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 HÀNG DÀNH RIÊNG CHO CÁ NHÂN**

|  |  |  |
| --- | --- | --- |
| **Giáo viên hướng dẫn:**  Nguyễn Ngọc Đan Thanh | | **Sinh viên thực hiện:**  Mã số sinh viên: 110122100  Họ tên: Huỳnh Quốc Kiệt  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:

* Xác định mục tiêu của Website muốn thực hiện
* Lê kế hoạch thực hiện từng bước
* Phân tích mục tiêu và đối tượng đang muốn hướng đến
* Lập kế hoạch cho các chức năng và nội dung của website.
* Thu thập thông tin đáp ứng đủ yêu cầu từ khách hàng.
* Thu Thập thêm các thông tin của các Mẫu website bán hàng

### Thiết kế giao diện:

* Nghiên cứu người dùng
* xác định cấu trúc của website:
* Màu sắc
* Font chữ
* Hình ảnh và biểu đồ
* Bố cục
* Thiết kế giao diện dựa các thông tin thu thập người dùng và trải nghiệm người dùng của website.
* Tạo bản mẫu giao diện thử nghiệm để phác thảo bố cục của website.
* Thu Thập thông tin phản hồi khách hàng .
* Thiết kế Website hoàn chỉnh.

### Phát triển website:

* Lựa chọn công nghệ phù hợp để phát triển website, ví dụ như HTML, CSS, JavaScript, PHP, MySQL,Bootstrap,.
* Viết code website theo các yêu cầu đã được xác định dựa trên các thông tin từ các cụ thể từ Thiết kế giao diện .
* Kiểm tra và khắc phục lỗi ,phát triển website nhiều hơn phụ với khách hàng.

### Thử nghiệm và triển khai:

* Thử nghiệm website trên các trình duyệt phổ biến và thiết bị thông minh khác nhau, đảm bảo rằng giao diện hoạt động trên mọi trình duyệt .
* Sửa lỗi và điều chỉnh giao diện để hoàn thiện website.
* Triển khai website lên hosting.

### Bảo trì và cập nhật:

* Thường xuyên kiểm tra kỹ lưỡng để xác định các lỗi và vấn đề của giao diện và chức năng,nội dung.
* Cập nhật sản phẩm mới và nội dung website thường xuyên.
* Khắc phục sửa lỗi của website khi cần thiết.
* Cập nhật websitet theo công nghệ mới phụ hợp với xu hướng.

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

* Phân tích thị trường và đối thủ cạnh tranh.
* Tối ưu hóa công cụ tìm kiếm cho website.
* Chăm sóc và Bảo Trì
* Marketing và quảng bá website.

## Ngôn ngữ HTML

### Khái niệm về HTML

- HTML (HyperText Markup Language) là ngôn ngữ cơ bản dùng để tạo ra các trang web. Nó cho phép chúng ta định dạng và tổ chức nội dung trên trang web bằng cách sử dụng các thẻ và các thuộc tính tương ứng. Nhờ vào HTML một trong những ngôn ngữ cơ bản được sử dụng để xây dựng và cấu trúc nội dung trên trang web, cho phép người dùng tạo ra cấu trúc hiển thị văn bản, hình ảnh, âm thanh, video ,liên kết và nhiều loại nội dung khác trên trình duyệt web..

### HTML cơ bản

- Một tài liệu HTML phải có phần khai báo loại tài liệu bắt đầu bằng

<!DOCTYPE html> mới xác định rằng tài liệu này là tài liệu HTML5.

- Phân tử <html> là phần tử chính của một trang .

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

- Phần tử <body>xác định nội dung của tài liệu và là nơi chứa tất cả nội dung hiển thị, chẳng hạn như tiêu đề, đoạn văn, hình ảnh, siêu liên kết, bảng, danh sách, v.v.

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

- Ví dụ:

<!DOCTYPE html>

<html>

<body>

<h1>Tiêu Đề </h1>

<p>Nội dung</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

- Một phần tử HTML được xác định bởi thẻ bắt đầu, một số nội dung và thẻ kết thúc.

- Phần tử HTML là cặp thẻ bắt đầu và kết thúc để làm một nhiệm vụ nào đó:

<tên thẻ> Nội dung ở đây... </tên thẻ>

- Ví dụ về một số thành phần HTML:

<head>Đây thẻ dùng để xác định phần đầu của tài liệu </head>

<title> Đây là thẻ xác định tiêu đề của trang Web</title>

<h1> Đây là thẻ tiêu đề </h1>

<p> Đây là thẻ hiển thị đoạn văn bản. </p>

\*Các phần tử HTML lồng nhau:

- Các phần tử HTML có thể được lồng vào nhau (điều này có nghĩa là các phần tử có thể chứa các phần tử khác).

- Tất cả các tài liệu HTML bao gồm các phần tử HTML lồng nhau.

- Ví dụ sau chứa bốn phần tử HTML ( <html>,<head>,<title>, <body>, <h1> và <p>):

<!DOCTYPE html>

<html>

<head>

<title>tiêu đề của trang Web<title>

</head>

<body>

<h1>Thẻ tiêu đề.</h1>

<p>Thẻ đoạn văn bản.</p>

</body>

</html>

Giải thích ví dụ:

- Phần tử <html> là phần tử gốc và nó xác định toàn bộ tài liệu HTML.

- Nó có thẻ bắt đầu <html>và thẻ kết thúc </html>.

- Sau đó, bên trong phần tử <html> phần tử <head> và <body>.Trong đó <head> phần đầu của tài liệu và <body> phần thân của tài liệu.

<html>

<head> </head>

-Trong phần tử <head> có phần tử <title> phần tiêu đề của trang web.

<head>

<title>tiêu đề của trang Web<title>

</head>

-Tiếp khi đóng </head> đến phần tử <body>.

</head>

<body>

</body>

-Trong <body> chứa các phẩn tử của tài liệu phần thân <h1>và <p>.

<body>

<h1>Thẻ tiêu đề.</h1>

<p>Thẻ đoạn văn bản.</p>

</body>

- Phần tử<body> xác định phần thân của tài liệu.

- Nó có thẻ bắt đầu <body>và thẻ kết thúc </body>.

Sau đó, bên trong phần tử <body> có hai phần tử khác: <h1>và <p>:

<h1>Thẻ tiêu đề</h1>

<p>Thẻ đoạn văn bản.</p>

- Phần tử <h1> xác định một tiêu đề.

- Nó có thẻ bắt đầu <h1>và thẻ kết thúc </h1>:

<h1>Thẻ tiêu đề</h1>

- Phần tử<p> xác định một đoạn văn.

- Nó có thẻ bắt đầu <p>và thẻ kết thúc </p>:

<p>Thẻ đoạn văn bản.</p>

- Không bao giờ được phép quên thẻ đóng vì nó sẽ gây ra lỗi nghiêm trọng!

- Các phần tử HTML trống là các phần tử không có nội dung và thẻ đóng.

- Ví dụ thẻ <br> xác định ngắt dòng và là phần tử trống không có thẻ đóng:

<p>Đây là thẻ đoạn văn bản <br> với thẻ ngắt dòng.</p>

-Thẻ <hr> tạo đường kẻ phân cách nằm ngang :

<h2>báo cáo 1</h2><hr>

<h2> báo cáo 2</h2>

- Thẻ HTML không phân biệt chữ hoa chữ thường: <P> và <p> là như nhau.

### Thuộc tính HTML

- Thuộc tính HTML cung cấp thông tin bổ sung về các phần tử HTML.

- Tất cả các phần tử HTML đều có thể có thuộc tính.

- Thuộc tính cung cấp thông tin bổ sung về các phần tử.

- Các thuộc tính luôn được chỉ định trong thẻ bắt đầu.

- Các thuộc tính thường có cặp tên/giá trị như: name="value".

\*Thuộc tính href:

- Thẻ <a>xác định một liên kết. Thuộc tính href chỉ định URL của trang web cần liên kết:

- Ví dụ:

<a href="https://www.w3schools.com">Tham gia W3school!</a>

\*Thuộc tính src:

- Thẻ <img>được sử dụng để nhúng hình ảnh vào trang HTML. Thuộc tính src chỉ định đường dẫn đến hình ảnh sẽ được hiển thị:

- Ví dụ:

<img src="test.jpg">

\*Thuộc tính chiều rộng và chiều cao:

- Thẻ <img> cũng có thể chứa các thuộc tính width và height giúp xác định chiều rộng và chiều cao của hình ảnh (tính bằng pixel):

- Ví dụ:

<img src="test.jpg" width="500" height="600">

\*Thuộc tính id:

-Việc áp dụng thẻ id: cho các phần tử giúp định danh và tìm kiếm chúng dễ dàng hơn khi sử dụng CSS và JavaScript.

Ví dụ

<section id="contact">

\*Thuộc tính alt:

- Thuộc tính alt là bắt buộc cho thẻ <img> chỉ định văn bản thay thế cho hình ảnh nếu vì lý do nào đó mà hình ảnh không thể hiển thị. Điều này có thể do kết nối chậm hoặc lỗi thuộc tính src hoặc do người dùng sử dụng trình đọc màn hình.

- Ví dụ:

<img src="test.jpg" alt="Đây là bức ảnh!">

\*Thuộc tính style:

Thuộc tính style được sử dụng để thêm kiểu định dạng cho một phần tử, chẳng hạn như màu sắc, phông chữ, kích thước, v.v.

- Ví dụ:

<p style="color:red;">Đây là đoạn văn bản màu đỏ.</p>

\*Thuộc tính lang:

- Cần phải luôn luôn them thuộc tính lang vào bên trong thẻ <html> để khai báo ngôn ngữ của trang Web. Điều này nhằm hỗ trợ các công cụ tìm kiếm của trình duyệt.

- Ví dụ sau chỉ định tiếng Việt là ngôn ngữ:

<!DOCTYPE html>

<html lang="vi">

<body>

...

</body>

</html>

### Tiêu đề trong HTML

- Tiêu đề trong HTML là thẻ tiêu đề mà bạn muốn hiển thị trên trang web.

- Tiêu đề trong HTML được xác định bằng thẻ <h1> đến <h6>.

- <h1> giúp xác định tiêu đề quan trọng nhất và lớn nhất còn <h6> thì xác định tiêu đề ít quan trọng nhất và nhỏ nhất.

- Ví dụ:

<h1> Tiêu đề 1 </h1>

<h2> Tiêu đề 2 </h2>

<h3> Tiêu đề 3 </h3>

<h4> Tiêu đề 4 </h4>

<h5> Tiêu đề 5 </h5>

<h6> Tiêu đề 6 </h6>

\*Tiêu đề rất quan trọng:

- Công cụ tìm kiếm sử dụng các tiêu đề để tạo cấu trúc và nội dung trang web của bạn.

- Người dùng thường lướt qua một trang theo tiêu đề của nó. Điều quan trọng là sử dụng các tiêu đề để hiển thị cấu trúc tài liệu.

- <h1>nên sử dụng các tiêu đề cho các tiêu đề chính, tiếp theo là <h2>các tiêu đề ít quan trọng hơn <h3>, v.v.

### Đoạn văn bản trong HTML

- Đoạn văn luôn bắt đầu trên một dòng mới và thường là ở dạng khối.

- Phần tử <p> giúp xác định một đoạn văn bản.

- Một đoạn văn luôn bắt đầu trên một dòng mới và trình duyệt sẽ tự động thêm một số khoảng trắng (lề) trước và sau một đoạn văn.

- Ví dụ:

<p>Sinh viên lớp DA22TTD.</p>

<p> mã số :110122100.</p>

\*Thẻ thanh ngang:

- Thẻ <hr> xác định điểm ngắt trong trang HTML và thường được hiển thị dưới dạng thanh ngang.

- Phần tử <hr> được sử dụng để phân tách nội dung (hoặc xác định thay đổi) trong trang HTML:

- Ví dụ:

<h1>Sinh viên A </h1>

<p>Học Lớp D.</p>

<hr>

<h2>Sinh viên B</h2>

<p>Học lớp D</p>

<hr>

- Lưu ý: Thẻ <hr> là thẻ trống, nghĩa là không có thẻ kết thúc.

\*Thẻ ngắt dòng:

- Phần tử <br> xác định ngắt dòng.

- Sử dụng thẻ <br> nếu bạn muốn ngắt dòng (một dòng mới) mà không bắt đầu một đoạn mới:

- Ví dụ:

<p>Đây là<br>một đoạn<br>văn bản.</p>

\*Phần tử <pre>:

- Phần tử <pre> xác định văn bản được định dạng sẵn.

- Văn bản bên trong một phần tử <pre> được hiển thị bằng phông chữ có chiều rộng cố định và nó giữ nguyên cả dấu cách và ngắt dòng:

- Ví dụ:

<pre>

Văn bản.

Sẽ hiển thị.

Giống như.

Thế n

</pre>

### Định dạng văn bản trong HTML

- HTML chứa một số phần tử để xác định văn bản có ý nghĩa đặc biệt.

- Các phần tử định dạng được thiết kế để hiển thị các loại văn bản đặc biệt:

- <b> Chữ in đậm.

- <strong> Văn bản quan trọng.

- <i> Văn bản in nghiêng.

- <em> Đoạn văn bản được nhấn mạnh.

- <mark> Văn bản được đánh dấu.

- <small> Văn bản nhỏ hơn.

- <del> Văn bản đã xóa.

- <ins> Đã chèn văn bản.

- <sub> Văn bản phía trên.

- <sup> Văn bản phía dưới.

### Nhận xét trong HTML:

- Nhận xét trong HTML không được hiển thị trong trình duyệt nhưng chúng có thể giúp ghi lại mã nguồn HTML của bạn.

- Bạn có thể thêm nhận xét vào nguồn HTML của mình bằng cách sử dụng cú pháp sau:

<!-- Ghi nhận xét tại đây! -->

- Với nhận xét, bạn có thể ghi chú lại mã và thêm vào các thông điệp giúp giải thích mã của bạn:

- Ví dụ:

<!—Đây là ghi chú -->

<p>Đây là văn bản.</p>

<!—Những điều quan trọng cần nhớ -->

\*Ẩn nội dung:

- Nhận xét có thể được sử dụng để ẩn nội dung.

- Điều này có thể hữu ích nếu bạn muốn tạm thời ẩn nội dung để sửa lỗi:

- Ví dụ:

<p>Đây là văn bản.</p>

<!-- <p>Đây là văn bản bị ẩn </p> -->

<p>Đây là văn bản.</p>

- Bạn cũng có thể ẩn nhiều dòng. Mọi thứ giữa <!-- và --> sẽ bị ẩn khỏi màn hình.

- Ví dụ:

- Ẩn một phần mã HTML:

<p>Đây là đoạn văn bản.</p>

<!--

<p>Đoạn văn bản:</p>

<img border="0" src="test.jpg" alt="Đây là ảnh">

-->

<p>Đây là văn bản.</p>

\*Ẩn nội dung nội tuyến:

- Nhận xét có thể được sử dụng để ẩn các phần ở giữa mã HTML.

- Ví dụ:

Ẩn một phần của đoạn văn:

<p>Đây <!-- là --> một đoạn văn bản.</p>

### Liên kết trong HTML

- Thẻ <a> dùng để xác định một liên kết. Nó có cú pháp sau:

<a href="url">Văn bản liên kết</a>

Thuộc tính quan trọng nhất của phần tử <a> là thuộc tính href, cho biết đích đến của liên kết.

- Văn bản liên kết là phần mà người đọc sẽ nhìn thấy.

- Nhấp vào văn bản liên kết sẽ đưa người đọc đến địa chỉ URL được chỉ định.

- Ví dụ này cho thấy cách tạo liên kết đến W3Schools.com:

<a href="https://www.w3schools.com/">Đến W3Schools.com!</a>

- Theo mặc định, các liên kết sẽ xuất hiện như sau trong tất cả các trình duyệt:

- Một liên kết chưa được truy cập được gạch chân và có màu xanh lam.

- Một liên kết đã truy cập được gạch chân và có màu tím.

- Một liên kết đang hoạt động được gạch chân và có màu đỏ.

\*Thuộc tính đích:

- Theo mặc định, trang được liên kết sẽ được hiển thị trong cửa sổ trình duyệt hiện tại. Để thay đổi điều này, bạn phải chỉ định một mục tiêu khác cho liên kết.

- Thuộc tính target chỉ định nơi mở tài liệu được liên kết.

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

\_self- Mặc định. Mở tài liệu trong cùng cửa sổ/tab khi tài liệu được nhấp vào.

\_blank- Mở tài liệu trong cửa sổ hoặc tab mới.

\_parent- Mở tài liệu trong khung cha.

\_top- Mở tài liệu ở chế độ toàn bộ cửa sổ.

- Ví dụ:

Sử dụng target="\_blank" để mở tài liệu được liên kết trong cửa sổ hoặc tab trình duyệt mới:

<a href="https://www.w3schools.com/" target="\_blank">Đến W3Schools!</a>

\*URL tuyệt đối so với URL tương đối:

- Cả hai ví dụ trên đều sử dụng URL tuyệt đối (địa chỉ web đầy đủ) trong thuộc tính href.

- Liên kết cục bộ (liên kết đến một trang trong cùng một trang web) được chỉ định bằng URL tương đối (không có phần "https://www"):

- Ví dụ:

<h2>URL tuyệt đối:</h2>

<p><a href="https://www.w3.org/">W3C</a></p>

<p><a href="https://www.google.com/">Google</a></p>

<h2>URL tương đối:</h2>

<p><a href="html\_images.asp">Ảnh HTML</a></p>

<p><a href="/css/default.asp">CSS cơ bản</a></p>

\*Sử dụng hình ảnh làm liên kết:

- Để sử dụng hình ảnh làm liên kết, chỉ cần đặt thẻ <img> bên trong thẻ <a>:

- Ví dụ:

<a href="default.asp">

<img src="test.gif" alt="HTML test" style="width:42px;height:42px;">

</a>

Liên kết HTML có thể được sử dụng để tạo dấu trang để người đọc có thể chuyển đến các phần cụ thể của trang web.

\*Tạo dấu trang trong HTML:

- Dấu trang có thể hữu ích nếu một trang web rất dài.

- Để tạo dấu trang - trước tiên hãy tạo dấu trang rồi thêm liên kết vào dấu trang đó.

- Khi nhấp vào liên kết, trang sẽ cuộn xuống hoặc cuộn lên vị trí có dấu trang.

- Ví dụ:

- Đầu tiên, sử dụng thuộc tính id để tạo dấu trang:

<h2 id="HVT">Họ và Tên</h2>

- Sau đó, thêm liên kết vào dấu trang ("Chuyển tới Chương 4"), từ trong cùng một trang:

<a href="#HVT">Chuyển tới Họ và Tên</a>

- Bạn cũng có thể thêm liên kết vào dấu trang trên một trang khác:

<a href="html\_demo.html#HVT">Chuyển tới Họ và Tên</a>

### Hình ảnh trong HTML

- Hình ảnh có thể cải thiện thiết kế và sự xuất hiện của một trang web.

\*Cú pháp thêm hình ảnh vào HTML:

- Thẻ <img> được sử dụng để nhúng hình ảnh vào trang web.

- Hình ảnh không được chèn vào trang web về mặt kỹ thuật; hình ảnh được liên kết đến các trang web. Thẻ <img> tạo một không gian lưu giữ cho hình ảnh được tham chiếu.

- Thẻ <img> là thẻ trống, chỉ chứa các thuộc tính và không có thẻ đóng.

- Thẻ <img>có hai thuộc tính bắt buộc:

* src - Chỉ định đường dẫn đến hình ảnh.
* alt - Chỉ định văn bản thay thế cho hình ảnh.

- Cú pháp:

<img src="url" alt="vanbanthaythe">

\*Thuộc tính src:

- Thuộc tính src là bắt buộc chỉ định đường dẫn (URL) tới hình ảnh.

- Lưu ý: Khi một trang web tải, tại thời điểm đó, trình duyệt sẽ lấy hình ảnh từ máy chủ web và chèn nó vào trang. Do đó, hãy đảm bảo rằng hình ảnh thực sự ở cùng một vị trí so với trang web, nếu không khách truy cập của bạn sẽ nhận được biểu tượng liên kết bị hỏng. Biểu tượng liên kết bị hỏng và alt văn bản được hiển thị nếu trình duyệt không tìm thấy hình ảnh.

- Ví dụ:

<img src="hoa.jpg" alt="Hình ảnh bông hoa">

\*Hình ảnh trong thư mục khác:

- Nếu bạn có hình ảnh của mình trong thư mục con, bạn phải đưa tên thư mục vào thuộc tính src:

- Ví dụ:

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

\*Hình ảnh trên Máy chủ/Trang web khác:

- Một số trang web trỏ đến một hình ảnh trên một máy chủ khác.

- Để trỏ đến hình ảnh trên máy chủ khác, bạn phải chỉ định URL tuyệt đối (đầy đủ) trong thuộc tính src:

- Ví dụ:

<img src="https://www.w3schools.com/images/w3schools\_green.jpg" alt="W3Schools.com">

### Bảng trong HTML:

- Bảng HTML cho phép các nhà phát triển web sắp xếp dữ liệu thành hàng và cột.

\*Xác định bảng HTML:

- Một bảng trong HTML bao gồm các ô bảng bên trong các hàng và cột.

- Một có các phần tử như <table> ,<tr>,<th>,<td>.

-Và có thêm 2 thuộc tính thẻ colspan và rowspan.

- Ví dụ một bảng HTML đơn giản:

<table>

<tr>

<th>Kiều Thị Thu Hằng</th>

<th>Trần Anh Đức</th>

<th>Vương Thị Lê Na</th>

</tr>

<tr>

<td>20/10/1995</td>

<td>03/08/1993</td>

<td>08/03/1999</td>

</tr>

<tr>

<td>Nữ</td>

<td>Nam</td>

<td>Nữ</td>

</tr>

</table>

\*Ô bảng:

Mỗi ô của bảng được xác định bởi thẻ mở <td>và một thẻ đóng</td>.

Mọi thứ ở giữa thẻ <td>và thẻ </td>là nội dung của ô bảng.

- Ví dụ:

<table>

<tr>

<td>Emil</td>

<td>gmail</td>

<td>outlock</td>

</tr>

</table>

\*Hàng bảng:

- Mỗi hàng của bảng bắt đầu bằng thẻ mở <tr>và kết thúc bằng một thẻ đóng </tr>.

- Ví dụ:

<table>

<tr>

<td>Trang</td>

<td>Quyên</td>

<td>Nhi</td>

</tr>

<tr>

<td>10</td>

<td>11</td>

<td>12</td>

</tr>

</table>

- Bạn có thể có bao nhiêu hàng tùy thích trong một bảng; chỉ cần đảm bảo rằng số lượng ô trong mỗi hàng giống nhau.

\*Tiêu đề bảng:

- Đôi khi bạn muốn các ô của mình là ô tiêu đề bảng. Trong những trường hợp đó, hãy sử dụng thẻ <th> thay vì thẻ <td>:

- Ví dụ đặt hàng đầu tiên là ô tiêu đề bảng:

|  |  |  |
| --- | --- | --- |
| <table>  <tr>  <th>Person 1</th>  <th>Person 2</th>  <th>Person 3</th>  </tr> | <tr>  <td>Emil</td>  <td>Tobias</td>  <td>Linus</td>  </tr> | <tr>  <td>16</td>  <td>14</td>  <td>10</td>  </tr>  </table> |

- Theo mặc định, các văn bản trong <th> được in đậm và căn giữa nhưng bạn có thể thay đổi điều đó bằng CSS.

\*Bảng chú thích:

- Bạn có thể thêm chú thích làm tiêu đề cho toàn bộ bảng.

- Để thêm chú thích vào bảng, hãy sử dụng thẻ <caption>:

<table style="width:100%">

<caption>VIETNAM</caption>

<tr>

<th>VN30</th>

<th>HNX30</th>

</tr>

<tr>

<td>VCB</td>

<td>5$</td>

</tr>

<tr>

<td>STB</td>

<td>3$</td>

</tr>

</table>

\*Colspan:

- Để tạo một ô trải dài trên nhiều cột, hãy sử dụng thuộc tính colspan:

- Ví dụ:

|  |  |  |
| --- | --- | --- |
| <table>  <tr>  <th colspan="2">Name</th>  <th>Age</th>  </tr>  <tr> | <td>Jill</td>  <td>Smith</td>  <td>43</td>  </tr> | <tr>  <td>Eve</td>  <td>Jackson</td>  <td>57</td>  </tr>  </table> |

\*Rowspan:

- Để tạo một ô trải dài trên nhiều hàng, hãy sử dụng thuộc tính rowspan:

- Ví dụ:

<table>

<tr>

<th>Name</th>

<td>Jill</td>

</tr>

<tr>

<th rowspan="2">Phone</th>

<td>555-1234</td>

</tr>

<tr>

<td>555-8745</td>

</tr>

</table>

\*Nhóm bảng trong HTML:

- Nếu bạn muốn tạo kiểu cho hai cột đầu tiên của bảng, hãy sử dụng các phần tử <colgroup> và <col> .

- Phần tử <colgroup> này nên được sử dụng làm nơi chứa các thông số kỹ thuật của cột.

- Mỗi nhóm được chỉ định bằng một phần tử <col>.

- Thuộc tính span chỉ định số lượng cột có kiểu.

- Thuộc tính style chỉ định kiểu cho các cột.

- Ví dụ:

|  |  |  |
| --- | --- | --- |
| <table>  <colgroup>  <colspan="2" style="background-color: #D6EEEE"> | </colgroup>  <tr>  <th>MON</th> | <th>TUE</th>  <th>WED</th>  <th>THU</th> |

### Danh sách trong HTML

- Danh sách HTML cho phép các nhà phát triển web nhóm một tập hợp các mục liên quan vào danh sách.

\*Danh sách HTML không có thứ tự:

- Danh sách không có thứ tự bắt đầu bằng thẻ <ul>. Mỗi mục danh sách bắt đầu bằng thẻ <li>.

- Các mục trong danh sách sẽ được đánh dấu bằng dấu đầu dòng (vòng tròn nhỏ màu đen) theo mặc định.

- Ví dụ:

<ul>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ul>

\*Danh sách HTML có thứ tự:

- Một danh sách có thứ tự bắt đầu bằng thẻ <ol>. Mỗi mục danh sách bắt đầu bằng thẻ <li>.

- Các mục trong danh sách sẽ được đánh dấu bằng số theo mặc định.

- Ví dụ:

<ol>

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

\*Danh sách mô tả HTML:

- HTML cũng hỗ trợ danh sách mô tả.

- Danh sách mô tả là danh sách các thuật ngữ, kèm theo mô tả cho từng thuật ngữ.

- Thẻ <dl> xác định danh sách mô tả, thẻ <dt> xác định thuật ngữ (tên) và thẻ <dd> mô tả từng thuật ngữ:

- Ví dụ:

<dl>

<dt>Coffee</dt>

<dd>- black hot drink</dd>

<dt>Milk</dt>

<dd>- white cold drink</dd>

</dl>

\*Danh sách HTML lồng nhau:

- Danh sách có thể được lồng vào nhau (danh sách bên trong danh sách)

- Ví dụ:

<ul>

<li>Coffee</li>

<li>Tea

<ul>

<li>Black tea</li>

<li>Green tea</li>

</ul>

</li>

<li>Milk</li>

</ul>

\*Danh sách HTML có thứ tự - Thuộc tính loại:

- Thuộc tính type của thẻ <ol>, xác định loại điểm đánh dấu mục danh sách:

Loại Mô tả

type="1" Các mục trong danh sách sẽ được đánh số bằng số (mặc định).

type="A" Các mục trong danh sách sẽ được đánh số bằng chữ cái in hoa.

type="a" Các mục trong danh sách sẽ được đánh số bằng chữ cái thường.

type="I" Các mục trong danh sách sẽ được đánh số bằng số La Mã in hoa.

type="i" Các mục trong danh sách sẽ được đánh số bằng số La Mã thường.

\*Thay đổi thứ tự danh sách:

- Theo mặc định, danh sách có thứ tự sẽ bắt đầu đếm từ 1. Nếu muốn bắt đầu đếm từ một số được chỉ định, bạn có thể sử dụng thuộc tính start:

- Ví dụ:

<ol start="50">

<li>Coffee</li>

<li>Tea</li>

<li>Milk</li>

</ol>

### Phần tử khối và phần tử nội tuyến:

- Mọi phần tử HTML đều có giá trị hiển thị mặc định, tùy thuộc vào loại phần tử đó.

- Hai giá trị hiển thị phổ biến nhất là khối và nội tuyến.

\*Các phần tử cấp khối:

- Phần tử cấp khối luôn bắt đầu trên một dòng mới và trình duyệt sẽ tự động thêm một số khoảng trắng (lề) trước và sau phần tử.

- Phần tử cấp khối luôn chiếm toàn bộ chiều rộng có sẵn (kéo dài sang trái và phải hết mức có thể).

- Hai phần tử khối thường được sử dụng là: <p> và <div>.

- Phần tử <p> xác định một đoạn văn bản trong tài liệu HTML.

- Phần tử <div> xác định một bộ phận hoặc một phần trong tài liệu HTML.

- Ví dụ:

<p>Hello World</p>

<div>Hello World</div>

Dưới đây là các phần tử cấp khối trong HTML:

<address>,<article>,<aside>,<blockquote>,<canvas>,<dd>,<div>,<dl>,<dt>,

<fieldset>,<figcaption>,<figure>,<footer>,<form>,<h1>,<h6>,<header>,<hr>,

<li><,main>,<nav>,<noscript>,<ol>,<p>,<pre>,<section>,<table>,<tfoot>,<ul> , <video>.

\*Các phần tử nội tuyến:

- Phần tử nội tuyến không bắt đầu trên một dòng mới.

- Phần tử nội tuyến chỉ chiếm nhiều chiều rộng nếu cần thiết.

- Ví dụ:

<span>Hello World</span>

Dưới đây là các phần tử nội tuyến trong HTML:

<a>,<abbr>,<acronym>,<b>,<bdo>,<big>,<br>,<button>,<cite>,<code>,<dfn>,

<em>,<i>,<img>,<input>,<kbd>,<label>,<map>,<object>,<output>,<q>,<samp>,

<script>,<select>,<small>,<span>,<strong>,<sub>,<sup>,<textarea>,<time>,<tt>,

<var>

\*Phần tử <div>:

- Phần tử <div> này thường được sử dụng làm vùng chứa cho các phần tử HTML khác.

- Phần tử <div> không có thuộc tính bắt buộc nhưng style, class và id là phổ biến.

- Khi được sử dụng cùng với CSS, phần tử <div> có thể được sử dụng để tạo kiểu cho các khối nội dung.

- Ví dụ:

<div style="background-color:black;color:white;padding:20px;">

<h2>London</h2>

<p>London is the capital city of England. It is the most populous city in the United Kingdom, with a metropolitan area of over 13 million inhabitants.</p>

</div>

\*Phần tử <span>:

- Phần tử <span> là một vùng chứa nội tuyến được sử dụng để đánh dấu một phần của văn bản hoặc một phần của tài liệu.

- Phần tử <span> không có thuộc tính bắt buộc nhưng style, class và id là phổ biến.

- Khi được sử dụng cùng với CSS, phần tử <span> có thể được sử dụng để tạo kiểu cho các phần của văn bản

- Ví dụ:

<p>My mother has <span style="color:blue;font-weight:bold;">blue</span> eyes and my father has <span style="color:darkolivegreen;font-weight:bold;">dark green</span> eyes.</p>

### Khung nội tuyến trong HTML

- Khung nội tuyến HTML được sử dụng để hiển thị một trang web trong một trang web.

- Thẻ <iframe> chỉ định khung nội tuyến.

- Khung nội tuyến được sử dụng để nhúng tài liệu khác trong tài liệu HTML hiện tại.

- Cú pháp:

<iframe src="url" title="description"></iframe>

\*Đặt chiều cao và chiều rộng:

- Sử dụng các thuộc tính height và width để chỉ định kích thước của iframe.

- Chiều cao và chiều rộng được chỉ định bằng pixel theo mặc định

- Ví dụ:

<iframe src="demo\_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>

\*Mục tiêu cho một liên kết:

- Iframe có thể được sử dụng làm khung đích cho một liên kết.

- Thuộc tính target của liên kết phải tham chiếu đến thuộc tính name của iframe

- Ví dụ:

<iframe src="demo\_iframe.htm" name="iframe\_a" title="Iframe Example"></iframe>

<p><a href="https://www.w3schools.com" target="iframe\_a">W3Schools.com</a></p>

### Đường dẫn tệp trong HTML

- Đường dẫn tệp mô tả vị trí của tệp trong cấu trúc thư mục của trang web.

- Ví dụ về đường dẫn tệp:

Đường dẫn Miêu tả

<img src="picture.jpg"> Tệp "picture.jpg" nằm trong cùng thư mục với trang hiện tại

<img src="images/picture.jpg"> Tệp "picture.jpg" nằm trong thư mục images trong thư mục hiện tại

<img src="/images/picture.jpg"> Tệp "picture.jpg" nằm trong thư mục images ở thư mục gốc của trang web hiện tại

<img src="../picture.jpg"> Tệp "picture.jpg" nằm trong thư mục cao hơn một cấp so với thư mục hiện tại

\*Đường dẫn tệp tuyệt đối:

- Đường dẫn tệp tuyệt đối là URL đầy đủ của tệp:

Ví dụ

<img src="https://www.w3schools.com/images/picture.jpg" alt="Mountain">

\*Đường dẫn tệp tương đối:

- Đường dẫn tệp tương đối trỏ đến tệp tương ứng với trang hiện tại.

- Trong ví dụ sau, đường dẫn tệp trỏ đến một tệp trong thư mục images nằm ở thư mục gốc của trang web hiện tại:

- Ví dụ:

<img src="/images/picture.jpg" alt="Mountain">

### Biểu mẫu trong HTML

- Một biểu mẫu HTML được sử dụng để thu thập thông tin đầu vào của người dùng.

- Đầu vào của người dùng thường được gửi đến máy chủ để xử lý.

\*Phần tử <form>:

- Phần tử <form> được sử dụng để tạo biểu mẫu HTML cho người dùng nhập:

<form>

form elements

</form>

- Phần tử <form> là nơi chứa các loại phần tử đầu vào khác nhau, chẳng hạn như: trường văn bản, hộp kiểm, nút radio, nút gửi, v.v.

\*Phần tử <input>:

- Phần tử HTML <input> là phần tử biểu mẫu được sử dụng nhiều nhất.

- Một phần tử <input> có thể được hiển thị theo nhiều cách, tùy thuộc vào thuộc tính type.

\*Trường văn bản:

- <input type="text"> giúp xác định trường nhập một dòng để nhập văn bản.

- Ví dụ:

- Một biểu mẫu có các trường nhập văn bản:

<form>

<label for="Ten">Ten:</label><br>

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

<label for="HoTen">Họ và Tên:</label><br>

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

</form>

\*Phần tử <label>:

- Lưu ý việc sử dụng phần tử <label> trong ví dụ trên.

- Thẻ <label>xác định nhãn cho nhiều thành phần biểu mẫu.

- Phần tử <label> này hữu ích cho người dùng trình đọc màn hình vì trình đọc màn hình sẽ đọc to nhãn khi người dùng tập trung vào phần tử đầu vào.

- Phần tử <label> này cũng giúp người dùng gặp khó khăn khi nhấp vào các vùng rất nhỏ (chẳng hạn như nút radio hoặc hộp kiểm) - bởi vì khi người dùng nhấp vào văn bản trong phần <label>tử, nó sẽ chuyển đổi nút radio/hộp kiểm.

- Thuộc tính for của thẻ <label> phải giống thuộc tính id của phần tử <input> để liên kết chúng lại với nhau.

\*Nút chọn:

- <input type="radio"> xác định một nút radio.

- Nút radio cho phép người dùng chọn MỘT trong số các lựa chọn có giới hạn.

- Ví dụ:

Một biểu mẫu có các nút radio:

<p>Choose your favorite Web language:</p>

<form>

<input type="radio" id="html" name="fav\_language" value="HTML">

<label for="html">HTML</label><br>

<input type="radio" id="css" name="fav\_language" value="CSS">

<label for="css">CSS</label><br>

<input type="radio" id="javascript" name="fav\_language" value="JavaScript">

<label for="javascript">JavaScript</label>

</form>

\*Hộp kiểm:

- <input type="checkbox"> xác định một hộp kiểm .

- Các hộp kiểm cho phép người dùng chọn các tùy chọn KHÔNG hoặc NHIỀU tùy chọn trong một số lựa chọn giới hạn.

- Ví dụ:

Một biểu mẫu có các hộp kiểm:

<form>

<input type="checkbox" id="vehicle1" name="vehicle1" value="Bike">

<label for="vehicle1"> I have a bike</label><br>

<input type="checkbox" id="vehicle2" name="vehicle2" value="Car">

<label for="vehicle2"> I have a car</label><br>

<input type="checkbox" id="vehicle3" name="vehicle3" value="Boat">

<label for="vehicle3"> I have a boat</label>

</form>

\*Nút Gửi:

- <input type="submit"> xác định nút để gửi dữ liệu biểu mẫu tới trình xử lý biểu mẫu.

- Trình xử lý biểu mẫu thường là một tệp trên máy chủ có tập lệnh để xử lý dữ liệu đầu vào.

- Trình xử lý biểu mẫu được chỉ định trong thuộc tính action của biểu mẫu.

- Ví dụ:

Một biểu mẫu có nút gửi:

<form action="/action\_page.php">

<label for="fname">First name:</label><br>

<input type="text" id="fname" name="fname" value="John"><br>

<label for="lname">Last name:</label><br>

<input type="text" id="lname" name="lname" value="Doe"><br><br>

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

</form>

\*Thuộc tính tên cho <input>:

- Lưu ý rằng mỗi trường đầu vào phải có một thuộc tính name để được gửi.

- Nếu thuộc tính name bị bỏ qua, giá trị của trường đầu vào sẽ không được gửi đi.

- Ví dụ:

Ví dụ này sẽ không gửi giá trị của trường nhập "Tên":

<form action="/action\_page.php">

<label for="fname">First name:</label><br>

<input type="text" id="fname" value="John"><br><br>

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

</form>

\*Thuộc tính action:

- Thuộc tính action xác định hành động sẽ được thực hiện khi biểu mẫu được gửi.

- Thông thường, dữ liệu biểu mẫu được gửi đến một tệp trên máy chủ khi người dùng nhấp vào nút gửi.

- Trong ví dụ bên dưới, dữ liệu biểu mẫu được gửi đến tệp có tên "action\_page.php".

- Tệp này chứa tập lệnh phía máy chủ xử lý dữ liệu biểu mẫu:

- Ví dụ

Khi gửi, hãy gửi dữ liệu biểu mẫu tới "action\_page.php":

<form action="/action\_page.php">

<label for="fname">First name:</label><br>

<input type="text" id="fname" name="fname" value="John"><br>

<label for="lname">Last name:</label><br>

<input type="text" id="lname" name="lname" value="Doe"><br><br>

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

</form>

\*Thuộc tính method:

- Thuộc tính method chỉ định phương thức HTTP sẽ được sử dụng khi gửi dữ liệu biểu mẫu.

- Dữ liệu biểu mẫu có thể được gửi dưới dạng biến URL (với method="get") hoặc dưới dạng giao dịch bài đăng HTTP (với method="post").

- Phương thức HTTP mặc định khi gửi dữ liệu biểu mẫu là GET.

- Ví dụ:

Ví dụ này sử dụng phương thức GET khi gửi dữ liệu biểu mẫu:

<form action="/action\_page.php" method="get">

\*Các phần tử biểu mẫu HTML:

- Phần tử HTML <form>có thể chứa một hoặc nhiều phần tử biểu mẫu sau:

<input>;<label>;<select>;<textarea>;<button>;<fieldset>;<legend>;<datalist>;

<output>;<option>;<optgroup>

\*Phần tử <select>:

- Phần tử <select> giúp xác định danh sách thả xuống:

- Ví dụ:

<label for="cars">Choose a car:</label>

<select id="cars" name="cars">

<option value="volvo">Volvo</option>

<option value="saab">Saab</option>

<option value="fiat">Fiat</option>

<option value="audi">Audi</option> </select>

- Phần tử <option> xác định một tùy chọn có thể được chọn.

- Theo mặc định, mục đầu tiên trong danh sách thả xuống được chọn.

- Để xác định tùy chọn được chọn trước, hãy thêm thuộc tính selected vào <option>:

Ví dụ

<option value="fiat" selected>Fiat</option>

\*Phần tử <textarea>:

- Phần tử <textarea> xác định trường nhập nhiều dòng (vùng văn bản):

- Ví dụ:

<textarea name="message" rows="10" cols="30">

The cat was playing in the garden.

</textarea>

- Thuộc tính rows chỉ định số dòng hiển thị trong vùng văn bản.

- Thuộc tính cols chỉ định chiều rộng hiển thị của vùng văn bản.

\*Các phần tử <fieldset> và <legend>:

- Phần tử <fieldset> được sử dụng để nhóm dữ liệu liên quan trong một biểu mẫu.

- Phần tử <legend> xác định chú thích cho phần tử <fieldset>.

Ví dụ

<form action="/action\_page.php">

<fieldset>

<legend>Personalia:</legend>

<label for="fname">First name:</label><br>

<input type="text" id="fname" name="fname" value="John"><br>

<label for="lname">Last name:</label><br>

<input type="text" id="lname" name="lname" value="Doe"><br><br>

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

</fieldset>

</form>

## Bảng định kiểu CSS

### CSS nội tuyến (Inline CSS)

- CSS nội tuyến được viết trực tiếp trong thẻ HTML, sử dụng thuộc tính style. Loại CSS này được sử dụng để định dạng các phần tử HTML cụ thể. Ví dụ:

<p style="color: red; font-size: 20px;">Đây là một đoạn văn bản được định dạng bằng CSS nội tuyến.</p>

### CSS nội bộ (Internal CSS)

- CSS nội bộ được viết trong thẻ <style> nằm trong phần <head> của tài liệu HTML. Loại CSS này được sử dụng để định dạng nhiều phần tử HTML trong cùng một trang web. Ví dụ:

<head>

<style>

p {

color: red;

font-size: 10px;

}

</style>

</head>

### CSS bên ngoài (External CSS)

- CSS bên ngoài được viết trong một tệp riêng có phần mở rộng .css. Tệp này được liên kết với tài liệu HTML bằng thẻ <link> trong phần <head> của tài liệu HTML. Loại CSS này được sử dụng để định dạng nhiều trang web trong cùng một dự án. Ví dụ:

<head>

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

</head>

## Ngôn ngữ Javascript

### Giới thiệu về Javascript

- JavaScript là ngôn ngữ lập trình bổ sung tính tương tác và hành vi tùy chỉnh cho các trang web của chúng tôi. Nó là ngôn ngữ kịch bản phía máy khách, có nghĩa là nó chạy trên máy của người dùng chứ không phải trên máy chủ, như các ngôn ngữ lập trình web khác như PHP và Ruby.

- Điều đó có nghĩa là JavaScript, JavaScript cũng được biết đến như một ngôn ngữ lập trình động và được gõ lỏng lẻo .JavaScript là một ngôn ngữ kịch bản nhẹ nhưng cực kỳ mạnh mẽ. Chúng ta thường gặp nó nhất thông qua trình duyệt của mình, nhưng JavaScript đã xâm nhập vào mọi thứ, từ ứng dụng gốc, tệp PDF đến sách điện tử.

- Thông thường chúng ta sẽ gặp JavaScript như một cách để thêm tính tương tác vào một trang.

- Tất cả các thành phần, thuộc tính và văn bản trên một trang web đều có thể được truy cập bằng các tập lệnh sử dụng DOM.

- JavaScript cho phép bạn tạo các giao diện có độ phản hồi cao để cải thiện trải nghiệm người dùng và cung cấp chức năng động mà không cần đợi máy chủ tải trang mới.

### Thêm Javascript vào trang

- Giống như CSS, bạn có thể nhúng tập lệnh ngay trong tài liệu hoặc giữ nó ở tệp bên ngoài và liên kết nó với trang. Cả hai phương pháp đều sử dụng phần tử script .

\*Tập lệnh nhúng

-Để nhúng tập lệnh vào một trang, chỉ cần thêm mã làm nội dung của tập lệnh

**\*Tập lệnh ngoài**

***Nhược điểm:***

- Phương thức khác sử dụng thuộc tính src để trỏ đến tệp tập lệnh (có phần mở rộng .js hậu tố) theo URL của nó. Trong trường hợp này, phần tử script không có nội dung:

<script src="my\_script.js"></script>

***Ưu điểm:***

- Tập lệnh bên ngoài là bạn có thể áp dụng cùng một tập lệnh cho nhiều trang (cùng lợi ích mà biểu định kiểu bên ngoài mang lại). Tất nhiên, nhược điểm là mỗi tập lệnh bên ngoài yêu cầu một yêu cầu HTTP bổ sung của máy chủ, điều này làm chậm hiệu suất.

**\* Vị trí tập lệnh:**

- Phần tử tập lệnh có thể xuất hiện ở bất kỳ đâu trong tài liệu, nhưng vị trí phổ biến nhất dành cho tập lệnh là ở phần đầu của tài liệu và ở cuối phần nội dung. Bạn không nên rải chúng khắp tài liệu vì chúng sẽ khó tìm và bảo trì.

- Đối với hầu hết các tập lệnh, phần cuối của tài liệu, ngay trước thẻ </body> , là vị trí ưu tiên vì trình duyệt sẽ hoàn tất việc phân tích tài liệu và cấu trúc DOM của nó:

### Javascript cơ bản

- Có một số quy tắc cú pháp phổ biến áp dụng cho tất cả JavaScript. Điều quan trọng cần biết là JavaScript phân biệt chữ hoa chữ thường. Một biến có tên myVariable, một biến có tên myvariable và một biến có tên MYVariable sẽ được coi là ba đối tượng khác nhau

**Statements(Các câu lệnh)**

- Một câu lệnh là một lệnh cho trình duyệt biết phải làm gì. Đây là một câu lệnh đơn giản giúp trình duyệt hiển thị cảnh báo với cụm từ “Thank you”:

alert(“Thanks you.”);

- Dấu chấm phẩy ở cuối câu lệnh cho JavaScript biết rằng đó là phần cuối của lệnh, giống như dấu chấm kết thúc một câu.

**Comments(Bình luận)**

- Có hai phương pháp sử dụng chú thích. Đối với các chú thích một dòng, hãy sử dụng hai ký tự gạch chéo **(//)** ở đầu dòng

**//** Đây là chú thích một dòng.

- Mọi thứ trong ký tự **/\* \*/** đều bị trình duyệt bỏ qua. Bạn có thể sử dụng cú pháp này để **“nhận xét”** các ghi chú và thậm chí cả các đoạn tập lệnh khi khắc phục sự cố.

**/\***

Đây là chú thích nhiều dòng.

\*/

### Variables(Biến)

- Một biến giống như một thùng chứa thông tin. Bạn đặt tên cho nó rồi gán cho nó một giá trị, giá trị này có thể là số, chuỗi văn bản, một phần tử trong DOM hoặc một hàm—thực ra là bất kỳ thứ gì. Điều này cho chúng ta một cách thuận tiện để tham chiếu giá trị đó sau này theo tên. Bản thân giá trị có thể được sửa đổi và gán lại theo bất kỳ cách nào mà logic của tập lệnh của chúng tôi yêu cầu.

- Khai báo sau đây tạo một biến có tên foo và gán cho nó giá trị 5:

var foo = 5;

- Một số quy tắc đặt tên biến:

• Nó phải bắt đầu bằng một chữ cái hoặc dấu gạch dưới.

• Nó có thể chứa các chữ cái, chữ số và dấu gạch dưới dưới bất kỳ hình thức kết hợp nào.

• Nó có thể không chứa khoảng trắng ký tự. Ngoài ra, hãy sử dụng dấu gạch dưới thay cho dấu cách hoặc đóng khoảng trắng và sử dụng kiểu chữ lạc đà thay thế (ví dụ: my\_variable hoặc myVariable).

• Nó không được chứa các ký tự đặc biệt (ví dụ: ! /\ + \* =).

### Data types(Loại dữ liệu)

**Undefined(Không xác định)**

**-** Loại dữ liệu đơn giản nhất trong số này có thể không được xác định. Nếu chúng ta khai báo một biến bằng cách đặt tên nhưng không có giá trị thì biến đó chứa giá trị không xác định.

Var foo;

Alert(foo);

**Null(Vô giá trị)**

- Tương tự như không xác định, việc gán một biến null (một lần nữa, phân biệt chữ hoa chữ thường)

Var foo = null;

Alert(foo);

**Numbers(số)**

- Biến này hoạt động giống như chính số đó, vì vậy bạn có thể thực hiện các thao tác với nó giống như bạn sẽ làm với bất kỳ số nào khác bằng cách sử dụng ký hiệu toán học cổ điển: +, -, \* và / tương ứng cho cộng, trừ, nhân và chia. Trong ví dụ này, chúng ta sử dụng dấu cộng (+) để thêm foo vào chính nó (foo + foo).

Var foo = 5;

Alert(foo+foo); //10

**Strings(dây)**

- Một loại dữ liệu khác có thể được lưu vào biến là chuỗi, về cơ bản là một dòng văn bản. Việc đặt các ký tự trong một tập hợp dấu ngoặc đơn hoặc dấu ngoặc kép cho biết đó là một chuỗi.

Var foo = “number”;

Alert(foo); //number

Trước đó chúng ta đã thấy dấu cộng (+) dùng để cộng số. Khi dấu cộng được sử dụng với các chuỗi, nó sẽ nối các chuỗi lại với nhau (gọi là nối) thành một chuỗi dài:

Var foo = “1”;

Alert(foo+foo); //11

**Boolean**

- Chúng ta cũng có thể gán cho một biến một giá trị đúng hoặc sai. Đây được gọi là giá trị Boolean và nó là cơ sở cho tất cả các loại logic nâng cao. Các giá trị Boolean sử dụng các từ khóa đúng và sai được tích hợp trong JavaScript, vì vậy dấu ngoặc kép là không cần thiết.

Var foo = true;

**Arrays(Mảng)**

- Mảng là một nhóm gồm nhiều giá trị (được gọi là thành viên) có thể được gán cho một biến duy nhất. Các giá trị trong một mảng được cho là được lập chỉ mục, nghĩa là bạn có thể tham chiếu chúng theo số theo thứ tự chúng xuất hiện trong danh sách.

- Bây giờ, bất cứ khi nào bạn cần truy cập bất kỳ giá trị nào trong số đó, bạn có thể lấy chúng từ một mảng foo bằng cách tham chiếu số chỉ mục của chúng:

Alert(foo[0]);

Alert(foo[1]);

Alert(foo[2]);

### Comparison Operators(Toán tử so sánh)

- Có một tập hợp các ký tự đặc biệt được gọi là toán tử so sánh để đánh giá và so sánh các giá trị theo nhiều cách khác nhau:

Toán tử Mô tả

== bằng

=== bằng giá trị và loại dữ liệu

!= không bằng

!== không bằng giá trị hoặc loại dữ liệu

> lớn hơn

< nhỏ hơn

>= lớn hơn hoặc bằng

<= nhỏ hơn hoặc bằng

? toán tử ba ngôi

- Khi chúng ta so sánh hai giá trị, **JavaScript** sẽ đánh giá câu lệnh và trả về cho chúng ta giá trị Boolean tùy thuộc vào câu lệnh đó là đúng hay sai.

Alert(5==5); //true

Alert(5!=5); //false

Alert(5<5);//false

### Mathematical operators (Toán tử toán học)

- Loại toán tử khác là toán tử toán học, thực hiện các hàm toán học trên các giá trị số (và tất nhiên, các biến chứa giá trị số). Chúng ta đã đề cập sơ qua về các toán tử đơn giản để cộng (+), trừ (-), nhân (\*) và chia (/). Ngoài ra còn có một số phím tắt hữu ích bạn nên biết:

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

- Các câu lệnh if/else là cách chúng ta khiến JavaScript tự hỏi một câu hỏi đúng/sai. Chúng ít nhiều là nền tảng cho tất cả logic nâng cao có thể được viết bằng JavaScript và chúng cũng đơn giản như việc lập trình. Trên thực tế, chúng gần như được viết bằng tiếng Anh đơn giản. Cấu trúc của câu lệnh điều kiện như sau:

If(true){

// lệnh

}

- Chúng ta cũng có thể kiểm tra rõ ràng xem có điều gì đó sai hay không bằng cách sử dụng toán tử so sánh != , có nghĩa là “không bằng”.

Var two = 2;

If(1 == 2){

two = 1;

}

### Loops(vòng lặp)

- Vòng lặp cho phép chúng ta dễ dàng xử lý các tập dữ liệu khổng lồ.

- Có một số cách để viết vòng lặp, nhưng phương thức for là một trong những cách phổ biến nhất. Cấu trúc cơ bản của vòng lặp for như sau:

- Các nhà phát triển sử dụng vòng lặp để thực hiện một số tác vụ, chẳng hạn như sau:

• Lặp qua danh sách các phần tử trên trang và kiểm tra giá trị của từng phần tử, áp dụng một kiểu cho từng phần tử hoặc thêm/xóa/thay đổi thuộc tính trên mỗi phần tử.

• Tạo một mảng các phần tử mới trong mảng ban đầu có giá trị nhất định.

### Functions(chức năng)

- Các hàm cho phép sử dụng lại mã bất cứ khi nào nó được tham chiếu, do đó bạn không cần phải viết đi viết lại.

- Có hai loại hàm: những loại hàm “có sẵn” (hàm JavaScript gốc) và những loại do bạn tự tạo ra (hàm tùy chỉnh). Chúng ta hãy nhìn vào từng cái.

**Hàm gốc**

- Hàng trăm hàm được xác định trước được tích hợp vào JavaScript, bao gồm:

alert(), confirm(), and prompt()

- Các chức năng này kích hoạt hộp thoại cấp trình duyệt

Date()

- Trả về ngày và giờ hiện tại.

parseInt("123")

- Hàm này, trong số những thứ khác, sẽ lấy kiểu dữ liệu chuỗi chứa số và biến nó thành kiểu dữ liệu số. Chuỗi được truyền cho hàm dưới dạng đối số.

setTimeout(functionName, 5000)

- Thực thi một chức năng sau một khoảng thời gian trì hoãn. Hàm được chỉ định trong đối số đầu tiên và độ trễ được chỉ định bằng mili giây trong đối số thứ hai (trong ví dụ: 5.000 mili giây, tương đương 5 giây).

**Chức năng tùy chỉnh**

- Để tạo một hàm tùy chỉnh, chúng ta nhập từ khóa hàm , theo sau là tên của hàm, tiếp theo là mở và đóng dấu ngoặc đơn, sau đó là mở và đóng dấu ngoặc nhọn:

**Arguments(Tranh Luận)**

- Việc có một hàm thực thi cùng một mã trong toàn bộ tập lệnh của bạn dường như không hữu ích lắm. Chúng ta có thể “truyền đối số” (cung cấp dữ liệu) cho các hàm gốc và hàm tùy chỉnh để áp dụng logic của hàm cho các tập hợp dữ liệu khác nhau tại các thời điểm khác nhau. Để giữ một vị trí cho các đối số, hãy tạo một tên có thể thay đổi (hoặc một chuỗi tên được phân tách bằng dấu phẩy) trong dấu ngoặc đơn sau tên của hàm tại thời điểm hàm được xác định.

**Trả về một giá trị**

- Việc sử dụng một hàm để tính toán một cái gì đó rồi trả lại cho bạn một giá trị mà bạn có thể sử dụng ở nơi khác trong tập lệnh của mình là điều khá phổ biến. Chúng ta có thể thực hiện điều này bằng cách sử dụng những gì chúng ta biết hiện nay, thông qua việc áp dụng thông minh các biến số, nhưng có một cách dễ dàng hơn nhiều.

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

- Khái niệm về tính sẵn có của biến này được gọi là phạm vi của nó. Một biến có thể được bất kỳ tập lệnh nào trên trang của bạn sử dụng đều có phạm vi toàn cầu và một biến chỉ có sẵn trong hàm gốc của nó là có phạm vi cục bộ.

- Các biến JavaScript sử dụng các hàm để quản lý phạm vi của chúng. Nếu một biến được xác định bên ngoài một hàm, biến đó sẽ có phạm vi toàn cầu và có sẵn cho tất cả các tập lệnh. Khi bạn xác định một biến trong một hàm và bạn muốn nó chỉ được sử dụng bởi hàm đó, bạn có thể gắn cờ biến đó là nằm trong phạm vi cục bộ bằng cách đặt trước tên biến bằng từ khóa var :

- Hãy nhớ rằng không phải lúc nào chúng ta cũng có thể kiểm soát tất cả mã đang hoạt động trên trang của mình. Việc các trang bao gồm mã do bên thứ ba viết là rất phổ biến, ví dụ:

• ***Tập lệnh hiển thị quảng cáo***

***• Tập lệnh phân tích và theo dõi người dùng***

***• Nút “chia sẻ” trên mạng xã hội***

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

-Ngoài khả năng kiểm soát các thành phần trên trang web, JavaScript còn cho phép bạn truy cập và khả năng thao tác các phần của trình duyệt.

-Trong JavaScript, trình duyệt được gọi là đối tượng cửa sổ . Đối tượng window có một số thuộc tính và phương thức mà chúng ta có thể sử dụng để tương tác với nó

+Bảng dưới đây liệt kê một vài thuộc tính và phương thức có thể được sử dụng với window,có thể truy cập đường link sao để tham khảo đầy đủ (developer.mozilla.org/en-US/docs/Web/API/Window).

### Events

- Sự kiện là một hành động có thể được phát hiện bằng JavaScript, chẳng hạn như khi tải tài liệu hoặc khi người dùng nhấp vào một phần tử hoặc chỉ di chuyển chuột qua phần tử đó. HTML 4.0 giúp tập lệnh có thể được gắn với các sự kiện trên trang, cho dù do người dùng, chính trình duyệt hay các tập lệnh khác khởi tạo. Điều này được gọi là ràng buộc sự kiện.

- Bạn có thể chỉ định hàm sẽ được chạy trong một thuộc tính trong phần đánh dấu.ví dụ:

/\* myFunction will now run when the user clicks anything within 'body' \*/

- Mặc dù vẫn hoạt động nhưng đây là một cách lỗi thời để gắn các sự kiện vào các thành phần trong trang. Nên tránh điều này vì lý do tương tự như việc chúng ta tránh sử dụng thuộc tính kiểu trong đánh dấu của mình để áp dụng kiểu cho các thành phần riêng lẻ.

- Trong trường hợp này, nó làm mờ ranh giới giữa lớp ngữ nghĩa và lớp hành vi trên các trang của chúng tôi và có thể nhanh chóng dẫn đến cơn ác mộng về bảo trì.

**Là một phương pháp**

- Là một cách tiếp cận có phần lỗi thời khác để đính kèm các sự kiện, mặc dù nó giữ mọi thứ nghiêm ngặt trong tập lệnh của chúng tôi.

window.onclick = myFunction; /\* myFunction sẽ chạy khi người dùng nhấp vào bất cứ thứ gì trong cửa sổ trình duyệt \*/

- Chúng ta cũng có thể sử dụng hàm ẩn danh thay vì hàm được xác định trước:

window.onclick = function() {

/\* Bất kỳ mã nào được đặt ở đây sẽ chạy khi người dùng nhấp vào bất cứ thứ gìtrong cửa sổ trình duyệt \*/

};

- Cách tiếp cận này có lợi ích là vừa đơn giản vừa dễ bảo trì, nhưng có một nhược điểm khá lớn.

window.onclick = myFunction;

window.addEventListener

**addEventListener**

- Mặc dù thoạt nhìn phức tạp hơn một chút, nhưng cách tiếp cận này cho phép chúng ta giữ logic trong tập lệnh của mình và cho phép chúng ta thực hiện nhiều liên kết trên một đối tượng. Cú pháp dài dòng hơn một chút.

window.addEventListener("click", myFunction);

- Lưu ý rằng chúng ta bỏ qua từ “on” trước đó trong trình xử lý sự kiện bằng cú pháp này.

- Giống như phương thức trước, addEventListener() cũng có thể được sử dụng với hàm ẩn danh:

window.addEventListener("click", function(e) { });

- Đây chỉ là phần giới thiệu ngắn gọn nên tôi khuyên bạn nên tìm hiểu thêm thông tin về addEventListener() tại trang “eventTarget.addEventListener” trên Tài liệu WebMDN (developer.mozilla.org/en/DOM/element.addEventListener).

### DOM

**DOM là gì?**

DOM là một tiêu chuẩn W3C (World Wide Web Consortium).

DOM xác định một tiêu chuẩn để truy cập tài liệu:

"Mô hình đối tượng tài liệu W3C (DOM) là một nền tảng và giao diện trung lập về ngôn ngữ cho phép các chương trình và tập lệnh truy cập và cập nhật động nội dung, cấu trúc và kiểu dáng của tài liệu."

Tiêu chuẩn W3C DOM được chia thành 3 phần khác nhau:

Core DOM - mô hình chuẩn cho mọi loại tài liệu

XML DOM - mô hình chuẩn cho tài liệu XML

HTML DOM - mô hình chuẩn cho tài liệu HTML

**DOM HTML là gì?**

HTML DOM là một mô hình đối tượng và giao diện lập trình tiêu chuẩn cho HTML. Nó định nghĩa:

·Các phần tử HTML dưới dạng đối tượng

·Thuộc tính của tất cả các phần tử HTML

·Các phương pháp truy cập tất cả các phần tử HTML

·Các sự kiện cho tất cả các phần tử HTML

Nói cách khác: HTML DOM là một tiêu chuẩn về cách lấy, thay đổi, thêm hoặc xóa các phần tử HTML.

**Giao diện lập trình DOM**

HTML DOM có thể được truy cập bằng JavaScript (và bằng các ngôn ngữ lập trình khác).

Trong DOM, tất cả các phần tử HTML được định nghĩa là đối tượng .

Giao diện lập trình là các thuộc tính và phương thức của từng đối tượng.

Thuộc tính là một giá trị mà bạn có thể nhận hoặc đặt (như thay đổi nội dung của phần tử HTML).

Phương thức là một hành động bạn có thể thực hiện (như thêm hoặc xóa phần tử HTML).

**Đối tượng tài liệu HTML DOM**

Đối tượng tài liệu đại diện cho trang web của bạn.

Nếu bạn muốn truy cập bất kỳ phần tử nào trong trang HTML, bạn luôn bắt đầu bằng việc truy cập đối tượng tài liệu.

Dưới đây là một số ví dụ về cách bạn có thể sử dụng đối tượng tài liệu để truy cập và thao tác HTML.

**Tìm các phần tử HTML**

document.getElementById(id)

document.getElementsByTagName(name)

document.getElementsByClassName(name)

**Thay đổi các phần tử HTML**

element.innerHTML =  new html content

element.attribute = new value

element.style.property = new style

element.setAttribute(attribute, value)

**Thêm và xóa các phần tử**

document.createElement(element)

document.removeChild(element)

document.appendChild(element)

document.replaceChild(new, old)

document.write(text)

**Thêm trình xử lý sự kiện**

document.getElementById(id).onclick = function(){code}

Tìm đối tượng HTML

document.anchors

document.applets

document.baseURI

document.body

document.cookie

document.doctype

document.documentElement

document.documentMode

document.documentURI

document.domain

document.domConfig

document.embeds

document.forms

document.head

document.images

document.implementation

document.inputEncoding

document.lastModified

document.links

document.readyState

document.referrer

document.scripts

document.strictErrorChecking

document.title

document.URL

**Tìm các phần tử HTML**

Thông thường, với JavaScript, bạn muốn thao tác với các phần tử HTML.

**Để làm như vậy, trước tiên bạn phải tìm các phần tử. Có nhiều hướng khác nhau để làm điều đó:**

·Tìm các phần tử HTML theo id

·Tìm các phần tử HTML theo tên thẻ

·Tìm các phần tử HTML theo tên lớp

·Tìm các phần tử HTML bằng bộ chọn CSS

·Tìm các phần tử HTML bằng bộ sưu tập đối tượng HTML

**Tìm phần tử HTML theo Id**

Cách dễ nhất để tìm một phần tử HTML trong DOM là sử dụng id phần tử.

Vd:

<!DOCTYPE html>

<html>

<body>

<p id="intro">hello</p>

<p>zhongli<b>venti</b>raiden</p>

<p id="demo"></p>

<script>

const element = document.getElementById("intro");

document.getElementById("demo").innerHTML =

"kq: " + element.innerHTML;

</script>

</body>

</html>

**Tìm các phần tử HTML theo tên lớp**

Nếu bạn muốn tìm tất cả các thành phần HTML có cùng tên lớp, hãy sử dụng getElementsByClassName().

Để thay đổi nội dung của phần tử HTML, hãy sử dụng cú pháp sau:

document.getElementById(id).innerHTML = new HTML

Thay đổi giá trị của một thuộc tính

Để thay đổi giá trị của thuộc tính HTML, hãy sử dụng cú pháp sau:

document.getElem Nội dung HTML động

JavaScript có thể tạo nội dung HTML động:

Ngày : Thứ Hai, 11/03/2024 20:55:46 GMT+0700 ( Giờ Đông Dương)entById(id).attribute = new value;

Không bao giờ sử dụng document.write()sau khi tài liệu được tải. Nó sẽ ghi đè lên tài liệu.

**Ví dụ về JavaScript**

function validateForm() {

  let x = document.forms["myForm"]["fname"].value;

  if (x == "") {

    alert("Name must be filled out");

     return false;

  }

}

JavaScript có thể xác thực đầu vào số

JavaScript thường được sử dụng để xác thực dữ liệu đầu vào dạng số:

Nếu bạn muốn tìm tất cả các phần tử HTML khớp với một bộ chọn CSS được chỉ định (id, tên lớp, loại, thuộc tính, giá trị của thuộc tính, v.v.), hãy sử dụng querySelectorAll()phương thức này.

Thay đổi nội dung HTML

Cách dễ nhất để sửa đổi nội dung của phần tử HTML là sử dụng thuộc tính innerHTML.

Để thay đổi nội dung của phần tử HTML, hãy sử dụng cú pháp sau:

document.getElementById(id).innerHTML = new HTML

Để thay đổi giá trị của thuộc tính HTML, hãy sử dụng cú pháp sau:

document.getE JavaScript có thể tạo nội dung HTML động:

Ngày : Thứ Hai, 11/03/2024 20:55:46 GMT+0700 ( Giờ Đông Dương)

lementById(id).attribute = new value

Trong JavaScript, document.write()có thể được sử dụng để ghi trực tiếp vào luồng đầu ra HTML:

Xác thực mẫu HTML có thể được thực hiện bằng JavaScript.

Nếu trường biểu mẫu (fname) trống, hàm này sẽ cảnh báo một thông báo và trả về sai để ngăn việc gửi biểu mẫu:

JavaScript thường được sử dụng để xác thực dữ liệu đầu vào dạng số:

Trình duyệt có thể thực hiện xác thực biểu mẫu HTML một cách tự động:

Nếu trường biểu mẫu (fname) trống, requiredthuộc tính sẽ ngăn việc gửi biểu mẫu này:

Để thay đổi kiểu của phần tử HTML, hãy sử dụng cú pháp sau:

document.getElementById(id).style.property = new style

HTML DOM cho phép bạn thực thi mã khi có sự kiện xảy ra.

Các sự kiện được trình duyệt tạo ra khi "sự việc xảy ra" với các phần tử HTML:

·-Một phần tử được nhấp vào

·-Trang đã được tải

·-Các trường đầu vào được thay đổi

Bạn sẽ tìm hiểu thêm về các sự kiện trong chương tiếp theo của hướng dẫn này.

Một trang web cơ bản

Để trình bày cách tạo hoạt ảnh HTML bằng JavaScript, chúng tôi sẽ sử dụng một trang web đơn giản

Ví dụ về các sự kiện HTML:

·Khi người dùng click chuột

·Khi một trang web được tải

·Khi một hình ảnh đã được tải

·Khi chuột di chuyển qua một phần tử

·Khi một trường đầu vào được thay đổi

·Khi một biểu mẫu HTML được gửi

·Khi người dùng nhấn một phím

Để gán sự kiện cho các phần tử HTML, bạn có thể sử dụng thuộc tính sự kiện.

HTML DOM cho phép bạn gán các sự kiện cho các thành phần HTML bằng JavaScript:

Sự kiện onloadvà onunloadđược kích hoạt khi người dùng vào hoặc rời khỏi trang.

Sự kiện này onloadcó thể được sử dụng để kiểm tra loại trình duyệt và phiên bản trình duyệt của khách truy cập, đồng thời tải phiên bản phù hợp của trang web dựa trên thông tin.

Các sự kiện onloadvà onunloadcó thể được sử dụng để xử lý cookie.

Theo tiêu chuẩn W3C HTML DOM, mọi thứ trong tài liệu HTML đều là một nút:

·-Toàn bộ tài liệu là một nút tài liệu

·-Mỗi phần tử HTML là một nút phần tử

·-Văn bản bên trong các phần tử HTML là các nút văn bản

·-Mỗi thuộc tính HTML là một nút thuộc tính (không được dùng nữa)

·-Tất cả các bình luận đều là nút bình luận

·-Với HTML DOM, tất cả các nút trong cây nút có thể được truy cập bằng JavaScript.

·-Các nút mới có thể được tạo và tất cả các nút có thể được sửa đổi hoặc xóa.

Các nút trong cây nút có mối quan hệ phân cấp với nhau.

Các thuật ngữ cha mẹ, con cái và anh chị em được sử dụng để mô tả các mối quan hệ.

·-Trong cây nút, nút trên cùng được gọi là nút gốc (hoặc nút gốc)

·-Mỗi nút có chính xác một nút cha, ngoại trừ nút gốc (không có nút cha)

·-Một nút có thể có nhiều nút con

·-Anh chị em ruột (anh chị em) là các nút có cùng cha mẹ

Từ HTML ở trên bạn có thể đọc:

·<html>là nút gốc

·<html>không có cha mẹ

·<html>là cha mẹ của <head>và<body>

·<head>là đứa con đầu lòng của<html>

·<body>là đứa con cuối cùng của<html>

Và:

·<head>có một đứa con:<title>

·<title>có một con (nút văn bản): "Hướng dẫn DOM"

·<body>có hai con: <h1>và<p>

·<h1>có một đứa con: "DOM Bài học một"

·<p>có một đứa con: "Xin chào thế giới!"

·<h1>và <p>là anh em ruột

Bạn có thể sử dụng các thuộc tính nút sau để điều hướng giữa các nút bằng JavaScript:

·-*parentNode*

*·-childNodes[nodenumber]*

*·-firstChild*

*·-lastChild*

*·-nextSibling*

*·-previousSibling*

Một lỗi phổ biến trong quá trình xử lý DOM là mong muốn một nút phần tử chứa văn bản.

Ví dụ:

<title id="demo">DOM Tutorial</title>

Nút phần tử <title>(trong ví dụ trên) không chứa văn bản.

Nó chứa một nút văn bản có giá trị "Hướng dẫn DOM".

Giá trị của nút văn bản có thể được truy cập bằng innerHTMLthuộc tính của nút:

myTitle = document.getElementById("demo").innerHTML;

Truy cập thuộc tính bên trongHTML cũng giống như truy cập vào thuộc nodeValue tính con đầu tiên:

myTitle = document.getElementById("demo").firstChild.nodeValue;

Việc truy cập đứa trẻ đầu tiên cũng có thể được thực hiện như thế này:

myTitle = document.getElementById("demo").childNodes[0].nodeValue;

Tất cả (3) ví dụ sau đây truy xuất văn bản của một <h1>phần tử và sao chép nó vào một <p>phần tử:

Trong hướng dẫn này, chúng tôi sử dụng thuộc tính InnerHTML để truy xuất nội dung của một phần tử HTML.

Tuy nhiên, việc tìm hiểu các phương pháp khác ở trên rất hữu ích để hiểu cấu trúc cây và cách điều hướng của DOM.

Có hai thuộc tính đặc biệt cho phép truy cập vào toàn bộ tài liệu:

·-document.body- Phần thân của tài liệu

·-document.documentElement- Giấy tờ đầy đủ

Thuộc nodeNametính chỉ định tên của một nút.

·-nodeName ở chế độ chỉ đọc

·-nodeName của một nút phần tử giống với tên thẻ

·-nodeName của một nút thuộc tính là tên thuộc tính

·-nodeName của nút văn bản luôn là #text

-nodeName của nút tài liệu luôn là #documen

Thuộc nodeValuetính chỉ định giá trị của một nút.

·-nodeValue cho các nút phần tử lànull

·-nodeValue cho các nút văn bản chính là văn bản

·-nodeValue cho các nút thuộc tính là giá trị thuộc tính

Thuộc nodeTypetính chỉ được đọc. Nó trả về loại nút.

document.getElementById("id02").innerHTML = document.getElementById("id01").nodeTy<!DOCTYPE html>

Các thuộc tính nodeType quan trọng nhất là:

DOCUMENT\_TYPE\_NODE

COMMENT\_NODE

TEXT\_NODE

ATTRIBUTE\_NODE

ELEMENT\_NODE

Loại 2 không được dùng nữa trong HTML DOM (nhưng vẫn hoạt động). Nó không được dùng nữa trong XML DOM.

Để thêm một phần tử mới vào HTML DOM, trước tiên bạn phải tạo phần tử (nút phần tử), sau đó nối nó vào phần tử hiện có

Phương appendChild()thức trong ví dụ trước đã thêm phần tử mới làm phần tử con cuối cùng của phần tử cha.

Nếu không muốn, bạn có thể sử dụng insertBefore()phương pháp:

Để xóa phần tử HTML, hãy sử dụng remove() phương thức:

Đối với các trình duyệt không hỗ trợ *remove()*phương thức, bạn phải tìm nút cha để xóa một phần tử:

Để thay thế một phần tử vào HTML DOM, hãy sử dụng replaceChild()phương thức:

Phương *getElementsByTagName()*thức trả về một*HTMLCollection*đối tượng.

Một *HTMLCollection*đối tượng là một danh sách (bộ sưu tập) các phần tử HTML dạng mảng.

Đoạn mã sau chọn tất cả *<p>*các thành phần trong tài liệu:

Ví dụ

*const myCollection = document.getElementsByTagName("p");*

Các phần tử trong bộ sưu tập có thể được truy cập bằng số chỉ mục.

Để truy cập phần tử <p> thứ hai, bạn có thể viết:

myCollection[1]

Lưu ý: Chỉ số bắt đầu từ 0.

Thuộc*lengthtính* xác định số HTMLCollection có thể trông giống như một mảng nhưng thực tế không phải vậy.

Bạn có thể lặp qua danh sách và tham chiếu đến các phần tử bằng một số (giống như một mảng).

Tuy nhiên, bạn không thể sử dụng các phương thức mảng như valueOf(), pop(), push() hoặc join() trên HTMLCollection.

phần tử trong một *HTMLCollection:*

Một NodeListđối tượng là một danh sách (tập hợp) các nút được trích xuất từ ​​một tài liệu.

Một NodeListđối tượng gần giống như một HTMLCollectionđối tượng.

Một số trình duyệt (cũ hơn) trả về đối tượng NodeList thay vì HTMLCollection cho các phương thức như getElementsByClassName().

Tất cả các trình duyệt đều trả về một đối tượng NodeList cho thuộc tính này childNodes.

Hầu hết các trình duyệt đều trả về đối tượng NodeList cho phương thức này querySelectorAll().

Đoạn mã sau chọn tất cả <p>các nút trong tài liệu:

const myNodeList = document.querySelectorAll("p");

Các phần tử trong NodeList có thể được truy cập bằng số chỉ mục.

Để truy cập nút <p> thứ hai, bạn có thể viết:

myNodeList[1]

Polyfill là một thuật ngữ do Remy Sharp đặt ra để mô tả một “shim” JavaScript giúp bình thường hóa các hành vi khác nhau giữa các trình duyệt (remysharp. com/2010/10/08/what-is-a-polyfill).

Shim HTML5 (hoặc shiv)

Picturefill

Để sử dụng Picturefill, hãy tải xuống tập lệnh và thêm nó vào phần đầu tài liệu. Tập lệnh đầu tiên tạo thành phần hình ảnh cho các trình duyệt không nhận ra nó. Tập lệnh thứ hai gọi chính tập lệnh Picturefill và thuộc tính async cho trình duyệt biết nó có thể tải Picturefill một cách không đồng bộ—nghĩa là không cần đợi tập lệnh hoàn tất trước khi tải phần còn lại của tài liệu.

*<head>*

*document.createElement("picture" );*

*<script>*

*<script src=”picturefill.js” async>*

*</script>*

*</head>*

Mặt khác, các trình duyệt không có JavaScript cũng không hỗ trợ thành phần hình ảnh sẽ chỉ nhìn thấy văn bản thay thế cho hình ảnh. Tải xuống Picturefill và nhận thông tin về cách sử dụng nó tại scottjehl.github.io/picturefill/.

## Kết chương

Trong chương này, chúng ta đã tìm hiểu về ba thành phần cơ bản nhưng cực kỳ quan trọng của lập trình web: HTML, CSS và JavaScript.

HTML, ngôn ngữ đánh dấu siêu văn bản, là cốt lõi của mọi trang web. Nó giúp chúng ta xây dựng cấu trúc của trang web bằng cách sử dụng các thẻ và phần tử. Chúng ta đã thảo luận về cách sử dụng các thẻ HTML để tạo ra các phần tử như tiêu đề, đoạn văn bản, hình ảnh và liên kết, cũng như cách chúng tương tác với nhau.

CSS, ngôn ngữ kiểu trang, là công cụ mạnh mẽ để tùy chỉnh giao diện của trang web. Chúng ta đã tìm hiểu về cách sử dụng CSS để thay đổi màu sắc, font chữ, kích thước và vị trí của các phần tử HTML, tạo ra giao diện trực quan và thú vị cho người dùng.

Cuối cùng, JavaScript là ngôn ngữ lập trình cho phía client, cho phép chúng ta thêm tính năng động và tương tác vào trang web. Chúng ta đã thảo luận về cách sử dụng JavaScript để thực hiện các hành động như xử lý sự kiện, thay đổi nội dung và tương tác với người dùng.

Bằng cách hiểu và vận dụng HTML, CSS và JavaScript một cách hiệu quả, chúng ta có thể xây dựng những trang web đẹp và chức năng, mang lại trải nghiệm tốt nhất cho người dùng.

# Phân tích thiết kế hệ thống

## Mô tả bài toán

### **Giao diện và trải nghiệm người dùng**

-Thiết kế giao diện cho khách hàng dễ dàng sử dụng đẹp mắt , hình ảnh sắc nét và thân thiện với người dùng .

-Trải nghiệm người dùng cần được tối ưu hóa để đảm bảo khách hàng dễ dàng ctìm kiếm sản phẩm và thực hiện giao dịch.

### **Tối ưa chuẩn SEO**

-Website cần dược tối ưa chuẩn SEO để xuất hiện trong kết quả tìm kiếm các công cụ hỗ trợ tìm kiếm của khách hàng như google.

- Sử dụng từ khóa liên quan tới đồng hồ và nhãn hiệu ,ngành công nghiệp.

### Tốc độ tải trang nhanh

-Việc tải trang giúp khách hàng tránh việc mất thời gian, tạo thiện cảm với khách hàng ,web cần được tải tải nhanh để cung cấp trải nghiệm tốt cho người dùng.

-Tối ưa hóa hình ảnh và mã nguồn để giảm thời gian tải trang.

### Responsive design (tương thích với mọi thiết bị)

-Website cần hiển thị giao diện đẹp trên mọi thiết bị , bao gồm laptop,máy tính,tablet và điện thoại di động.

### Bảo mật cao

-Đẩm bảo thông tin khách hàng và giao dịch được bảo mật.

## Đặc tả các yêu cầu

### Yêu cầu chức năng

-Hiển thị sản phẩm: Trang web bán đồng hồ thường hiển thị danh sách các sản phẩm đồng hồ với hình ảnh đẹp và chi tiết. Điều này giúp khách hàng có cái nhìn rõ ràng về sản phẩm trước khi quyết định mua.

-Thông tin sản phẩm: Trang web cung cấp thông tin chi tiết về từng sản phẩm, bao gồm giá cả, mô tả, thông số kỹ thuật và các tính năng đặc biệt. Khách hàng có thể dễ dàng tìm hiểu về sản phẩm mà họ quan tâm.

-Chính sách giao hàng và hỗ trợ khách hàng: Trang web bán đồng hồ thường cung cấp thông tin về chính sách giao hàng, bảo hành, đổi trả và hỗ trợ khách hàng. Điều này giúp khách hàng cảm thấy an tâm khi mua sắm trực tuyến.

-Tương tác khách hàng: Một số tính năng tương tác khách hàng bao gồm cho phép khách hàng đánh giá sản phẩm, viết bình luận và chia sẻ trải nghiệm của họ. Điều này tạo ra sự gắn kết giữa khách hàng và thương hiệu.

-Ngoài ra, việc thiết kế website bán đồng hồ cần phải tối ưu hóa cho SEO (tối ưu hóa công cụ tìm kiếm) và đảm bảo giao diện di động thân thiện để phục vụ tốt cho người dùng trên các thiết bị di động.

### Yêu cầu phi chức năng

-Tìm kiếm: Phi chức năng này cho phép người dùng tìm kiếm sản phẩm theo từ khóa hoặc danh mục. Tìm kiếm hiệu quả giúp khách hàng dễ dàng tìm thấy sản phẩm mình muốn mua.

-Đăng ký và đăng nhập: Phi chức năng này liên quan đến việc quản lý tài khoản người dùng. Khách hàng có thể đăng ký tài khoản mới, đăng nhập vào tài khoản hiện có để theo dõi đơn hàng, lưu sản phẩm yêu thích và thực hiện các giao dịch.

-Giỏ hàng: Phi chức năng giỏ hàng cho phép khách hàng thêm sản phẩm vào giỏ hàng, xem tổng giá trị đơn hàng và tiến hành thanh toán.

-Bình luận và đánh giá sản phẩm: Khách hàng có thể viết bình luận, đánh giá sản phẩm và chia sẻ trải nghiệm của họ. Điều này giúp tạo ra sự tin tưởng và tương tác giữa khách hàng và cộng đồng mua sắm.

-Liên hệ và hỗ trợ khách hàng: Cung cấp thông tin liên hệ và hỗ trợ khách hàng, bao gồm số điện thoại, email hoặc chat trực tuyến. Điều này giúp khách hàng có thể liên hệ khi cần hỗ trợ hoặc có thắc mắc.

-Chia sẻ trên mạng xã hội: Phi chức năng này cho phép khách hàng chia sẻ sản phẩm yêu thích lên các mạng xã hội như Facebook, Instagram, Twitter, v.v. Điều này giúp tăng khả năng tiếp cận và quảng cáo cho trang web.

-Nhớ rằng, việc thiết kế phi chức năng cũng cần phải tối ưu hóa cho trải nghiệm người dùng và đảm bảo tính bảo mật của thông tin cá nhân. Nếu bạn đang xây dựng trang web bán đồng hồ, hãy cân nhắc tích hợp các phi chức năng trên để tạo ra một trải nghiệm mua sắm tốt cho khách hàng của bạn

## Thiết kế dữ liệu

Tìm và sắp xếp thông tin cần thiết: Thu thập thông tin cần lưu trữ và xác định các thực thể, thuộc tính và mối quan hệ giữa chúng.

Phân chia thông tin vào các bảng: Xác định các bảng dữ liệu và phân chia thông tin vào chúng. Mỗi bảng đại diện cho một thực thể hoặc mối quan hệ.

Biến mục thông tin thành các cột: Đối với mỗi bảng, xác định các cột (hoặc trường) để lưu trữ thông tin cụ thể.

Chỉ định khóa chính: Xác định khóa chính cho mỗi bảng. Khóa chính là trường duy nhất để xác định mỗi bản ghi trong bảng.

Tạo mối quan hệ cho bảng: Nếu có mối quan hệ giữa các thực thể, bạn cần xác định mối quan hệ này trong cơ sở dữ liệu.

Tinh chỉnh thiết kế: Kiểm tra lại thiết kế để đảm bảo tính logic và hiệu quả. Điều này bao gồm việc kiểm tra các ràng buộc, kiểu dữ liệu và cấu trúc bảng.

Áp dụng các quy tắc chuẩn hóa: Sử dụng các quy tắc chuẩn hóa để loại bỏ sự dư thừa và đảm bảo tính nhất quán của dữ liệu.

### Mô hình dữ liệu

Trình bày mô hình ERD

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

Bảng 2.1 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 2.2 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)

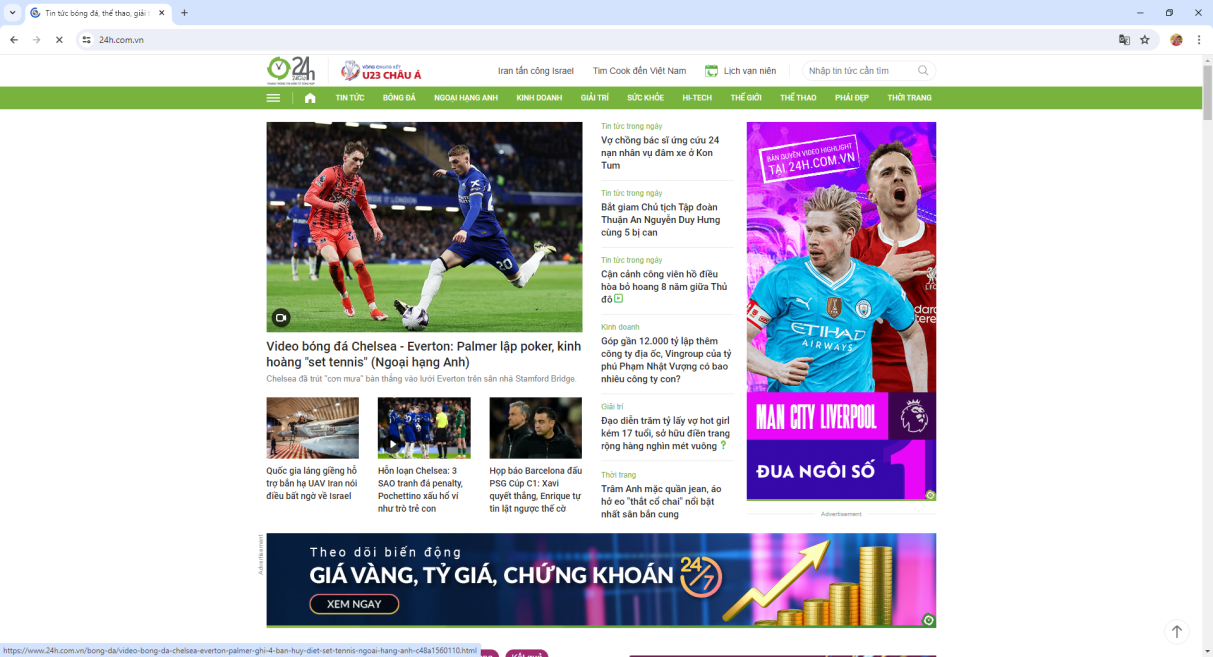
(4)

## Thiết kế giao diện

### Sơ đồ hệ thống

Hình 2.1 Sơ đồ hệ thống

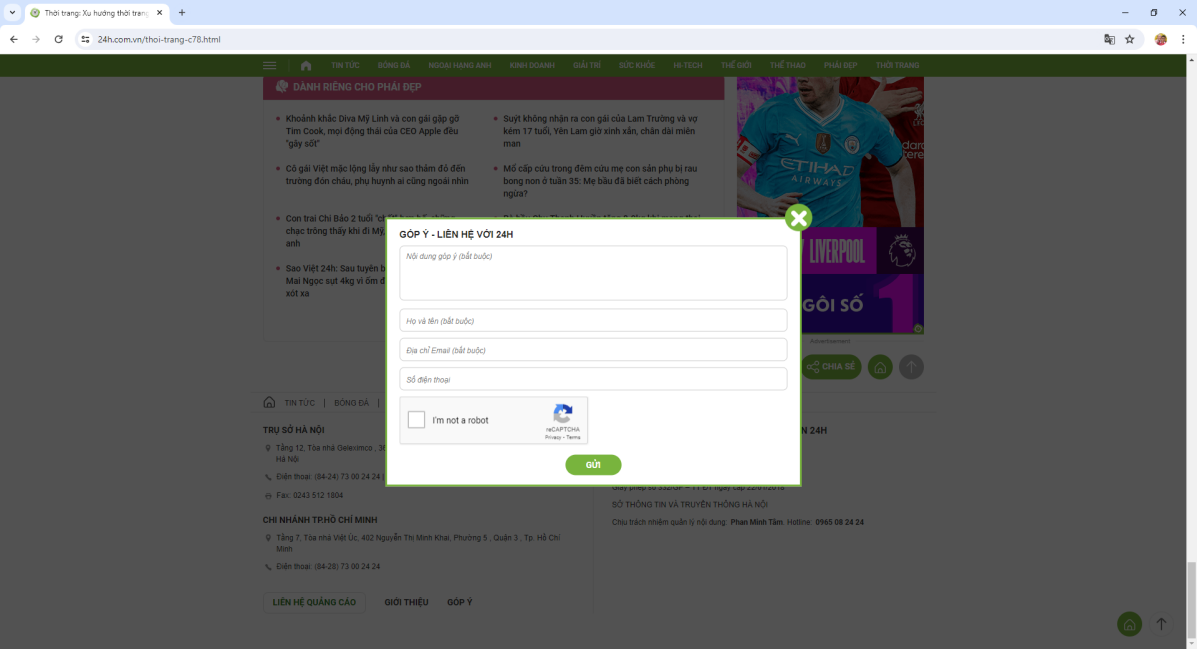
### Giao diện trang chủ người dùng



Hình 2.2 Giao diện trang chủ của người dùng

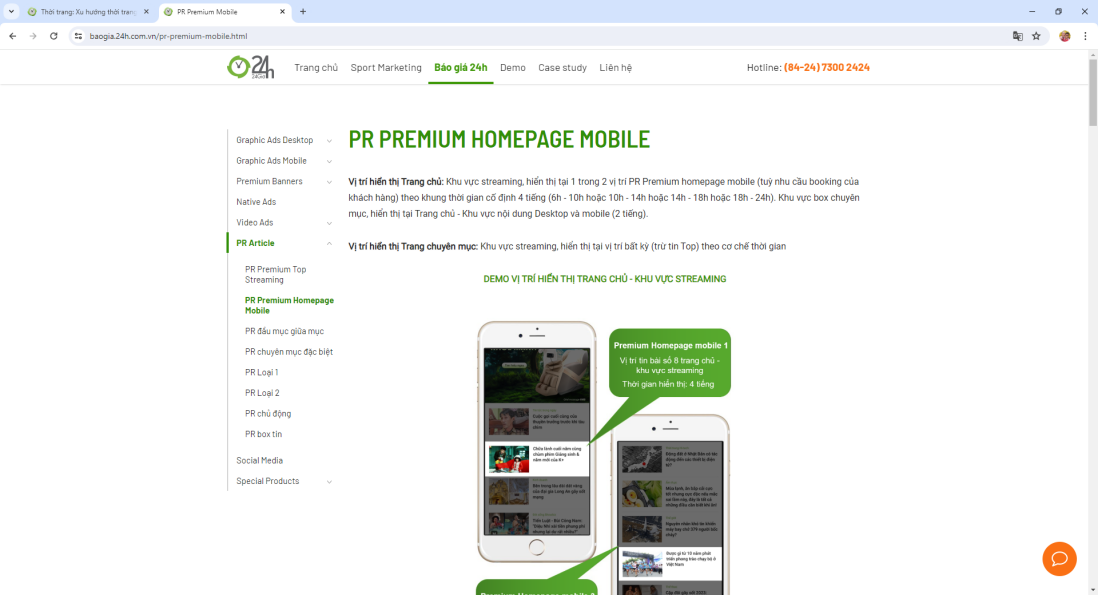
Mô tả chi tiết về màn hình bao gồm các thành phần và cách bố cục

#### Giao diện chức năng …



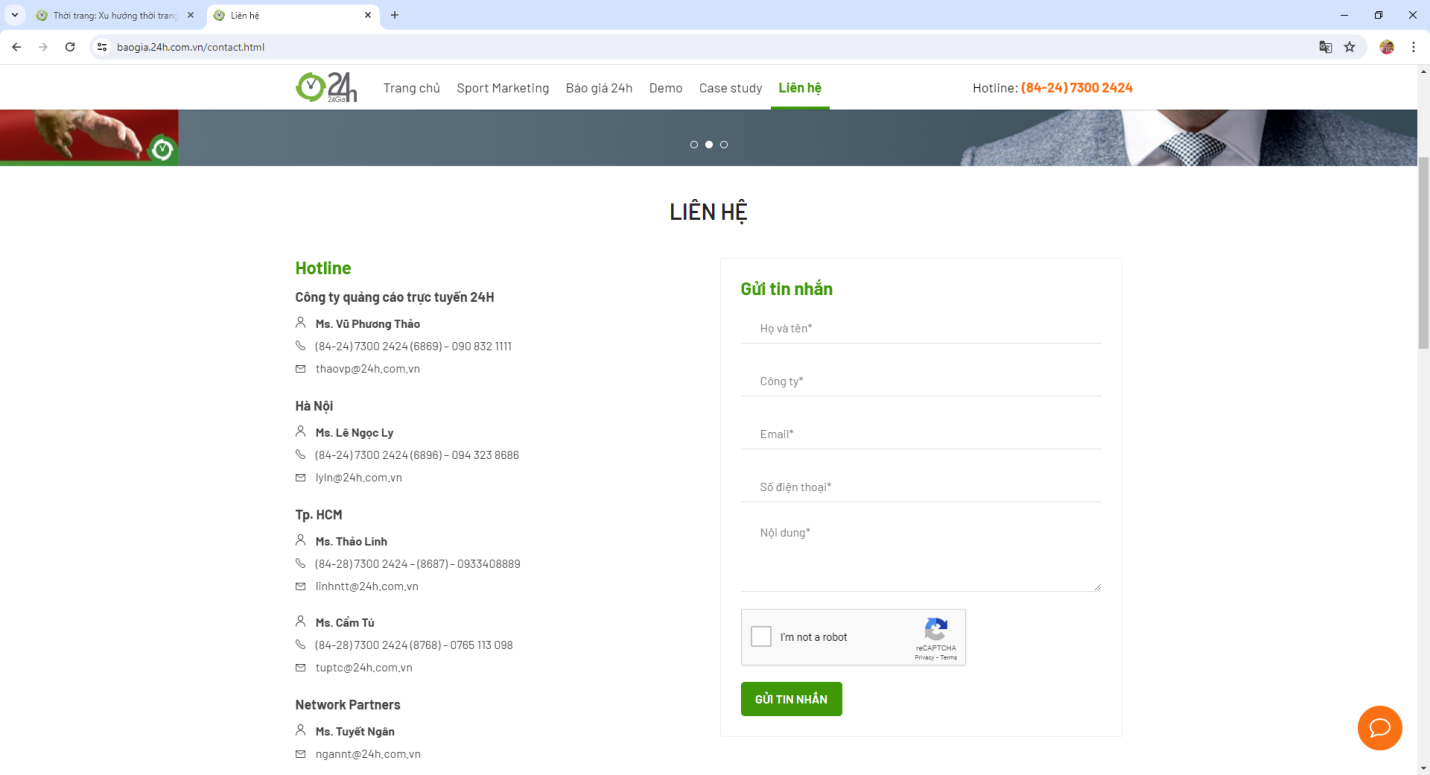
Hình 2.3 Giao diện chức năng …

#### Giao diện chức năng …



Hình 2.4 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