`) hoặc dấu đô la (`$`).
* Tên biến có thể chứa chữ cái, số, dấu gạch dưới hoặc dấu đô la.
* Tên biến phân biệt chữ hoa/chữ thường (case-sensitive).
* Nên đặt tên biến theo cách có ý nghĩa, dễ hiểu.

Phạm vi của biến (Scope):

* `var`: Biến có phạm vi toàn cục (global) hoặc phạm vi hàm (function scope).
* `let` và `const`: Biến có phạm vi khối (block scope), tức là chỉ tồn tại trong cặp dấu `{}` chứa nó.

Sử dụng biến:

* Gán giá trị cho biến: `name = "John";`
* Truy cập giá trị của biến: `console.log(name);`
* Cập nhật giá trị của biến: `age = 31;`

Lưu ý:

* Sử dụng `const` để khai báo biến hằng số, không thể thay đổi giá trị.
* Tránh sử dụng `var` vì nó có phạm vi toàn cục, dễ gây ra lỗi. Thay vào đó, sử dụng `let` hoặc `const`.
* Khai báo biến trước khi sử dụng để tránh lỗi.

### Data types(Loại dữ liệu)

Trong JavaScript, các kiểu dữ liệu (data types) là những loại dữ liệu cơ bản mà biến có thể lưu trữ. Có 7 kiểu dữ liệu chính trong JavaScript:

* \*\*Number\*\*: Kiểu số, bao gồm cả số nguyên và số thực. Ví dụ: `42`, `3.14`, `-10`.
* \*\*String\*\*: Kiểu chuỗi ký tự, được bao quanh bởi dấu nháy đơn `'`, dấu nháy kép `"` hoặc dấu backtick `` ` ``. Ví dụ: `"Hello, world!"`, `'JavaScript'`, `\`Multiline\n string\```.
* \*\*Boolean\*\*: Kiểu logic, chỉ có hai giá trị là `true` hoặc `false`.
* \*\*Undefined\*\*: Giá trị mặc định của biến chưa được gán giá trị.
* \*\*Null\*\*: Giá trị đại diện cho sự vắng mặt hoặc không tồn tại của giá trị.
* \*\*Object\*\*: Kiểu dữ liệu phức tạp, có thể chứa nhiều cặp key-value. Ví dụ: `{ name: "John", age: 30 }`.
* \*\*Symbol\*\*: Kiểu dữ liệu duy nhất, được sử dụng làm khóa (key) cho các thuộc tính của đối tượng.

Ngoài ra, còn có một số kiểu dữ liệu đặc biệt khác như:

* \*\*Array\*\*: Một đối tượng đặc biệt, có thể chứa nhiều giá trị khác nhau.
* \*\*Function\*\*: Một đối tượng có thể được gọi và thực thi.
* \*\*Date\*\*: Đại diện cho một ngày/thời gian cụ thể.
* \*\*RegExp\*\*: Đại diện cho một mẫu (pattern) để so khớp chuỗi.

Để kiểm tra kiểu dữ liệu của một biến, bạn có thể sử dụng toán tử `typeof`. Ví dụ:

typeof 42; // "number"

typeof "Hello"; // "string"

typeof true; // "boolean"

typeof undefined; // "undefined"

typeof null; // "object" (đây là một lỗi trong JavaScript)

typeof {}; // "object"

typeof []; // "object"

typeof function() {}; // "function"

### Comparison Operators(Toán tử so sánh)

Các toán tử so sánh (comparison operators) trong JavaScript được sử dụng để so sánh hai giá trị và trả về kết quả là `true` hoặc `false`. Các toán tử so sánh chính bao gồm:

* \*\*Toán tử bằng (==)\*\*: So sánh hai giá trị về mặt giá trị, không so sánh kiểu dữ liệu.

- Ví dụ: `5 == "5"` sẽ trả về `true` vì cả hai đều có giá trị là `5`.

* \*\*Toán tử bằng nghiêm ngặt (===)\*\*: So sánh cả giá trị và kiểu dữ liệu.

- Ví dụ: `5 === "5"` sẽ trả về `false` vì mặc dù giá trị bằng nhau, nhưng kiểu dữ liệu khác nhau (số và chuỗi).

* \*\*Toán tử không bằng (!=)\*\*: So sánh hai giá trị, không so sánh kiểu dữ liệu.

- Ví dụ: `5 != "5"` sẽ trả về `false` vì cả hai đều có giá trị là `5`.

* \*\*Toán tử không bằng nghiêm ngặt (!==)\*\*: So sánh cả giá trị và kiểu dữ liệu.

- Ví dụ: `5 !== "5"` sẽ trả về `true` vì mặc dù giá trị bằng nhau, nhưng kiểu dữ liệu khác nhau.

* \*\*Toán tử lớn hơn (>)\*\*: So sánh xem giá trị bên trái có lớn hơn giá trị bên phải hay không.

- Ví dụ: `10 > 5` sẽ trả về `true`.

* \*\*Toán tử lớn hơn hoặc bằng (>=)\*\*: So sánh xem giá trị bên trái có lớn hơn hoặc bằng giá trị bên phải hay không.

- Ví dụ: `10 >= 10` sẽ trả về `true`.

* \*\*Toán tử nhỏ hơn (<)\*\*: So sánh xem giá trị bên trái có nhỏ hơn giá trị bên phải hay không.

- Ví dụ: `5 < 10` sẽ trả về `true`.

* \*\*Toán tử nhỏ hơn hoặc bằng (<=)\*\*: So sánh xem giá trị bên trái có nhỏ hơn hoặc bằng giá trị bên phải hay không.

- Ví dụ: `5 <= 5` sẽ trả về `true`.

### Mathematical operators (Toán tử toán học)

Các toán tử toán học (mathematical operators) trong JavaScript được sử dụng để thực hiện các phép tính số học. Các toán tử toán học chính bao gồm:

* \*\*Toán tử cộng (+)\*\*: Dùng để cộng hai số hoặc nối hai chuỗi.

- Ví dụ: `5 + 3` sẽ trả về `8`, `"Hello" + " " + "world"` sẽ trả về `"Hello world"`.

* \*\*Toán tử trừ (-)\*\*: Dùng để trừ hai số.

- Ví dụ: `10 - 4` sẽ trả về `6`.

* \*\*Toán tử nhân (\*)\*\*: Dùng để nhân hai số.

- Ví dụ: `7 \* 6` sẽ trả về `42`.

* \*\*Toán tử chia (/)\*\*: Dùng để chia hai số.

- Ví dụ: `15 / 3` sẽ trả về `5`.

* \*\*Toán tử chia lấy phần dư (%)\*\*: Dùng để lấy phần dư khi chia hai số.

- Ví dụ: `10 % 3` sẽ trả về `1`.

* \*\*Toán tử lũy thừa (\*\*)\*\*: Dùng để tính lũy thừa của một số.

- Ví dụ: `2 \*\* 3` sẽ trả về `8` (2 mũ 3).

* \*\*Toán tử tăng (++)\*\*: Dùng để tăng giá trị của biến lên 1.

- Ví dụ: `let x = 5; x++;` sẽ khiến `x` có giá trị `6`.

* \*\*Toán tử giảm (--)\*\*: Dùng để giảm giá trị của biến đi 1.

- Ví dụ: `let y = 10; y--;` sẽ khiến `y` có giá trị `9`.

### if/else statements(câu lệnh if/else)

Trong JavaScript, if/else statements là một trong những cấu trúc điều khiển cơ bản. Chúng được sử dụng để thực hiện các quyết định dựa trên các điều kiện nhất định.

Cú pháp cơ bản của if/else statements như sau:

if (condition) {

// code block to be executed if the condition is true

} else {

// code block to be executed if the condition is false

}

Trong đó:

* `condition` là một biểu thức logic (true/false) mà JavaScript sẽ đánh giá.
* Nếu `condition` là `true`, khối code bên trong `if` sẽ được thực thi.
* Nếu `condition` là `false`, khối code bên trong `else` sẽ được thực thi.

Ví dụ:

let age = 18;

if (age >= 18) {

console.log("Bạn đủ tuổi để bỏ phiếu.");

} else {

console.log("Bạn chưa đủ tuổi để bỏ phiếu.");

}

### Loops(vòng lặp)

Trong JavaScript, có các loại vòng lặp (loops) sau:

* \*\*for loop\*\*:

- Cú pháp: `for (initialization; condition; increment/decrement) { // code block }`

- Dùng để lặp một số lần xác định trước.

- Ví dụ: `for (let i = 0; i < 5; i++) { console.log(i); }`

* \*\*while loop\*\*:

- Cú pháp: `while (condition) { // code block }`

- Dùng để lặp khi điều kiện là true.

- Ví dụ: `let i = 0; while (i < 5) { console.log(i); i++; }`

* \*\*do-while loop\*\*:

- Cú pháp: `do { // code block } while (condition);`

- Tương tự while loop, nhưng luôn chạy ít nhất một lần.

- Ví dụ: `let i = 0; do { console.log(i); i++; } while (i < 5);`

* \*\*for-in loop\*\*:

- Cú pháp: `for (variable in object) { // code block }`

- Dùng để lặp qua các thuộc tính của một đối tượng.

- Ví dụ: `const person = { name: 'John', age: 30, city: 'New York' }; for (let prop in person) { console.log(prop, person[prop]); }`

* \*\*for-of loop\*\*:

- Cú pháp: `for (variable of iterable) { // code block }`

- Dùng để lặp qua các phần tử của một đối tượng có thể lặp lại, như mảng, chuỗi, v.v.

- Ví dụ: `const numbers = [1, 2, 3, 4, 5]; for (let num of numbers) { console.log(num); }`

### Functions(chức năng)

Các hàm (functions) trong JavaScript là một trong những khái niệm cốt lõi và quan trọng nhất của ngôn ngữ này. Chúng cho phép bạn nhóm các đoạn code lại với nhau, tái sử dụng chúng, và tổ chức code của bạn theo cách có ý nghĩa. Dưới đây là một số đặc điểm chính của các hàm trong JavaScript:

* \*\*Khai báo hàm\*\*:

- Cú pháp: `function functionName(parameters) { // code block }`

- Ví dụ: `function greet(name) { console.log("Hello, " + name + "!"); }`

* \*\*Gọi hàm\*\*:

- Để sử dụng hàm, bạn gọi nó bằng cách sử dụng tên hàm và truyền các tham số (nếu có).

- Ví dụ: `greet("John");`

* \*\*Tham số (parameters)\*\*:

- Các biến được định nghĩa trong danh sách tham số của hàm.

- Ví dụ: `function add(a, b) { return a + b; }`

* \*\*Giá trị trả về (return)\*\*:

- Từ khóa `return` được sử dụng để trả về giá trị từ hàm.

- Ví dụ: `function square(x) { return x \* x; }`

* \*\*Hàm vô danh (anonymous functions)\*\*:

- Hàm không có tên, thường được gán cho một biến hoặc sử dụng như một tham số của hàm khác.

- Ví dụ: `let greet = function(name) { console.log("Hello, " + name + "!"); };`

* \*\*Hàm mũi tên (arrow functions)\*\*:

- Cú pháp ngắn gọn hơn so với hàm truyền thống.

- Ví dụ: `let square = (x) => x \* x;`

Các hàm trong JavaScript rất linh hoạt và có thể được sử dụng để tổ chức code, tái sử dụng code, và tăng tính modular của ứng dụng. Chúng giúp code trở nên dễ đọc, dễ bảo trì và dễ mở rộng hơn./-strong/-heart:>:o:-((:-h

### Variable Scope and the var Keyword(Phạm vi biến và từ khóa var)

Trong JavaScript, phạm vi biến (variable scope) và từ khóa `var` là những khái niệm quan trọng liên quan đến cách biến được định nghĩa và truy cập.

* \*\*Phạm vi biến (Variable Scope)\*\*:

- Phạm vi biến xác định nơi biến có thể được truy cập và sử dụng trong code.

- Có hai loại phạm vi chính:

- \*\*Phạm vi toàn cục (Global Scope)\*\*: Biến được khai báo ngoài bất kỳ hàm hoặc khối code nào. Chúng có thể được truy cập từ bất kỳ nơi nào trong code.

- \*\*Phạm vi cục bộ (Local Scope)\*\*: Biến được khai báo bên trong một hàm hoặc khối code. Chúng chỉ có thể được truy cập trong phạm vi của hàm hoặc khối code đó.

* \*\*Từ khóa `var`\*\*:

- `var` là từ khóa được sử dụng để khai báo biến trong JavaScript.

- Khi sử dụng `var`, biến sẽ có phạm vi cục bộ nếu được khai báo bên trong một hàm, hoặc phạm vi toàn cục nếu được khai báo bên ngoài bất kỳ hàm nào.

- Ví dụ:

```javascript

var globalVar = "I am global";

function myFunction() {

var localVar = "I am local";

console.log(globalVar); // Truy cập biến toàn cục

console.log(localVar); // Truy cập biến cục bộ

}

myFunction();

console.log(globalVar); // Truy cập biến toàn cục

console.log(localVar); // Lỗi: localVar is not defined

```

Lưu ý rằng, từ ES6 (ECMAScript 2015), các từ khóa `let` và `const` được giới thiệu, cung cấp cách khai báo biến với phạm vi chính xác hơn so với `var`. Tuy nhiên, `var` vẫn được sử dụng rộng rãi, đặc biệt trong các dự án cũ hơn.

### THE BROWSER OBJECT (ĐỐI TƯỢNG TRÌNH DUYỆT)

Trong JavaScript, đối tượng trình duyệt (browser object) là một tập hợp các đối tượng, thuộc tính và phương thức được cung cấp bởi trình duyệt web, cho phép tương tác với môi trường trình duyệt. Một số đối tượng và tính năng chính của đối tượng trình duyệt bao gồm:

* \*\*Window Object\*\*:

- Đây là đối tượng gốc của trình duyệt, đại diện cho cửa sổ trình duyệt.

- Cung cấp các phương thức và thuộc tính để tương tác với cửa sổ trình duyệt, như `window.alert()`, `window.innerWidth`, `window.scrollTo()`, v.v.

* \*\*Document Object\*\*:

- Đại diện cho tài liệu HTML đang được hiển thị trong trình duyệt.

- Cung cấp các phương thức và thuộc tính để truy cập và thao tác với nội dung trang web, như `document.getElementById()`, `document.createElement()`, `document.body`, v.v.

* \*\*Navigator Object\*\*:

- Cung cấp thông tin về trình duyệt và hệ điều hành của người dùng.

- Các thuộc tính như `navigator.userAgent`, `navigator.language`, `navigator.platform`, v.v.

* \*\*Screen Object\*\*:

- Cung cấp thông tin về màn hình của thiết bị người dùng.

- Các thuộc tính như `screen.width`, `screen.height`, `screen.colorDepth`, v.v.

* \*\*History Object\*\*:

- Cung cấp thông tin về lịch sử điều hướng của người dùng.

- Các phương thức như `history.back()`, `history.forward()`, `history.go()`.

* \*\*Location Object\*\*:

- Cung cấp thông tin về URL hiện tại và cho phép điều hướng đến các URL khác.

- Các thuộc tính như `location.href`, `location.protocol`, `location.host`, `location.pathname`, v.v.

### Events

Trong JavaScript, sự kiện (events) là một cơ chế quan trọng cho phép tương tác giữa người dùng và trang web. Các sự kiện được kích hoạt khi một hành động nhất định xảy ra, chẳng hạn như người dùng nhấp chuột, nhấn phím, di chuyển chuột, v.v. Khi một sự kiện được kích hoạt, bạn có thể viết code để xử lý sự kiện đó.

Một số ví dụ về các sự kiện phổ biến trong JavaScript:

* \*\*Sự kiện chuột (Mouse Events)\*\*:

- `click`: Khi người dùng nhấp chuột.

- `mouseover`: Khi con trỏ chuột di chuyển vào một phần tử.

- `mouseout`: Khi con trỏ chuột di chuyển ra khỏi một phần tử.

- `mousedown`: Khi người dùng nhấn chuột xuống.

- `mouseup`: Khi người dùng nhả chuột ra.

* \*\*Sự kiện bàn phím (Keyboard Events)\*\*:

- `keydown`: Khi người dùng nhấn một phím.

- `keyup`: Khi người dùng nhả một phím.

- `keypress`: Khi người dùng nhấn và nhả một phím.

* \*\*Sự kiện form (Form Events)\*\*:

- `submit`: Khi người dùng gửi một biểu mẫu.

- `change`: Khi giá trị của một phần tử biểu mẫu thay đổi.

- `focus`: Khi một phần tử biểu mẫu nhận được sự tập trung.

- `blur`: Khi một phần tử biểu mẫu mất sự tập trung.

* \*\*Sự kiện trang (Document/Window Events)\*\*:

- `load`: Khi trang web được tải xong.

- `resize`: Khi kích thước cửa sổ trình duyệt thay đổi.

- `scroll`: Khi người dùng cuộn trang.

- `DOMContentLoaded`: Khi DOM (Document Object Model) được tải xong.

Để xử lý các sự kiện, bạn có thể sử dụng phương thức `addEventListener()` để gắn một hàm xử lý sự kiện vào một phần tử. Khi sự kiện được kích hoạt, hàm xử lý sẽ được gọi và có thể thực hiện các hành động tương ứng.

Ví dụ:

const button = document.getElementById('myButton');

button.addEventListener('click', function() {

console.log('Button was clicked!');

});

### DOM

### DOM là gì?

**DOM** có tên gọi tắt là **Document Object Model**, được dịch là mô hình các đối tượng tài liệu, sử dụng với mục đích truy xuất hoặc thao tác cùng[tài liệu HTML](https://vietnix.vn/html-la-gi/)và [XML](https://vietnix.vn/xml-la-gi/) trong các ngôn ngữ lập trình thông dụng như [JavaScript](https://vietnix.vn/javascript-la-gi/), [PHP](https://vietnix.vn/php-la-gi/),….

Theo W3C (Tổ chức Web Toàn Cầu – World Wide Web Consortium), tiêu chuẩn kỹ thuật này cần thống nhất để DOM có mô hình nhất định. Tuy nhiên, DOM vẫn hoạt động độc lập và chỉ dựa vào kỹ thuật mô tả để lập trình đối tượng cụ thể.

## HTML DOM là gì?

**HTML DOM** có thể hiểu đơn giản là công cụ giúp thao tác thay đổi [dữ liệu](https://vietnix.vn/du-lieu-la-gi/) tài liệu theo đối tượng mô hình. Bên trong 1 tài liệu bao gồm các cấu trúc như: đối tượng, phương thức hoặc thuộc tính được định nghĩa phần tử riêng nhằm mục đích hỗ trợ truy xuất đơn giản hơn mà vẫn đảm bảo cấu trúc. Các phần tử này đều là một đối tượng hoặc chủ sở hữu, phương thức để làm việc cùng thuộc tính các thao tác như: xóa, sửa, thêm, cập nhật.

Ngoài ra, người dùng có thể tùy ý thêm và xóa các phần tử theo mong muốn để đảm bảo nội dung và cấu trúc truy cập web của mình luôn luôn được cập nhật, làm mới.

## Chi tiết về cây cấu trúc trong DOM

Các yếu tố cấu thành nên cấu trúc của DOM được chia tách thành dạng nhánh cây như sau:

### Node (Nút)

Phần tử bên trong DOM được chia tách thành dạng cây, chính vì vậy, mô hình bên trong mỗi thành phần nhỏ được gọi là các Node. Các Node trong đây được chia thành nhiều phân loại khác nhau. Tuy nhiên, [lập trình viên](https://vietnix.vn/lap-trinh-vien-la-gi/) cần chú ý đặc biệt hơn ở 3 Node chính: Node gốc, Node phần tử và Node văn bản.

Cụ thể như sau:

* **Node gốc:**Là Node chính hay còn gọi là Node HTML tài liệu gốc, thường được sử dụng bởi thẻ <html>.
* **Node phần tử:**Dùng để hiển thị cho 1 phần tử HTML riêng.
* **Node văn bản:**Hay còn gọi là Node Text, được gọi là những đoạn kí tự bên trong HTML. Khi biểu diễn Node Text thì có thể hoạt động qua tên một trang web <title>, tên đề mục chính <h1> hoặc là một đoạn văn bản <p>.

### Quan hệ giữa các Node

* Ở phần đầu tiên là Node gốc (Document).
* Nếu Node đó không phải là Node gốc, tất cả Node còn lại có thể hiểu là Node cha (Node Parent).
* Một Node có thể chứa một đến nhiều con trong một Node hoặc có thể rơi vào trạng thái không có con nào trong Node.
* Những Node cùng trong một Node cha thì được gọi là Node anh em (Siblings).
* Các Node anh em trong một cha, Node đầu được gọi là con cả, Node cuối cùng được gọi là con út (F irstChild).

## Nhiệm vụ DOM trong JavaScript

Ta có thể tổng hợp 4 nhiệm vụ cơ bản của DOM trong JavaScript như sau:

1. Truy xuất đến các thẻ HTML được thông qua.
2. Thay đổi thuộc tính của HTML.
3. Thay đổi [CSS](https://vietnix.vn/css-la-gi/) của HTML.
4. Xóa, sửa, thêm hoặc tạo trên HTML.

## Phân loại DOM trong JavaScript

Để xử lý nhanh chóng HTML dễ dàng hơn, JavaScript phân DOM ra thành nhiều phân loại khác nhau, cụ thể:

* **DOM document:**Các thành phần có trong document và trang web sẽ được lưu trữ tại đây.
* **DOM element:** Nhiệm vụ chủ yếu là truy xuất HTML bằng cách xác định các thuộc tính như: Class, ID hoặc name có trong HTML.
* **DOM HTML:** Hỗ trợ thay đổi nội dung hoặc giá trị thuộc tính trong HTML.
* **DOM CSS:**Các thẻ HTML sẽ được thay đổi định dạng CSS.
* **DOM Event:**Mục đích gán các dữ liệu như onload(), onclick() và các HTML.
* **DOM Listener:** Giúp nghe hiểu sự kiện nào đã tác động vào thẻ HTML.
* **DOM Navigation:**Dùng để thể hiện mối quan hệ cha con có trong HTML và trực tiếp quản lý các thao tác tại đây.
* **DOM Node, Nodelist:** Được sử dụng để thao tác với HTML thông qua đối tượng.

## Thao tác với DOM trong JavaScript

Thao tác với DOM trong JavaScript được ví như việc có thể thay đổi thế giới, bởi thông qua DOM bạn có thể thao tác và cập nhật các nội dung tại đây. Mọi thay đổi bạn có thể làm như: định dạng chữ, nội dung, cấu trúc Node, thêm, xóa, sửa,… đều được hỗ trợ.

### Các thuộc tính và phương thức thường gặp

Cùng tìm hiểu thuộc tính và phương thức thường gặp để thao tác với DOM dễ dàng hơn:

#### Thuộc tính:

* **ID:** Là thuộc tính một phần tử nên thường xuyên được dùng truy xuất DOM trực tiếp.
* **ClassName:** Giống như ID, className cũng dùng truy xuất DOM nhưng dùng được trong nhiều phần tử hơn.
* **TagName:**Tên thẻ của HTML.
* **InnerHTML:**Hoàn về mã HTML có trong phần tử hiện tại, chứa tất cả các phần tử bên trong dưới dạng chuỗi ký tự và bao gồm cả phần tử và Node văn bản.
* **OuterHTML:**Hoàn về mã HTML có trong phần tử hiện tại, có thể nói rằng outerHTML = tagName + innerHTML.
* **TextContent:** Trả về mã chuỗi ký tự mà trong đó có chứa nội dung của Node văn bản hiện có.
* **Attributes:** Tập các thuộc tính như ID, Name, Class, Href, Title,…
* **Style:**Tập các định dạng của phần tử hiện tại.
* **Value:** Lấy giá trị của thành phần được chọn thành một biến.

#### Phương thức:

* **GetElementById (ID):**Phương thức truy xuất đến 1 phần tử duy nhất thông qua ID của nó.
* **GetElementsByTagName (Tagname):**Truy xuất đến tất cả các phần tử HTML thông qua name trong thẻ cần tìm.
* **GetElementsByName (Name):**Truy xuất đến tất cả các Node có thuộc tính name cần tìm.
* **GetAttribute (Attribute Name):** Lấy giá trị của thuộc tính.
* **SetAttribute (Attribute Name, value):**Sửa giá trị của thuộc tính.
* **AppendChild (Node):**Dùng khi thêm tạo 1 Node con vào Node hiện tại.
* **RemoveChild (Node):**Xóa 1 Node nào đó khỏi Node hiện tại.

Có thể hiểu theo cách khác, cây cấu trúc của DOM đều chứa các phần tử con của DOM. Dựa vào mối quan hệ tương tác này, ta có thể dễ dàng truy xuất DOM thông qua quan hệ và vị trí phần tử HTML.

**Các thuộc tính quan hệ dưới đây nhằm biểu diễn sự tương tác và phụ thuộc lẫn nhau của các Node:**

* **ParentNode:**Node cha.
* **ChildNodes:** Các Node con.
* **FirstChild:**Node con đầu tiên.
* **LastChild:** Node con cuối cùng.
* **NextSibling:**Node anh em liền kề sau.
* **PreviousSibling:** Node anh em liền kề trước.

### Truy xuất trong DOM

#### Truy xuất gián tiếp

Trong mỗi Node tạo nên cấu trúc cây của DOM có 6 thuộc tính quan hệ nhằm hỗ trợ truy xuất dựa trên vị trí Node một cách gián tiếp:

* **ParentNode:**Đề cập đến Node cha của Node hiện tại, là Node duy nhất. Do đó, nếu bạn cần tìm nguồn gốc chính của một Node, bạn phải nối các Node lại nhiều lần. Ví dụ: Node.parentNode.parentNode.
* **ChildNodes:** Đề cập đến các Node con ngay lập tức của Node hiện tại và tạo ra một mảng các đối tượng. Lưu ý rằng, các Node con không được phân biệt theo loại Node, vì vậy kết quả mảng trả về có thể chứa nhiều loại Node khác nhau.
* **FirstChild:**Tham chiếu đến Node con đầu tiên của Node hiện tại, tương đương với việc gọi Node.childNodes [0].
* **LastChild:** Đề cập đến Node con cuối cùng của Node hiện tại, tương đương với việc gọi Node.childNodes [Element.childNodes.length-1].
* **NextSibling:** Tham chiếu đến một Node anh em kề với Node hiện tại.
* **PreviousSibling:** Tham chiếu đến Node anh em trước đó của Node hiện tại.

#### Truy xuất trực tiếp

Khác với truy xuất gián tiếp, việc sử dụng truy xuất trực tiếp sẽ tiết kiệm thời gian, giúp bạn không cần hiểu biết quá rõ về các Node trong DOM.

**3 phương thức khi truy xuất trực tiếp phù hợp cho bạn lựa chọn bao gồm:**

* document.getElementById(‘id\_cần\_tìm’).
* document.getElementsByTagName(‘div’).
* document.getElementsByName(‘tên\_cần\_tìm’).

## Kết chương

Qua chương này, chúng ta đã tìm hiểu về HTML, CSS và JavaScript, và hiểu được tầm quan trọng và đóng vai trò nhất định trong lập trình website.

HTML là nền tảng cơ bản để xây dựng và cấu trúc các trang web, đóng vai trò quan trọng trong việc trình bày, tương tác và cải thiện khả năng truy cập cũng như tối ưu hóa công cụ tìm kiếm cho các trang web.

CSS đóng vai trò quan trọng trong việc tách biệt nội dung và trình bày, đảm bảo tính nhất quán, tối ưu hóa hiệu suất, tính di động và đa nền tảng, cũng như tối ưu hóa SEO cho các trang web.

JavaScript đóng vai trò quan trọng trong việc tạo ra các trang web động, tương tác, cải thiện trải nghiệm người dùng, xử lý logic phía client, xây dựng ứng dụng web đơn trang, tích hợp với các API và tối ưu hóa SEO. Nó đã trở thành một ngôn ngữ lập trình không thể thiếu trong phát triển web hiện đại

Qua đó, nếu ta vận dụng được 3 thành phân trên thì sẽ tạo nên một trang web đa chức năng và tiện ích.

# Phân tích thiết kế hệ thống

## Mô tả bài toán

## Đặc tả các yêu cầu

### Yêu cầu chức năng

### Yêu cầu phi chức năng

## Thiết kế dữ liệu

### Mô hình dữ liệu

Trình bày mô hình ERD

### Danh sách các thực thể và mối kết hợp

Bảng . Danh sách các thực thể và mối kết hợp

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Tên thực thể/Mối kết hợp | Diễn giải | Ghi chú |
|  |  |  |  |
|  |  |  |  |

### Chi tiết thực thể và mối kết hợp

Thực thể

Mô tả

Chi tiết thực thể

Bảng . Chi tiết thực thể …

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| STT | Tên thuộc tính | Diễn giải | Kiểu dữ liệu | Ràng buộc toàn vẹn |
|  |  |  |  |  |
|  |  |  |  |  |

## Thiết kế xử lý

### Sơ đồ DFD mức ngữ cảnh

### Sơ đồ DFD mức 1

### Sơ đồ DFD mức 2 (Theo từng chức năng)

Diễn giải từng bước:

(1)

(2)

(3)

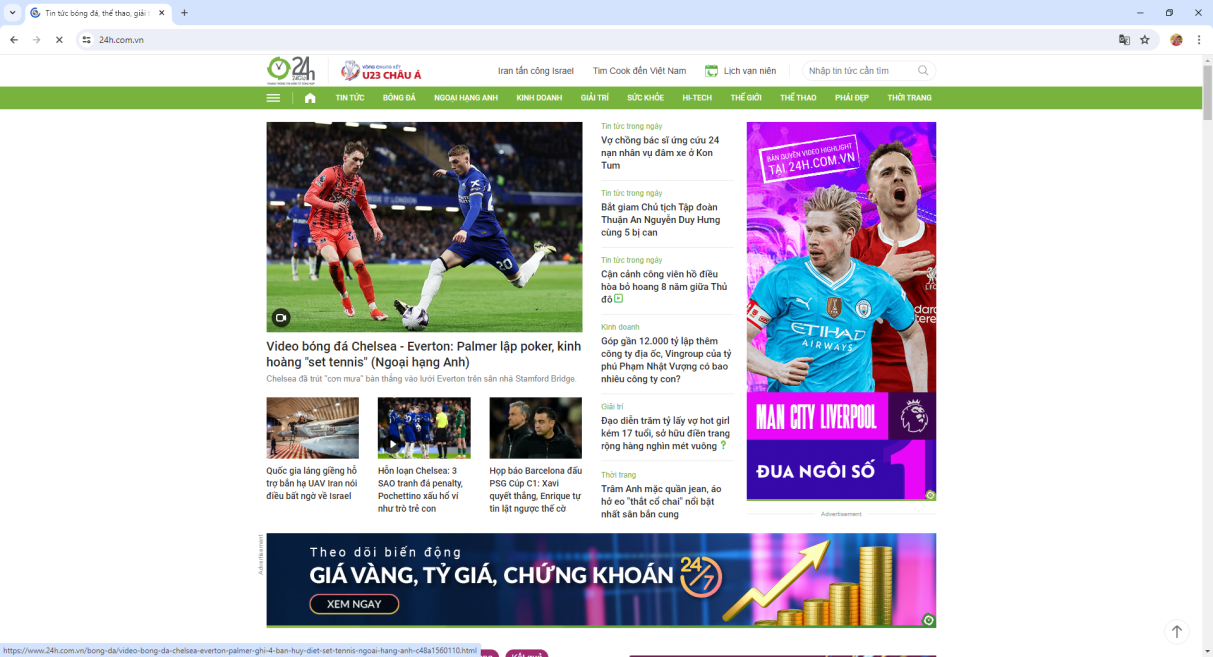
(4)

## Thiết kế giao diện

### Sơ đồ hệ thống

Hình . Sơ đồ hệ thống

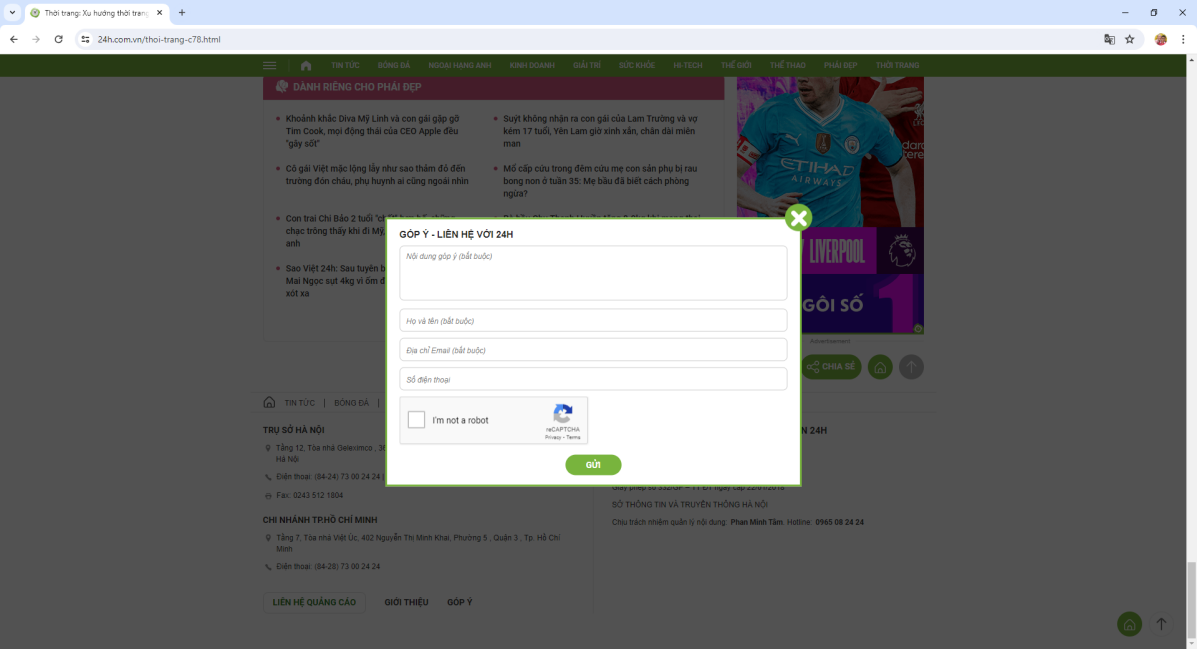
### Giao diện trang chủ người dùng



Hình . Giao diện trang chủ của người dùng

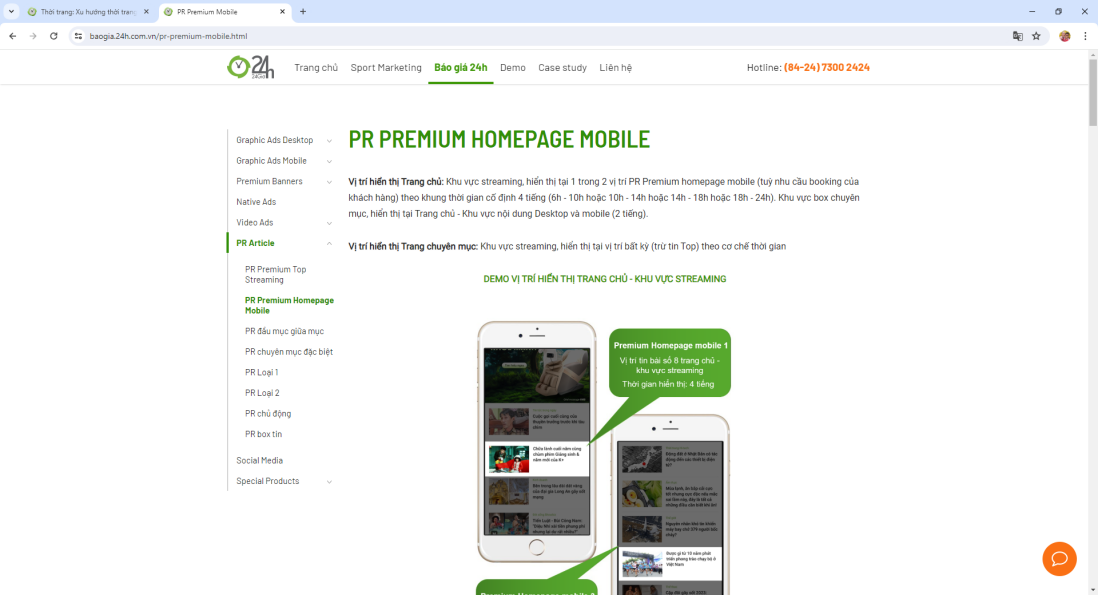
Mô tả chi tiết về màn hình bao gồm các thành phần và cách bố cục

#### Giao diện chức năng …



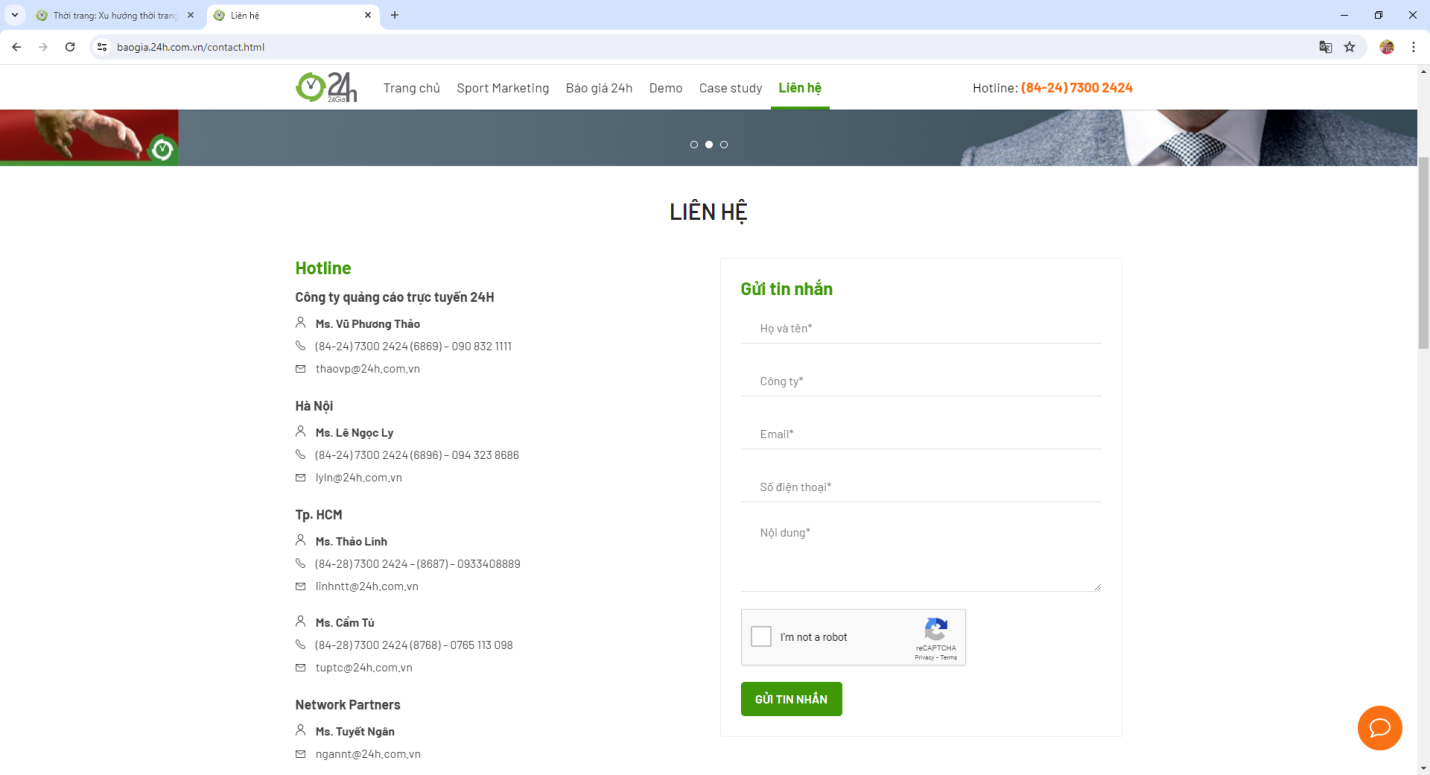
Hình . Giao diện chức năng …

#### Giao diện chức năng …



Hình . Hình …

#### Giao diện chức năng …



### Giao diện trang chủ quản trị

#### Giao diện chức năng …

#### Giao diện chức năng …

#### Giao diện chức năng …

### Kết chương

# Cài đặt thực nghiệm

## Bộ dữ liệu thử nghiệm

## Kết quả thực nghiệm

### Chức năng …

### Chức năng …

### Chức năng

## Kết chương

# Kết luận và hướng phát triển

## Kết luận

## Hướng phát triển

**DANH MỤC TÀI LIỆU THAM KHẢO**

Trình bày theo quy định của chuẩn IEEE