



Group:

Name:	Khuru Chí Khang	22161266
	Lê Anh Quát	22161311
	Lê Quốc Đạt	22161234

A. HTML (TAG, FEATURES), CSS (STYLE)

I. Thẻ HTML

1. Thẻ HTML Headings

a. Chức năng

Tiêu đề HTML là tiêu đề hoặc phụ đề mà bạn muốn hiển thị trên trang web.

b. Cú pháp

Tiêu đề HTML được định nghĩa bằng thẻ `<h1>` đến `<h6>`.

`<h1>` định nghĩa tiêu đề quan trọng nhất. `<h6>` định nghĩa tiêu đề ít quan trọng nhất.

<code><h1>Heading</code>	<code>1</h1></code>
<code><h2>Heading</code>	<code>2</h2></code>
<code><h3>Heading</code>	<code>3</h3></code>
<code><h4>Heading</code>	<code>4</h4></code>
<code><h5>Heading</code>	<code>5</h5></code>
<code><h6>Heading</code>	<code>6</h6></code>

* Chú ý: Trình duyệt tự động thêm một khoảng trắng (lè) trước và sau tiêu đề.

c. Ví dụ minh họa

Code mẫu	Kết quả
<code><!DOCTYPE html></code>	Văn bản 1
<code><html></code>	
<code><body></code>	
<code><h1>Văn bản 1</h1></code>	Văn bản 2
<code><h2>Văn bản 2</h2></code>	
<code><h3>Văn bản 3</h3></code>	Văn bản 3
<code><h4>Văn bản 4</h4></code>	
<code><h5>Văn bản 5</h5></code>	Văn bản 4
<code><h6>Văn bản 6</h6></code>	
<code></body></code>	Văn bản 5
<code></html></code>	Văn bản 6

2. Thẻ HTML Paragraphs

a. Chức năng

Đoạn văn luôn bắt đầu trên một dòng mới và thường là một khối văn bản.

b. Cú pháp

Phần tử HTML `<p>` xác định một đoạn văn.

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

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

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <p>Đây là dòng 1.</p> <p>Đây là dòng 2.</p> <p>Đây là dòng 3.</p> </body> </html></pre>	Đây là dòng 1. Đây là dòng 2. Đây là dòng 3.

3. Thẻ HTML Styles

a. Chức năng: Thuộc tính style của HTML được sử dụng để thêm kiểu cho một phần tử, chẳng hạn như màu sắc, phông chữ, kích thước, v.v.

b. Cú pháp:

```
<tagname style="property:value;">
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><p style="color:red;font-size:50px">Tôi tên là Đạt</p></pre>	Tôi tên là Đạt

4. Thẻ HTML Fortmatting

a. Chức năng

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

b. Cú pháp

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

- `` - Bold text
- `` - Important text

- `<i>` - Italic text
- `` - Emphasized text
- `<mark>` - Marked text
- `<small>` - Smaller text
- `` - Deleted text
- `<ins>` - Inserted text
- `<sub>` - Subscript text
- `<sup>` - Superscript text

Phân tử `` của HTML định nghĩa văn bản in đậm, không có bất kỳ yếu tố quan trọng nào khác.

```
<b>This text is bold</b>
```

Phân tử HTML `` định nghĩa văn bản có tầm quan trọng lớn. Nội dung bên trong thường được hiển thị bằng chữ in đậm.

```
<strong>This text is important!</strong>
```

Phân tử HTML `<i>` định nghĩa một phân văn bản theo giọng điệu hoặc tâm trạng thay thế. Nội dung bên trong thường được hiển thị bằng chữ nghiêng.

Mẹo: Thẻ `<i>` thường được sử dụng để chỉ thuật ngữ kỹ thuật, cụm từ trong ngôn ngữ khác, ý nghĩ, tên tàu, v.v.

```
<i>This text is italic</i>
```

Phân tử HTML `` định nghĩa văn bản được nhấn mạnh. Phân tử HTML `<mark>` xác định văn bản cần được đánh dấu hoặc tô sáng:

Mẹo: Trình đọc màn hình sẽ phát âm các từ trong `` với sự nhấn mạnh, sử dụng trọng âm của động từ.

```
<em>This text is emphasized</em>
```

Phân tử HTML `<small>` xác định văn bản nhỏ hơn:

```
<small>This is some smaller text.</small>
```

Phân tử HTML `<mark>` xác định văn bản cần được đánh dấu hoặc tô sáng:

```
<p>Do not forget to buy <mark>milk</mark> today.</p>
```

Phân tử HTML `` định nghĩa văn bản đã bị xóa khỏi tài liệu. Trình duyệt thường sẽ gạch một dòng qua văn bản đã xóa:

```
<p>My favorite color is <del>blue</del> red.</p>
```

Phân tử HTML `<ins>` định nghĩa một văn bản đã được chèn vào tài liệu. Trình duyệt thường sẽ gạch chân văn bản đã chèn:

```
<p>My favorite color is <del>blue</del> <ins>red</ins>. </p>
```

Phân tử HTML `<sub>` định nghĩa văn bản chỉ số dưới. Văn bản chỉ số dưới xuất hiện nửa ký tự bên dưới dòng bình thường và đôi khi được hiển thị bằng phông chữ nhỏ hơn. Văn bản chỉ số dưới có thể được sử dụng cho các công thức hóa học, như H₂O:

```
<p>This is <sub>subscripted</sub> text.</p>
```

Phân tử HTML `<sup>` định nghĩa văn bản chỉ số trên. Văn bản chỉ số trên xuất hiện nửa ký tự trên dòng bình thường và đôi khi được hiển thị ở phông chữ nhỏ hơn. Văn bản chỉ số trên có thể được sử dụng cho chú thích, như WWW^[1]:

```
<p>This is <sup>superscripted</sup> text.</p>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <p>MU vô địch</p> <p><i>Chữ in nghiêng</i></p> <p>This is<sub> bot</sub> and <sup>top</sup></p> </body> </html></pre>	<p>MU vô địch</p> <p><i>Chữ in nghiêng</i></p> <p>This is_{bot} and^{top}</p>

5. Thẻ HTML Quotations

a. Chức năng

- Phân tử HTML `<blockquote>` xác định một phần được trích dẫn từ một nguồn khác.

Các trình duyệt thường thực hiện cách phân tử `<blockquote>`.

- Thẻ HTML `<q>` xác định một trích dẫn ngắn.

Các trình duyệt thường chèn dấu ngoặc kép xung quanh câu trích dẫn.

- Thẻ HTML `<abbr>` xác định từ viết tắt hoặc từ viết tắt, như "HTML", "CSS", "Mr.", "Dr.", "ASAP", "ATM".

Đánh dấu các chữ viết tắt có thể cung cấp thông tin hữu ích cho trình duyệt, hệ thống dịch thuật và công cụ tìm kiếm.

- Thẻ HTML `<address>` xác định thông tin liên hệ của tác giả/chủ sở hữu tài liệu hoặc bài viết.

Thông tin liên hệ có thể là địa chỉ email, URL, địa chỉ thực, số điện thoại, địa chỉ mạng xã hội, v.v.

Văn bản trong phân tử `<address>` thường hiển thị ở dạng in nghiêng và trình duyệt sẽ luôn thêm dấu ngắt dòng trước và sau phân tử `<address>`.

- Thẻ HTML `<cite>` xác định tiêu đề của một tác phẩm sáng tạo (ví dụ: một cuốn sách, một bài thơ, một bài hát, một bộ phim, một bức tranh, một tác phẩm điêu khắc, v.v.).

BDO là viết tắt của Ghi đè hai chiều.

- Thẻ HTML <bdo> được sử dụng để ghi đè hướng văn bản hiện tại.

b. Cú pháp

- HTML <blockquote>

```
<!DOCTYPE html>
<html>
<body>

<p>Here is a quote from WWF's website:</p>

<blockquote cite="http://www.worldwildlife.org/who/index.html">
For 60 years, WWF has worked to help people and nature thrive. As the world's leading conservation organization, WWF works in nearly 100 countries. At every level, we collaborate with people around the world to develop and deliver innovative solutions that protect communities, wildlife, and the places in which they live.
</blockquote>

</body>
</html>
```

- HTML <q>

```
<!DOCTYPE html>
<html>
<body>

<p>Browsers usually insert quotation marks around the q element.</p>

<p>WWF's goal is to: <q>Build a future where people live in harmony with nature.</q></p>

</body>
</html>
```

- HTML <abbr>

```
<!DOCTYPE html>
<html>
<body>

<p>The <abbr title="World Health Organization">WHO</abbr> was founded in 1948.</p>

<p>Marking up abbreviations can give useful information to browsers, translation systems and search-engines.</p>

</body>
</html>
```

- HTML <address>

```
<!DOCTYPE html>
<html>
<body>
```

```
<p>The HTML address element defines contact information (author/owner) of a document or article.</p>
```

```
<address>  
Written by John Doe.<br>  
Visit us at:<br>  
Example.com<br>  
Box 564, Disneyland<br>  
USA  
</address>  
  
</body>  
</html>
```

- HTML <cite>

```
<!DOCTYPE html>  
<html>  
<body>  
  
<p>The HTML cite element defines the title of a work.</p>  
<p>Browsers usually display cite elements in italic.</p>  
  
  
<p><cite>The Scream</cite> by Edvard Munch. Painted in 1893.</p>  
  
</body>  
</html>
```

- HTML <bdo>

```
<!DOCTYPE html>  
<html>  
<body>  
  
<p>If your browser supports bi-directional override (bdo), the next line will be written from right to left (rtl):</p>  
  
<bdo dir="rtl">This line will be written from right to left</bdo>  
  
</body>  
</html>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<!DOCTYPE html> <html> <body> <p>Here is a quote from WWF's website:</p>	Here is a quote from WWF's website: We have gathered 10 examples of smart city solutions that through practical cases illustrate how data and technology can facilitate energy efficiency, infrastructure, and ways of improving air quality in cities.

```

<blockquote
cite="https://stateofgreen.com/en/news/10-examples-of-smart-city-solutions/">
We have gathered 10 examples of smart city
solutions that through practical cases illustrate
how data and technology can facilitate energy
efficiency, infrastructure, and ways of improving
air quality in cities.
</blockquote>
<p>Steve Jobs từng nói: <q>Stay hungry, stay
foolish.</q></p>
<p><cite>The Great Gatsby</cite> được viết bởi
F. Scott Fitzgerald.</p>

</body>
</html>

```

Steve Jobs từng nói: Stay hungry, stay foolish.
The Great Gatsby được viết bởi F. Scott Fitzgerald.

6. Thẻ HTML Colors

a. Chức năng: Màu HTML được chỉ định bằng tên màu được xác định trước hoặc bằng giá trị RGB, HEX, HSL, RGBA hoặc HSLA.

b. Cú pháp

```

<tagname style="color: value;">
<tagname style="background-color: value;">
<tagname style="border:2px solid value;">

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<h1 style="background-color:DodgerBlue;">Hello World</h1>	Hello World

7. Thẻ HTML CSS

a. Chức năng

- CSS là viết tắt của Cascading Style Sheets.
- CSS tiết kiệm rất nhiều công sức. Nó có thể kiểm soát bố cục của nhiều trang web cùng một lúc.
- Cascading Style Sheets (CSS) được sử dụng để định dạng bố cục của trang web.
- Với CSS, bạn có thể kiểm soát màu sắc, phông chữ, kích thước văn bản, khoảng cách giữa các thành phần, cách các thành phần được định vị và bố trí, hình ảnh nền hoặc màu nền nào sẽ được sử dụng, các màn hình khác nhau cho các thiết bị và kích thước màn hình khác nhau, v.v.!
- Mẹo: Từ cascading có nghĩa là một kiểu được áp dụng cho một thành phần cha cũng sẽ áp dụng cho tất cả các thành phần con trong thành phần cha. Vì vậy, nếu bạn đặt màu của văn bản chính thành "xanh lam", tất cả các tiêu đề, đoạn văn và các thành phần văn bản khác trong nội dung chính cũng sẽ có cùng một màu (trừ khi bạn chỉ định một màu khác).

b. Cú pháp

- CSS có thể được thêm vào tài liệu HTML theo 3 cách:
 - Inline - bằng cách sử dụng thuộc tính style bên trong các phần tử HTML
 - Internal - bằng cách sử dụng phần tử `<style>` trong phần `<head>`
 - External - bằng cách sử dụng phần tử `<link>` để liên kết đến tệp CSS bên ngoài
- Cách phổ biến nhất để thêm CSS là giữ các kiểu trong các tệp CSS bên ngoài. Tuy nhiên, trong hướng dẫn này, chúng ta sẽ sử dụng các kiểu inline và internal, vì cách này dễ trình bày hơn và bạn cũng dễ tự mình thử hơn.
- Inline CSS
 - Inline CSS được sử dụng để áp dụng một kiểu duy nhất cho một phần tử HTML duy nhất.
 - Inline CSS sử dụng thuộc tính style của một phần tử HTML.
 - Ví dụ sau đặt màu văn bản của phần tử `<h1>` thành màu xanh lam và màu văn bản của phần tử `<p>` thành màu đỏ:

```
<h1 style="color:blue;">A Blue Heading</h1>  
<p style="color:red;">A red paragraph.</p>
```

- Internal CSS

- CSS nội bộ được sử dụng để định nghĩa kiểu cho một trang HTML duy nhất.
- CSS nội bộ được định nghĩa trong phần `<head>` của trang HTML, bên trong phần tử `<style>`.
- Ví dụ sau đây đặt màu văn bản của TẤT CẢ các phần tử `<h1>` (trên trang đó) thành màu xanh lam và màu văn bản của TẤT CẢ các phần tử `<p>` thành màu đỏ. Ngoài ra, trang sẽ được hiển thị với màu nền "powderblue":

```
<!DOCTYPE html>  
<html>  
<head>  
<style>  
body {background-color: powderblue;}  
h1 {color: blue;}  
p {color: red;}  
</style>  
</head>  
<body>  
  
<h1>This is a heading</h1>  
<p>This is a paragraph.</p>  
  
</body>  
</html>
```

- External CSS
 - Một bảng định kiểu bên ngoài được sử dụng để xác định kiểu cho nhiều trang HTML.
 - Để sử dụng một bảng định kiểu bên ngoài, hãy thêm liên kết đến bảng đó trong phần `<head>` của mỗi trang HTML.

```
<!DOCTYPE html>
<html>
<head>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

<h1>This is a heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

- Bảng định dạng bên ngoài có thể được viết bằng bất kỳ trình soạn thảo văn bản nào. Tệp không được chứa bất kỳ mã HTML nào và phải được lưu với phần mở rộng .css.
- Tệp "styles.css" trông như thế này:

```
body {
  background-color: powderblue;
}
h1 {
  color: blue;
}
p {
  color: red;
}
```

- Mẹo: Với bảng định dạng bên ngoài, bạn có thể thay đổi giao diện của toàn bộ trang web chỉ bằng cách thay đổi một tệp!
- CSS Colors, Fonts and Sizes:
 - Ở đây, chúng tôi sẽ trình bày một số thuộc tính CSS thường dùng. Bạn sẽ tìm hiểu thêm về chúng sau.
 - Thuộc tính CSS color định nghĩa màu văn bản sẽ được sử dụng.
 - Thuộc tính CSS font-family định nghĩa phông chữ sẽ được sử dụng.
 - Thuộc tính CSS font-size định nghĩa kích thước văn bản sẽ được sử dụng.

```
<!DOCTYPE html>
<html>
<head>
<style>
h1 {
  color: blue;
```

```

    font-family: verdana;
    font-size: 300%;
}
p {
    color: red;
    font-family: courier;
    font-size: 160%;
}

```

</style>

</head>

<body>

<h1>This is a heading</h1>

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

</body>

</html>

- CSS Border

- Thuộc tính border của CSS định nghĩa đường viền xung quanh một phần tử HTML.
- Mẹo: Bạn có thể định nghĩa đường viền cho hầu hết các phần tử HTML.

```

p {
    border: 2px solid powderblue;
}

```

- CSS Padding

Thuộc tính padding của CSS xác định khoảng đệm (khoảng cách) giữa văn bản và đường viền.

```

p {
    border: 2px solid powderblue;
    padding: 30px;
}

```

- CSS Margin

Thuộc tính margin của CSS xác định khoảng cách (lè) bên ngoài đường viền.

```

p {
    border: 2px solid powderblue;
    margin: 50px;
}

```

- Link to External CSS

Có thể tham chiếu đến các bảng định kiểu bên ngoài bằng URL đầy đủ hoặc bằng đường dẫn tương đối đến trang web hiện tại.

Ví dụ này sử dụng URL đầy đủ để liên kết đến một bảng định dạng:

```
<link rel="stylesheet" href="https://www.w3schools.com/html/styles.css">
```

Ví dụ này liên kết đến một bảng định dạng nằm trong thư mục html trên trang web hiện tại:

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

Ví dụ này liên kết đến một bảng định dạng nằm trong cùng thư mục với trang hiện tại:

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

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <h1 style="color:red;">A Red Heading</h1> <p style="color:green;">A green paragraph.</p> </body> </html></pre>	<p>A Red Heading</p> <p>A green paragraph.</p>

8. Thẻ HTML Links

a. Chức năng

HTML Links là siêu liên kết.

Bạn có thể nhấp vào một liên kết và chuyển sang tài liệu khác.

Khi bạn di chuyển chuột qua một liên kết, mũi tên chuột sẽ biến thành một bàn tay nhỏ.

b. Cú pháp

Thẻ HTML L `<a>` xác định một siêu liên kết. Nó có cú pháp sau:

```
<a href="url">Link text</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 được.

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

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <h1>HTML Links</h1> <p>Visit W3Schools.com!</p> </body> </html></pre>	<p>HTML Links</p> <p>Visit W3Schools.com!</p>

9. Thẻ HTML Images

a. Chức năng: Hình ảnh có thể cải thiện thiết kế và giao diện của trang web.

b. Cú pháp:

```

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre></pre>	

10. Thẻ HTML Tables**a. Chức năng**

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

b. Cú pháp

- Một bảng trong HTML bao gồm các ô trong bảng bên trong các hàng và cột.
- Table Cells
 - Mỗi Table Cells được định nghĩa bằng thẻ `<td>` và thẻ `</td>`.
 - td là viết tắt của dữ liệu bảng.
 - Mọi thứ giữa `<td>` và `</td>` là nội dung của Table Cells.

```
<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
    <td>Linus</td>
  </tr>
</table>
```

- Lưu ý: Một Table Cells có thể chứa đủ loại phần tử HTML: văn bản, hình ảnh, danh sách, liên kết, bảng khác, v.v.
- Table Rows
 - Mỗi hàng của bảng bắt đầu bằng thẻ `<tr>` và kết thúc bằng thẻ `</tr>`.
 - tr là viết tắt của hàng của bảng.

```
<table>
  <tr>
    <td>Emil</td>
    <td>Tobias</td>
```

```

<td>Linus</td>
</tr>
<tr>
  <td>16</td>
  <td>14</td>
  <td>10</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 là như nhau.
- Lưu ý: Có những lúc một hàng có thể có ít hoặc nhiều ô hơn hàng khác. Bạn sẽ tìm hiểu về điều đó trong chương sau.
- Table Headers
 - 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>`:
 - `th` là viết tắt của tiêu đề bảng.

```



```

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

c. Ví dụ minh họa

Code mẫu	Kết quả						
<pre data-bbox="257 143 551 1957"><!DOCTYPE html> <html> <html> <style> table, th, td { border:1px solid black; } </style> <body> <h2>A basic HTML table</h2> <table style="width :100%"> <tr> <th>Khóa</th> <th>Lớp</th> </tr> <tr> <td>K22</td> <td>22161V TVM1</td> </tr> <tr> <td>K22</td> <td>22161V TVM2</td> </tr> </table> <p>To understand the example better, we have added</pre>	<p data-bbox="649 143 948 175">A basic HTML table</p> <table border="1" data-bbox="559 175 1504 382"> <thead> <tr> <th data-bbox="559 175 1127 249">Khóa</th><th data-bbox="1127 175 1504 249">Lớp</th></tr> </thead> <tbody> <tr> <td data-bbox="559 249 1127 323">K22</td><td data-bbox="1127 249 1504 323">22161VTVM1</td></tr> <tr> <td data-bbox="559 323 1127 382">K22</td><td data-bbox="1127 323 1504 382">22161VTVM2</td></tr> </tbody> </table> <p data-bbox="649 392 1475 466">To understand the example better, we have added borders to the table.</p>	Khóa	Lớp	K22	22161VTVM1	K22	22161VTVM2
Khóa	Lớp						
K22	22161VTVM1						
K22	22161VTVM2						

<pre>borders to the table.</p> </body> </html></pre>	
---	--

11. Thẻ HTML Lists

a. Chức năng

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

b. Cú pháp

- Unordered HTML List

Danh sách không có thứ tự bắt đầu bằng thẻ ``. Mỗi mục danh sách bắt đầu bằng thẻ ``.

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

```
<!DOCTYPE html>
<html>
<body>

<h2>An unordered HTML list</h2>

<ul>
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ul>

</body>
</html>
```

- Ordered HTML List

Danh sách có thứ tự bắt đầu bằng thẻ ``. Mỗi mục danh sách bắt đầu bằng thẻ ``.

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

```
<!DOCTYPE html>
<html>
<body>

<h2>An ordered HTML list</h2>

<ol>
<li>Coffee</li>
<li>Tea</li>
<li>Milk</li>
</ol>

</body>
</html>
```

- HTML Description Lists

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:

```
<!DOCTYPE html>
<html>
<body>

<h2>A Description List</h2>

<dl>
<dt>Coffee</dt>
<dd>- black hot drink</dd>
<dt>Milk</dt>
<dd>- white cold drink</dd>
</dl>

</body>
</html>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<!DOCTYPE html> <html> <body> <h2>An Unordered HTML List</h2> Coffee Tea Milk <h2>An Ordered HTML List</h2> Coffee Tea Milk </body> </html>	An Unordered HTML List <ul style="list-style-type: none"> • Coffee • Tea • Milk An Ordered HTML List <ol style="list-style-type: none"> 1. Coffee 2. Tea 3. Milk

12. Thẻ HTML Block and Inline

a. Chức năng: Mỗi 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 block và inline.

b. Cú pháp:

* block-level element

```
<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>
```

* inline element:

```
<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>
```

c. Ví dụ minh họa

* block-level element:

Code mẫu	Kết quả
<p>Hello World</p>	Hello World

* inline element:

Code mẫu	Kết quả
Hello World	Hello World

13. Thẻ HTML Iframes

a. Chức năng

Iframe HTML được sử dụng để hiển thị một trang web trong một trang web khác.

b. Cú pháp

- Thẻ <iframe> HTML chỉ định một khung nội tuyến.

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

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

- Mẹo: Thực hành tốt nhất là luôn bao gồm thuộc tính tiêu đề cho <iframe>. Thuộc tính này được trình đọc màn hình sử dụng để đọc nội dung của iframe.

Iframe - Đặt Chiều cao và Chiều rộng

- Sử dụng các thuộc tính chiều cao và chiều rộng để chỉ định kích thước của iframe.
- Chiều cao và chiều rộng được chỉ định theo pixel theo mặc định:

```
<iframe src="demo_iframe.htm" height="200" width="300" title="Iframe Example"></iframe>
```

- Hoặc bạn có thể thêm thuộc tính style và sử dụng các thuộc tính height và width của CSS:

```
<iframe src="demo_iframe.htm" style="height:200px; width:300px;" title="Iframe Example"></iframe>
```

Iframe - Xóa đường viền

- Theo mặc định, iframe có đường viền xung quanh.

- Để xóa đường viền, hãy thêm thuộc tính style và sử dụng thuộc tính border của CSS:

```
<iframe src="demo_iframe.htm" style="border:none;" title="Iframe Example"></iframe>
```

- Với CSS, bạn cũng có thể thay đổi kích thước, kiểu dáng và màu sắc của đường viền iframe:

```
<iframe src="demo_iframe.htm" style="border:2px solid red;" title="Iframe Example"></iframe>
```

- Iframe - Mục tiêu cho một liên kết

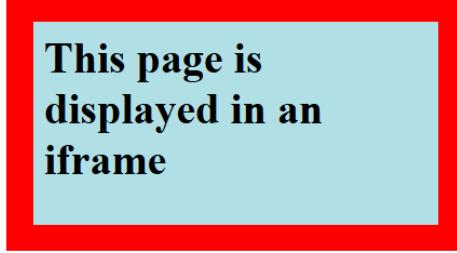
- Iframe có thể được sử dụng làm khung mục tiêu cho một liên kết.

- Thuộc tính mục tiêu của liên kết phải tham chiếu đến thuộc tính tên của iframe:

```
<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>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <h2>Custom Iframe Border</h2> <p>With CSS, you can also change the size, style and color of the iframe's border:</p> <iframe src="demo_iframe.htm" style="border:20px solid red;" title="Iframe Example"></iframe> </body> </html></pre>	<p>Custom Iframe Border</p> <p>With CSS, you can also change the size, style and color of the iframe's border:</p> 

14. Thủ thuật Javascripts

a. Chức năng

JavaScript làm cho các trang HTML năng động và tương tác hơn.

b. Cú pháp

- Thủ thuật <script> Tag

Thủ thuật HTML <script> được sử dụng để xác định tập lệnh phía máy khách (JavaScript).

Phần tử <script> chứa các câu lệnh script hoặc nó trỏ đến tệp script bên ngoài thông qua thuộc tính src.

Các cách sử dụng phổ biến của JavaScript là thao tác hình ảnh, xác thực biểu mẫu và thay đổi nội dung

động.

Để chọn một phần tử HTML, JavaScript thường sử dụng phương thức `document.getElementById()`.

Ví dụ JavaScript này viết "Xin chào JavaScript!" thành một phần tử HTML có id="demo":

```
<!DOCTYPE html>
<html>
<body>

<h2>Use JavaScript to Change Text</h2>
<p>This example writes "Hello JavaScript!" into an HTML element with id="demo":</p>

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

<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>

</body>
</html>
```

- The HTML <noscript> Tag

Thẻ HTML <noscript> xác định nội dung thay thế sẽ được hiển thị cho người dùng đã tắt tập lệnh trong trình duyệt của họ hoặc có trình duyệt không hỗ trợ tập lệnh:

```
<!DOCTYPE html>
<html>
<body>

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

<script>
document.getElementById("demo").innerHTML = "Hello JavaScript!";
</script>

<noscript>Sorry, your browser does not support JavaScript!</noscript>

<p>A browser without support for JavaScript will show the text written inside the noscript element.</p>

</body>
</html>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<!DOCTYPE html> <html> <body> <h1>My First JavaScript</h1> <p>Here, a JavaScript changes the value of the src (source) attribute of an image.</p> <script>	

```

function light(sw) {
  var pic;
  if (sw == 0) {
    pic = "pic_bulboff.gif"
  } else {
    pic = "pic_bulbon.gif"
  }
  document.getElementById('myImage').src = pic;
}



<p>
<button type="button" onclick="light(1)">Light
On</button>
<button type="button" onclick="light(0)">Light
Off</button>
</p>

</body>
</html>

```

My First JavaScript

Here, a JavaScript changes the value of the src (source) attribute of an image.



[Light On](#) [Light Off](#)

My First JavaScript

Here, a JavaScript changes the value of the src (source) attribute of an image.



[Light On](#) [Light Off](#)

15. Thẻ HTML Head

a. Chức năng:

Thẻ <head> chứa thông tin meta và tài nguyên giúp trình duyệt hiểu & hiển thị trang web đúng cách.

b. Cú pháp:

```
<head>,<title>,<style>,<link>,<meta>,<script>,<base>.
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre> <!DOCTYPE html> <html> <head> <title>Page Title</title> <link rel="stylesheet" href="mystyle.css"> </head> <body> <h1>This is a Heading</h1> <p>This is a paragraph.</p> </body> </pre>	<p>This is a Heading</p> <p>This is a paragraph.</p>

16. Thẻ HTML Layout

a. Chức năng

Các trang web thường hiển thị nội dung theo nhiều cột (giống như tạp chí hoặc báo).

b. Cú pháp

- HTML Layout Elements

- HTML có một số thành phần ngữ nghĩa xác định các phần khác nhau của một trang web:

- `<header>` - Defines a header for a document or a section
- `<nav>` - Defines a set of navigation links
- `<section>` - Defines a section in a document
- `<article>` - Defines independent, self-contained content
- `<aside>` - Defines content aside from the content (like a sidebar)
- `<footer>` - Defines a footer for a document or a section
- `<details>` - Defines additional details that the user can open and close on demand
- `<summary>` - Defines a heading for the `<details>` element

- HTML Layout Techniques

- Có bốn kỹ thuật khác nhau để tạo bố cục nhiều cột. Mỗi kỹ thuật đều có ưu và nhược điểm riêng:

- CSS framework
- CSS float property
- CSS flexbox
- CSS grid

- CSS Frameworks:

Nếu bạn muốn tạo bố cục nhanh chóng, bạn có thể sử dụng một khung CSS như W3.CSS hoặc Bootstrap.

- CSS Float Layout

Thường thì người ta sẽ tạo toàn bộ bố cục web bằng thuộc tính float của CSS. Float rất dễ học - bạn chỉ cần nhớ cách hoạt động của thuộc tính float và clear. Nhược điểm: Các phần tử float bị ràng buộc với luồng tài liệu, điều này có thể gây hại cho tính linh hoạt.

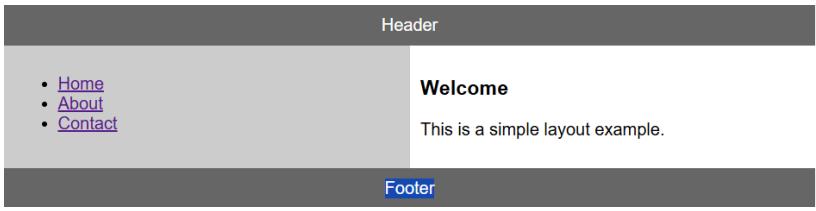
- CSS Flexbox Layout

Việc sử dụng flexbox đảm bảo các thành phần hoạt động theo cách có thể dự đoán được khi bố cục trang phải phù hợp với các kích thước màn hình và thiết bị hiển thị khác nhau.

- CSS Grid Layout

Mô-đun Bố cục Lưới CSS cung cấp hệ thống bố cục dạng lưới, với các hàng và cột, giúp thiết kế trang web dễ dàng hơn mà không cần sử dụng các ô nội và định vị.

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html lang="en"> <head> <meta charset="UTF-8"> <meta name="viewport" content="width=device-width, initial-scale=1.0"> <title>Simple Layout</title> <style> body {font-family: Arial, sans-serif; margin: 0;} header, footer {background: #666; color: white; text-align: center; padding: 10px;} nav {background: #ccc; padding: 10px;} article {padding: 10px;} section {display: flex;} nav, article {flex: 1;} </style> </head> <body> <header>Header</header> <section> <nav> Home About Contact > </nav> <article> <h1>Welcome</h1> This is a simple layout example. </article> </section> <footer>Footer</footer></pre>	

```

<p>This is a simple
layout example.</p>
</article>
</section>
<footer>Footer</footer>
</body>
</html>

```

17. Thẻ HTML Symbols

a. Chức năng

Các ký hiệu hoặc chữ cái không có trên bàn phím của bạn có thể được thêm vào HTML bằng cách sử dụng các thực thể.

b. Cú pháp

Các thực thể HTML đã được mô tả ở chương trước.

Nhiều ký hiệu toán học, kỹ thuật và tiền tệ không có trên bàn phím thông thường.

Để thêm các ký hiệu như vậy vào trang HTML, bạn có thể sử dụng tên thực thể hoặc số thực thể (tham chiếu thập phân hoặc thập lục phân) cho ký hiệu:

```

<!DOCTYPE html>
<html>
<body>

<p>I will display &euro;</p>
<p>I will display &#8364;</p>
<p>I will display &#x20AC;</p>

</body>
</html>

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<!DOCTYPE html> <html> <body>	Hiển thị €
<p>Hiển thị €</p> <p>Hiển thị ∇</p> <p>Hiển thị ∆</p>	Hiển thị ∇
</body> </html>	Hiển thị Δ

18. Thẻ HTML Emojis

a. Chức năng: Emojis are characters from the UTF-8 character set: 😊 🌎 🚨

b. Cú pháp:

```

<p>(mã UTF-8);</p>

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<h2>🗿</h2>	

19. Thủ HTML URL Encode

a. Chức năng

- URL là một từ khác để chỉ địa chỉ web.
- URL có thể bao gồm các từ (ví dụ: w3schools.com) hoặc địa chỉ Giao thức Internet (IP) (ví dụ: 192.68.20.50).
- Hầu hết mọi người nhập tên khi lướt web, vì tên dễ nhớ hơn số.

b. Cú pháp

- URL - Uniform Resource Locator

- Trình duyệt web yêu cầu các trang từ máy chủ web bằng cách sử dụng URL.
- Uniform Resource Locator (URL) được sử dụng để định địa chỉ một tài liệu (hoặc dữ liệu khác) trên web.
- Một địa chỉ web như https://www.w3schools.com/html/default.asp tuân theo các quy tắc cú pháp sau:

scheme://prefix.domain:port/path/filename

- Giải thích:

- scheme - định nghĩa loại dịch vụ Internet (phổ biến nhất là http hoặc https)
- prefix - định nghĩa tiền tố tên miền (mặc định cho http là www)
- domain - định nghĩa tên miền Internet (như w3schools.com)
- port - định nghĩa số cổng tại máy chủ (mặc định cho http là 80)
- path - định nghĩa đường dẫn tại máy chủ (Nếu bỏ qua: thư mục gốc của trang web)
- filename - định nghĩa tên của tài liệu hoặc tài nguyên

- Các lược đồ URL phổ biến

- Bảng dưới đây liệt kê một số lược đồ phổ biến:

Sơ đồ	Viết tắt của	Được dùng cho
http	HyperText Transfer Protocol	Các trang web phổ biến. Không được mã hóa
https	Secure HyperText Transfer Protocol	Các trang web an toàn. Được mã hóa
ftp	File Transfer Protocol	Tải xuống hoặc tải lên các tệp
file		Một tệp trên máy tính của bạn

- Mã hóa URL

- URL chỉ có thể được gửi qua Internet bằng bộ ký tự ASCII. Nếu URL chứa các ký tự nằm ngoài bộ ASCII, URL phải được chuyển đổi.

- Mã hóa URL chuyển đổi các ký tự không phải ASCII thành định dạng có thể truyền qua Internet.
- Mã hóa URL thay thế các ký tự không phải ASCII bằng "%" theo sau là các chữ số thập lục phân.
- URL không thể chứa khoảng trắng. Mã hóa URL thường thay thế khoảng trắng bằng dấu cộng (+) hoặc %20.

- Ví dụ về mã hóa ASCII

- Trình duyệt của bạn sẽ mã hóa dữ liệu đầu vào, theo bộ ký tự được sử dụng trong trang của bạn.
- Bộ ký tự mặc định trong HTML5 là UTF-8.

Character	From Windows-1252	From UTF-8
€	%80	%E2%82%AC
£	%A3	%C2%A3
(©)	%A9	%C2%A9
(®)	%AE	%C2%AE
À	%C0	%C3%80
Á	%C1	%C3%81
Â	%C2	%C3%82
Ã	%C3	%C3%83
Ä	%C4	%C3%84
Å	%C5	%C3%85

c. Ví dụ minh họa

Code mẫu	Kết quả
text=Hello+Khang+%C3%80	Hello Khang À

20. Thẻ HTML Froms

a. Chức năng

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.

b. Cú pháp

Phần tử HTML <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.

Tất cả các thành phần biểu mẫu khác nhau đều được đề cập trong chương này: Thành phần biểu mẫu HTML.

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <h2>HTML Forms</h2> <form action="/action_page.php"> <label for="fname">First name:</label>
 <input type="text" id="fname" name="fname" value="John">
 <label for="lname">Last name:</label>
 <input type="text" id="lname" name="lname" value="Doe">

 <input type="submit" value="Submit"> </form> <p>If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".</p> </body> </html></pre>	<p>HTML Forms</p> <p>First name: <input type="text" value="John"/></p> <p>Last name: <input type="text" value="Doe"/></p> <p><input type="button" value="Submit"/></p> <p>If you click the "Submit" button, the form-data will be sent to a page called "/action_page.php".</p> <p>Submitted Form Data</p> <p>Your input was received as:</p> <p><input type="text" value="fname=John&lname=Doe"/></p> <p>The server has processed your input and returned this answer.</p> <p>Note: This tutorial will not teach you how servers are processing input. Processing input is explained in our PHP tutorial.</p>

21. Thẻ HTML Graphics

* Canvas Graphics

a. Chức năng:

Phần tử HTML <canvas> được sử dụng để vẽ đồ họa, ngay lập tức, thông qua JavaScript.

Phần tử <canvas> chỉ là một vùng chứa đồ họa. Bạn phải sử dụng JavaScript để thực sự vẽ đồ họa.

Canvas có một số phương pháp để vẽ đường dẫn, hộp, hình tròn, văn bản và thêm hình ảnh

Canvas được hỗ trợ bởi tất cả các trình duyệt chính.

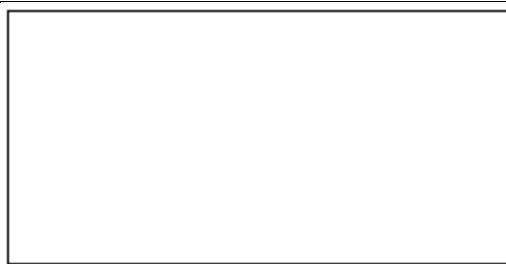
b. Cú pháp:

```
<canvas>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
----------	---------

```
<canvas id="myCanvas" width="200"
height="100" style="border:1px solid
#000000;">
</canvas>
```

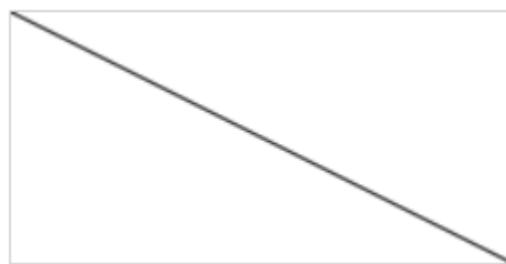


```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200"
height="100" style="border:1px solid
#d3d3d3;"></canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.moveTo(0,0);
ctx.lineTo(200,100);
ctx.stroke();
</script>

</body>
</html>
```

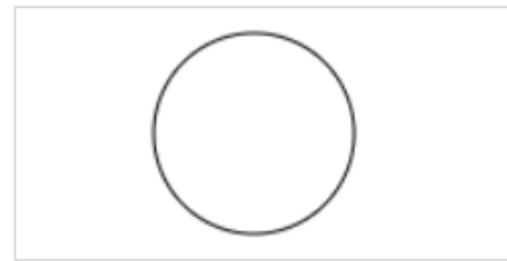


```
<!DOCTYPE html>
<html>
<body>

<canvas id="myCanvas" width="200"
height="100" style="border:1px solid #d3d3d3;">
</canvas>

<script>
var c = document.getElementById("myCanvas");
var ctx = c.getContext("2d");
ctx.beginPath();
ctx.arc(95,50,40,0,2*Math.PI);
ctx.stroke();
</script>

</body>
</html>
```



* SVG Graphics

a. Chức năng:

SVG là viết tắt của Scalable Vector Graphics

SVG được sử dụng để định nghĩa đồ họa dựa trên vector cho Web

SVG định nghĩa đồ họa ở định dạng XML

Mỗi phần tử và thuộc tính trong tệp SVG có thể được hoạt hình hóa

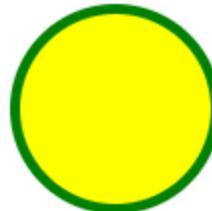
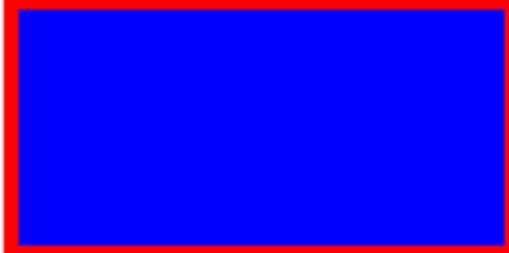
SVG là khuyến nghị của W3C

SVG tích hợp với các tiêu chuẩn khác, chẳng hạn như CSS, DOM, XSL và JavaScript

b. Cú pháp:

```
<svg>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <svg width="100" height="100"> <circle cx="50" cy="50" r="40" stroke="green" stroke-width="4" fill="yellow" /> Sorry, your browser does not support inline SVG. </svg> </body> </html></pre>	
<pre><!DOCTYPE html> <html> <body> <svg width="400" height="120"> <rect x="10" y="10" width="200" height="100" stroke="red" stroke-width="6" fill="blue" /> Sorry, your browser does not support inline SVG. </svg> </body> </html></pre>	

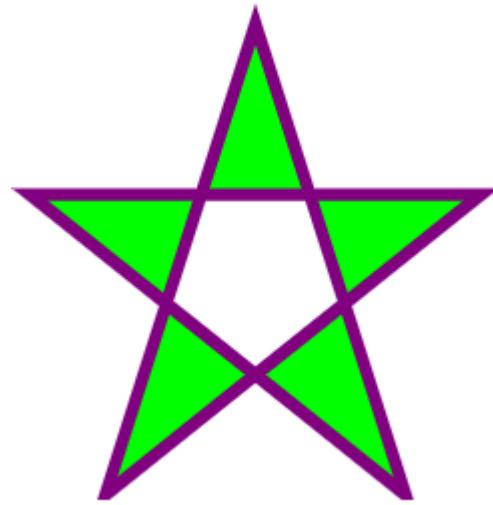
```

<!DOCTYPE html>
<html>
<body>

<svg width="300" height="200">
  <polygon points="100,10 40,198 190,78 10,78
160,198"
    style="fill:lime;stroke:purple;stroke-width:5;fill-
rule:evenodd;" />
  Sorry, your browser does not support inline SVG.
</svg>

</body>
</html>

```



22. Thẻ HTML Video

a. Chức năng

Phần tử HTML `<video>` được sử dụng để hiển thị video trên trang web.

b. Cú pháp

- Phần tử `<video>` HTML: Để hiển thị video trong HTML, hãy sử dụng phần tử `<video>`:

```

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

```

- Cách thức hoạt động

- Thuộc tính `controls` thêm các điều khiển video, như phát, tạm dừng và âm lượng.
- Bạn nên luôn bao gồm các thuộc tính `width` và `height`. Nếu `height` và `width` không được thiết lập, trang có thể nhấp nháy trong khi video tải.
- Phần tử `<source>` cho phép bạn chỉ định các tệp video thay thế mà trình duyệt có thể chọn. Trình duyệt sẽ sử dụng định dạng được nhận dạng đầu tiên.
- Văn bản giữa các thẻ `<video>` và `</video>` sẽ chỉ được hiển thị trong các trình duyệt không hỗ trợ phần tử `<video>`.

- HTML `<video>` Tự động phát

- Để tự động bắt đầu một video, hãy sử dụng thuộc tính `autoplay`:

```

<video width="320" height="240" autoplay>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>

```

- Lưu ý: Trình duyệt Chromium không cho phép phát tự động trong hầu hết các trường hợp. Tuy nhiên, phát tự động tắt tiếng luôn được phép.

- Thêm tắt tiếng sau khi tự động phát để cho phép video của bạn tự động phát (nhưng bị tắt tiếng):

```

<video width="320" height="240" autoplay muted>
  <source src="movie.mp4" type="video/mp4">
  <source src="movie.ogg" type="video/ogg">
Your browser does not support the video tag.
</video>

```

- Định dạng video HTML

Có ba định dạng video được hỗ trợ: MP4, WebM và Ogg. Trình duyệt hỗ trợ các định dạng khác nhau là:

Browser	MP4	WebM	Ogg
Edge	YES	YES	YES
Chrome	YES	YES	YES
Firefox	YES	YES	YES
Safari	YES	YES	NO
Opera	YES	YES	YES

- Video HTML - Các loại phương tiện

File Format	Media Type
MP4	video/mp4
WebM	video/webm
Ogg	video/ogg

- Video HTML - Phương pháp, Thuộc tính và Sự kiện

- HTML DOM định nghĩa phương pháp, thuộc tính và sự kiện cho phần tử `<video>`.
- Điều này cho phép bạn tải, phát và tạm dừng video, cũng như thiết lập thời lượng và âm lượng.
- Ngoài ra còn có các sự kiện DOM có thể thông báo cho bạn khi video bắt đầu phát, tạm dừng, v.v.

- Thể video HTML

Thể	Mô tả
<code><video></code>	Định nghĩa video hoặc phim
<code><source></code>	Định nghĩa nhiều tài nguyên phương tiện cho các thành phần phương tiện, chẳng hạn như <code><video></code> và <code><audio></code>
<code><track></code>	Định nghĩa các bản nhạc văn bản trong trình phát phương tiện

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre> <!DOCTYPE html> <html> <body> <div style="text-align:center"> <button onclick="playPause()">Play/Pause</button>
 <button onclick="makeBig()">Big</button> <button onclick="makeSmall()">Small</button> </pre>	

```

<button
  onclick="makeNormal()">Normal</button>
<br><br>
<video id="video1" width="420">
  <source          src="mov_bbb.mp4"
  type="video/mp4">
  <source          src="mov_bbb.ogv"
  type="video/ogg">
Your browser does not support HTML
video.
</video>
</div>

<script>
var           myVideo      =
document.getElementById("video1");

function playPause() {
  if (myVideo.paused)
    myVideo.play();
  else
    myVideo.pause();
}

function makeBig() {
  myVideo.width = 560;
}

function makeSmall() {
  myVideo.width = 320;
}

function makeNormal() {
  myVideo.width = 420;
}
</script>

<p>Video courtesy of <a href="https://www.bigbuckbunny.org/" target="_blank">Big Buck Bunny</a>.</p>

</body>
</html>

```



Video courtesy of [Big Buck Bunny](https://www.bigbuckbunny.org/).

[Play/Pause](#) [Big](#) [Small](#) [Normal](#)

23. Thủ số HTML ID

a. Chức năng

Thuộc tính HTML Id được sử dụng để chỉ định một id duy nhất cho một phần tử HTML.

Bạn không thể có nhiều phần tử có cùng id trong tài liệu HTML.

b. Cú pháp

Thuộc tính id chỉ định một id duy nhất cho một phần tử HTML. Giá trị của thuộc tính id phải là duy nhất trong tài liệu HTML.

Thuộc tính id được sử dụng để trỏ đến một khai báo kiểu cụ thể trong biểu định kiểu. Nó cũng được JavaScript sử dụng để truy cập và thao tác phần tử với id cụ thể.

Cú pháp của id là: viết ký tự băm (#), theo sau là tên id. Sau đó, xác định thuộc tính CSS trong dấu ngoặc nhọn {}.

Trong ví dụ sau, chúng ta có phần tử `<h1>` trỏ đến tên id "myHeader". Phần tử `<h1>` này sẽ được tạo kiểu theo định nghĩa kiểu #myHeader trong phần đầu:

```
<!DOCTYPE html>
<html>
<head>
<style>
#myHeader {
    background-color: lightblue;
    color: black;
    padding: 40px;
    text-align: center;
}
</style>
</head>
<body>

<h2>The id Attribute</h2>
<p>Use CSS to style an element with the id "myHeader":</p>

<h1 id="myHeader">My Header</h1>

</body>
</html>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <head> <style> /* Style the element with the id "myHeader" */ #myHeader { background-color: lightblue; color: black; padding: 40px; text-align: center; } /* Style all elements with the class name "city" */ .city { background-color: tomato; }</pre>	<p>Difference Between Class and ID</p> <p>A class name can be used by multiple HTML elements, while an id name must only be used by one HTML element within the page:</p> <p>The screenshot shows a web page with a header "My Cities". Below it are three items: "London", "Paris", and "Tokyo". The "London" and "Paris" items have a blue background and white text, while the "Tokyo" item has a red background. This illustrates that the ID "myHeader" is applied to the first <code><h1></code> element, while the class ".city" is applied to all <code><div></code> elements with that class.</p>

```

color: white;
padding: 10px;
}
</style>
</head>
<body>

<h2>Difference Between Class and ID</h2>
<p>A class name can be used by multiple HTML
elements, while an id name must only be used by
one HTML element within the page:</p>

<!-- An element with a unique id -->
<h1 id="myHeader">My Cities</h1>

<!-- Multiple elements with same class -->
<h2 class="city">London</h2>
<p>London is the capital of England.</p>

<h2 class="city">Paris</h2>
<p>Paris is the capital of France.</p>

<h2 class="city">Tokyo</h2>
<p>Tokyo is the capital of Japan.</p>

</body>
</html>

```

24. Thủ HTML Plug-ins

a. Chức năng: Các plug-in được thiết kế để sử dụng cho nhiều mục đích khác nhau:

Để chạy các applet Java

Để chạy các điều khiển Microsoft ActiveX

Để hiển thị phim Flash

Để hiển thị bản đồ

Để quét vi-rút

Để xác minh ID ngân hàng

b. Cú pháp:

<object>, <embed>

c. Ví dụ minh họa

Code mẫu	Kết quả
----------	---------

```

<!DOCTYPE html>
<html>
<body>
<object width="100%" height="500px" data="snippet.html"></object>
</body>
</html>

```



<pre> <!DOCTYPE html> <html> <body> <embed width="90%" height="500px" src="snippet.html"> </body> </html> </pre>	Alfreds Futterkiste Berglunds snabbköp Centro comercial Moctezuma Ernst Handel FISSA Fabrica Inter. Salchichas S.A. Galería del gastrónomo Island Trading Königlich Essen Laughing Bacchus Wine Cellars Magazzini Alimentari Riuniti North/South Paris spécialités Rattlesnake Canyon Grocery Simons bistro The Big Cheese Vaffeljernet Wolski Zajazd	Berlin Luleå México D.F. Graz Madrid Barcelona Cowes Brandenburg Vancouver Bergamo London Paris Albuquerque København Portland Århus Warszawa	Germany Sweden Mexico Austria Spain Spain UK Germany Canada Italy UK France USA Denmark USA Denmark Poland
--	---	---	--

25. Thủ HTML Youtube

a. Chức năng

- Cách dễ nhất để phát video ở định dạng HTML là sử dụng YouTube.
- ID video YouTube
- YouTube sẽ hiển thị một ID (như tgbNymZ7vqY) khi bạn lưu (hoặc phát) video.
- Bạn có thể sử dụng ID này và tham chiếu đến video của mình trong mã HTML.

b. Cú pháp

- Phát video YouTube trong HTML

- Để phát video của bạn trên trang web, hãy thực hiện như sau:

- Tải video lên YouTube
- Ghi lại id video

- Xác định phần tử <iframe> trong trang web của bạn
- Để thuộc tính src trả đến URL video
- Sử dụng thuộc tính width và height để chỉ định kích thước của trình phát
- Thêm bất kỳ tham số nào khác vào URL (xem bên dưới)

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY">
</iframe>
```

- Tự động phát YouTube + Tắt tiếng

- Bạn có thể để video của mình tự động phát khi người dùng truy cập trang bằng cách thêm autoplay=1 vào URL YouTube. Tuy nhiên, việc tự động phát video sẽ gây khó chịu cho khách truy cập!
- Lưu ý: Trình duyệt Chromium không cho phép tự động phát trong hầu hết các trường hợp. Tuy nhiên, tự động phát tắt tiếng luôn được phép.
- Thêm mute=1 sau đó autoplay=1 để video của bạn tự động phát (nhưng tắt tiếng).

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?autoplay=1&mute=1">
</iframe>
```

- Danh sách phát YouTube

- Danh sách các video được phân cách bằng dấu phẩy để phát (ngoài URL gốc).

- YouTube Loop

- Thêm playlist=videoID và loop=1 để video của bạn lặp lại mãi mãi.
- loop=0 (mặc định) - Video sẽ chỉ phát một lần.
- loop=1 - Video sẽ lặp lại (mãi mãi).

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?playlist=tgbNymZ7vqY&loop
=1">
</iframe>
```

- YouTube Controls

- Thêm controls=0 để KHÔNG hiển thị các điều khiển trong trình phát video.
- controls=0 - Các điều khiển của trình phát không hiển thị.
- controls=1 (mặc định) - Các điều khiển của trình phát được hiển thị.

```
<iframe width="420" height="315"
src="https://www.youtube.com/embed/tgbNymZ7vqY?controls=0">
</iframe>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <iframe width="500" height="400" src="https://www.youtube.com/embed/tgbNymZ7vqY" "> </iframe> </body> </html></pre>	

26. Thẻ HTML Geolocation

a. Chức năng

HTML Geolocation API được sử dụng để xác định vị trí của người dùng.

Vì điều này có thể ảnh hưởng đến quyền riêng tư nên vị trí này sẽ không có sẵn trừ khi người dùng chấp thuận.

b. Cú pháp

Phương thức getCurrentPosition() được sử dụng để trả về vị trí của người dùng.

Ví dụ bên dưới trả về vĩ độ và kinh độ vị trí của người dùng:

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <h1>HTML Geolocation</h1> <p>Click the button to get your coordinates.</p> <button onclick="getLocation()">Try It</button> <p id="demo"></p> <script> const x = document.getElementById("demo"); function getLocation() { if (navigator.geolocation) { navigator.geolocation.getCurrentPosition(showPosition); } else { x.innerHTML = "Geolocation is not supported by this browser."; } } function showPosition(position) {</pre>	

```

x.innerHTML = "Latitude: " + position.coords.latitude +
"<br>Longitude: " + position.coords.longitude;
}
</script>

</body>
</html>

```

Tham số thứ hai của phương thức getCurrentPosition() được sử dụng để xử lý lỗi. Nó chỉ định một chức năng sẽ chạy nếu không lấy được vị trí của người dùng:

```

<!DOCTYPE html>
<html>
<body>
<h1>HTML Geolocation</h1>
<p>Click the button to get your coordinates.</p>

<button onclick="getLocation()">Try It</button>

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

<script>
const x = document.getElementById("demo");

function getLocation() {
  if (navigator.geolocation) {
    navigator.geolocation.getCurrentPosition(showPosition, showError);
  } else {
    x.innerHTML = "Geolocation is not supported by this browser.";
  }
}

function showPosition(position) {
  x.innerHTML = "Latitude: " + position.coords.latitude +
  "<br>Longitude: " + position.coords.longitude;
}

function showError(error) {
  switch(error.code) {
    case error.PERMISSION_DENIED:
      x.innerHTML = "User denied the request for Geolocation."
      break;
    case error.POSITION_UNAVAILABLE:
      x.innerHTML = "Location information is unavailable."
      break;
    case error.TIMEOUT:
      x.innerHTML = "The request to get user location timed out."
      break;
    case error.UNKNOWN_ERROR:
      x.innerHTML = "An unknown error occurred."
      break;
  }
}
</script>

```

```
</body>  
</html>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <head> <title>Geolocation API Example</title> </head> <body> <h2>HTML5 Geolocation API</h2> <p>Nhấn vào nút bên dưới để lấy vị trí của bạn:</p> <button onclick="getLocation()">Lấy vị trí</button> <p id="location"></p> <script> function getLocation() { if (navigator.geolocation) { navigator.geolocation.getCurrentPosition(showPosition, showError); } else { document.getElementById("location").innerHTML = "Trình duyệt của bạn không hỗ trợ Geolocation." } function showPosition(position) { document.getElementById("location").innerHTML = "Vĩ độ: " + position.coords.latitude + "
Kinh độ: " + position.coords.longitude; } function showError(error) { switch(error.code) { case error.PERMISSION_DENIED: document.getElementById("location").innerHTML = "Người dùng từ chối quyền truy cập vị trí."; break; case error.POSITION_UNAVAILABLE: document.getElementById("location").innerHTML = "Thông tin vị trí không có sẵn."; break; case error.TIMEOUT:</pre>	<p>HTML5 Geolocation API</p> <p>Nhấn vào nút bên dưới để lấy vị trí của bạn:</p> <p><input type="button" value="Lấy vị trí"/></p> <hr/> <p>HTML5 Geolocation API</p> <p>Nhấn vào nút bên dưới để lấy vị trí của bạn:</p> <p><input type="button" value="Lấy vị trí"/></p> <p>Người dùng từ chối quyền truy cập vị trí.</p>

```

document.getElementById("location").innerHTML =
"Yêu cầu lấy vị trí đã hết thời gian.";
    break;
  case error.UNKNOWN_ERROR:

document.getElementById("location").innerHTML = "Lỗi
không xác định.";
    break;
  }
}
</script>

</body>
</html>

```

27. Thẻ HTML Div

a. Chức năng:

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

b. Cú pháp

```
<div>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre> <!DOCTYPE html> <html> <style> div { background-color: #FFF4A3; } </style> <body> Lorem Ipsum <div>I am a div</div> dolor sit amet. </body> </html> </pre>	<p> Lorem Ipsum I am a div dolor sit amet.</p>
<pre> <!DOCTYPE html> <html> <style> div { background-color: #FFF4A3; } </style> <body> <div> <h2>London</h2> <p>London is the capital city of England.</p> <p>London has over 9 million inhabitants.</p> </pre>	<p>London</p> <p>London is the capital city of England.</p> <p>London has over 9 million inhabitants.</p>

```
</div>
</body>
</html>
```

28. Thủ HTML Computercode

a. Chức năng

HTML chứa một số thành phần để xác định dữ liệu đầu vào của người dùng và mã máy tính.

b. Cú pháp

- HTML <kbd> cho đầu vào bàn phím

- Phản tử HTML <kbd> được sử dụng để xác định đầu vào bàn phím. Nội dung bên trong được hiển thị bằng phông chữ đơn cách mặc định của trình duyệt.
- Xác định một số văn bản làm dữ liệu nhập từ bàn phím trong tài liệu:

<p>Save the document by pressing <kbd>Ctrl + S</kbd></p>
Save the document by pressing Ctrl + S

- HTML <samp> cho đầu ra chương trình

- Phản tử HTML <samp> được sử dụng để xác định đầu ra mẫu từ một chương trình máy tính. Nội dung bên trong được hiển thị bằng phông chữ đơn cách mặc định của trình duyệt.
- Xác định một số văn bản làm mẫu đầu ra từ một chương trình máy tính trong một tài liệu:

<p>Message from my computer:</p>
<p><samp>File not found. Press F1 to continue</samp></p>
Message from my computer:
File not found.
Press F1 to continue

- HTML <code> cho mã máy tính

- Phản tử HTML <code> được sử dụng để định nghĩa một đoạn mã máy tính. Nội dung bên trong được hiển thị bằng phông chữ đơn cách mặc định của trình duyệt.
- Define some text as computer code in a document:

<code>
x = 5;
y = 6;
z = x + y;
</code>

- Giữ nguyên ngắt dòng

- Lưu ý rằng phản tử <code> KHÔNG giữ nguyên khoảng trắng và ngắt dòng thừa.
- Để giữ nguyên khoảng trắng và ngắt dòng thừa, bạn có thể đặt phản tử <code> bên trong phản tử <pre>:

```

<pre>
<code>
x = 5;
y = 6;
z = x + y;
</code>
</pre>

```

- HTML <var> cho Biến

- Phần tử HTML <var> được sử dụng để định nghĩa một biến trong lập trình hoặc trong biểu thức toán học. Nội dung bên trong thường được hiển thị bằng chữ in nghiêng.

<p>The area of a triangle is: $1/2 \times \text{<var>b</var>} \times \text{<var>h</var>}$, where <var>b</var> is the base, and <var>h</var> is the vertical height.</p>

- Tóm tắt chương

- Phần tử <kbd> định nghĩa đầu vào bàn phím
- Phần tử <samp> định nghĩa đầu ra mẫu từ chương trình máy tính
- Phần tử <code> định nghĩa một đoạn mã máy tính
- Phần tử <var> định nghĩa một biến trong lập trình hoặc trong biểu thức toán học
- Phần tử <pre> định nghĩa văn bản được định dạng sẵn

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre> <!DOCTYPE html> <html> <body> <pre> <code> x = 1; y = 3; z = x + y; </code> </pre> </body> </html> </pre>	<pre> x = 1; y = 3; z = x + y; </pre>

29. Thủ HTML Semantics

a. Chức năng

Phần tử ngữ nghĩa mô tả rõ ràng ý nghĩa của nó đối với cả trình duyệt và nhà phát triển.

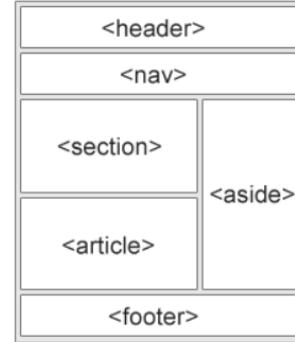
Ví dụ về các phần tử phi ngữ nghĩa: <div> và - Không cho biết gì về nội dung của nó.

Ví dụ về các phần tử ngữ nghĩa: , <table>, <article> - Xác định rõ ràng nội dung của nó.

Nhiều trang web chứa mã HTML như: <div id="nav"> <div class="header"> <div id="footer"> để biểu thị điều hướng, đầu trang và chân trang.

Trong HTML có một số thành phần ngữ nghĩa có thể được sử dụng để xác định các phần khác nhau của trang web:

- <article>
- <aside>
- <details>
- <figcaption>
- <figure>
- <footer>
- <header>
- <main>
- <mark>
- <nav>
- <section>
- <summary>
- <time>



b. Cú pháp

Phần tử <section> xác định một phần trong tài liệu.

Theo tài liệu HTML của W3C: "Phần là một nhóm nội dung theo chủ đề, thường có tiêu đề".

Ví dụ về nơi có thể sử dụng phần tử <section>:

chương

Giới thiệu

Mục tin tức

Thông tin liên hệ

Một trang web thường có thể được chia thành các phần để giới thiệu, nội dung và thông tin liên hệ.

```
<!DOCTYPE html>
<html>
<body>

<section>
<h1>WWF</h1>
<p>The World Wide Fund for Nature (WWF) is an international organization working on issues regarding the conservation, research and restoration of the environment, formerly named the World Wildlife Fund. WWF was founded in 1961.</p>
</section>

<section>
<h1>WWF's Panda symbol</h1>
<p>The Panda has become the symbol of WWF. The well-known panda logo of WWF originated from a panda named Chi Chi that was transferred from the Beijing Zoo to the London Zoo in the same year of the establishment of WWF.</p>
</section>
```

```
</body>  
</html>
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <h1>The article element</h1> <article> <h2>Google Chrome</h2> <p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today!</p> </article> <article> <h2>Mozilla Firefox</h2> <p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second most popular web browser since January, 2018.</p> </article> <article> <h2>Microsoft Edge</h2> <p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced Internet Explorer.</p> </article> </body> </html></pre>	<h2>The article element</h2> <h3>Google Chrome</h3> <p>Google Chrome is a web browser developed by Google, released in 2008. Chrome is the world's most popular web browser today!</p> <h3>Mozilla Firefox</h3> <p>Mozilla Firefox is an open-source web browser developed by Mozilla. Firefox has been the second most popular web browser since January, 2018.</p> <h3>Microsoft Edge</h3> <p>Microsoft Edge is a web browser developed by Microsoft, released in 2015. Microsoft Edge replaced Internet Explorer.</p>

30. Thẻ HTML Input Types

a. Chức năng:

Trong HTML, thẻ `<input>` được sử dụng để tạo các trường nhập liệu (input fields) trong biểu mẫu (form), cho phép người dùng nhập dữ liệu. Thuộc tính `type` của thẻ `<input>` quyết định loại trường nhập liệu sẽ được hiển thị và cách nó hoạt động

b. Cú pháp

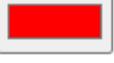
```
<input type="">
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre data-bbox="169 128 734 832"><!DOCTYPE html> <html> <body> <p>The input type="password" defines a password field:</p> <form action="/action_page.php"> <label for="username">Username:</label>
 <input type="text" id="username" name="username">
 <label for="pwd">Password:</label>
 <input type="password" id="pwd" name="pwd">

 <input type="submit" value="Submit"> </form> </body> </html></pre>	<p>Username:</p> <input id="username" name="username" type="text"/> <p>Password:</p> <input id="pwd" name="pwd" type="password"/> <input type="button" value="Submit"/>
<pre data-bbox="169 832 734 1634"><!DOCTYPE html> <html> <body> <p>Choose your favorite Web language:</p> <form action="/action_page.php"> <input type="radio" id="html" name="fav_language" value="HTML"> <label for="html">HTML</label>
 <input type="radio" id="css" name="fav_language" value="CSS"> <label for="css">CSS</label>
 <input type="radio" id="javascript" name="fav_language" value="JavaScript"> <label for="javascript">JavaScript</label>

 <input type="submit" value="Submit"> </form> </body> </html></pre>	<p>Choose your favorite Web language:</p> <ul style="list-style-type: none"> <input type="radio"/> HTML <input type="radio"/> CSS <input type="radio"/> JavaScript <input type="button" value="Submit"/>
<pre data-bbox="169 1634 734 1970"><!DOCTYPE html> <html> <body> <h2>Input Button</h2> <input type="button" onclick="alert('Hello World!')" value="Click Me!"></pre>	<h2>Input Button</h2> <input type="button" value="Click Me!"/>

<pre></body> </html></pre>	
<pre><!DOCTYPE html> <html> <body> <h2>Show a Color Picker</h2> <p>The input type="color" is used for input fields that should contain a color.</p> <form action="/action_page.php"> <label for="favcolor">Select your favorite color:</label> <input type="color" id="favcolor" name="favcolor" value="#ff0000"> <input type="submit" value="Submit"> </form> <p>Note: type="color" is not supported in Internet Explorer 11.</p> </body> </html></pre>	<h2>Show a Color Picker</h2> <p>Select your favorite color:  Submit</p>

II. Thẻ CSS

1. Thẻ CSS Color

a. Chức năng

Màu sắc được chỉ định bằng cách sử dụng tên màu được xác định trước hoặc các giá trị RGB, HEX, HSL, RGBA, HSLA

b. Cú pháp

```
style="color:
style="background-color:
style="border:2px solid
```

- RGB

Trong CSS, một màu có thể được chỉ định là giá trị RGB, sử dụng công thức này:

rgb(red, green, blue)

Mỗi tham số (red, green và blue) xác định cường độ của màu trong khoảng từ 0 đến 255.

Ví dụ, rgb(255, 0, 0) được hiển thị là màu đỏ, vì màu đỏ được đặt thành giá trị cao nhất (255) và các màu khác được đặt thành 0.

Để hiển thị màu đen, hãy đặt tất cả các tham số màu thành 0, như sau: rgb(0, 0, 0).

Để hiển thị màu trắng, hãy đặt tất cả các tham số màu thành 255, như sau: rgb(255, 255, 255).

- HEX

Trong CSS, một màu có thể được chỉ định bằng giá trị thập lục phân theo dạng:

#rrggb

Trong đó rr (đỏ), gg (xanh lá cây) và bb (xanh lam) là các giá trị thập lục phân từ 00 đến ff (tương tự như thập phân 0-255).

Ví dụ: #ff0000 được hiển thị là màu đỏ, vì màu đỏ được đặt thành giá trị cao nhất (ff) và các màu khác được đặt thành giá trị thấp nhất (00).

Để hiển thị màu đen, hãy đặt tất cả các giá trị thành 00, như sau: #000000.

Để hiển thị màu trắng, hãy đặt tất cả các giá trị thành ff, như sau: #ffffff.

- HSL

Trong CSS, một màu có thể được chỉ định bằng cách sử dụng sắc độ, độ bão hòa và độ sáng (HSL) theo dạng:

hsl(độ sắc, độ bão hòa, độ sáng)

Sắc độ là một mức độ trên bánh xe màu từ 0 đến 360. 0 là đỏ, 120 là xanh lá cây và 240 là xanh lam.

Độ bão hòa là một giá trị phần trăm. 0% có nghĩa là một sắc thái của màu xám và 100% là toàn bộ màu.

Độ sáng cũng là một phần trăm. 0% là đen, 50% không phải là sáng hay tối, 100% là trắng

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <body> <h1 style="background-color:DodgerBlue;">Hello World</h1> <p style="background-color:Tomato;"> Lorem ipsum dolor sit amet </body> </html></pre>	<p>Hello World</p> <p>Lore<i>m ipsum dolor sit amet</i></p>
<pre><!DOCTYPE html> <html> <body> <h3 style="color:Tomato;">Hello World</h3> <p style="color:DodgerBlue;">Lorem ipsum dolor sit amet</p> <p style="color:MediumSeaGreen;">Ut wisi enim ad minim veniam</p> </body> </html></pre>	<p>Hello World</p> <p><i>Lore<i>m ipsum dolor sit amet</i></i></p> <p><i>Ut wisi enim ad minim veniam</i></p>

```

<!DOCTYPE html>
<html>
<body>

<h1 style="border: 2px solid Tomato;">Hello
World</h1>

<h1 style="border: 2px solid DodgerBlue;">Hello
World</h1>

<h1 style="border: 2px solid Violet;">Hello
World</h1>

</body>
</html>

```

Hello World

Hello World

Hello World

2. Thủ CSS Background

a. Chức năng

Thuộc tính nền CSS được sử dụng để thêm hiệu ứng nền cho các thành phần.

b. Cú pháp

```

background-color: red; /* Hoặc mã màu HEX, RGB, HSL */
background-image: url('image.jpg');
background-position: center top; /* (trái/phải/giữa + trên/dưới) */
background-size: cover; /* Hoặc contain, auto, px, % */
background-repeat: no-repeat; /* Hoặc repeat, repeat-x, repeat-y */
background-attachment: fixed; /* Hoặc scroll, local */
background: red url('image.jpg') no-repeat center/cover fixed;

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre> <html> <head> <style> body { background-color: lightblue; } </style> </head> <body> <h1>Hello World!</h1> <p>This page has a light blue background color!</p> </body> </html> </pre>	<p>Hello World!</p> <p>This page has a light blue background color!</p>

<pre> <!DOCTYPE html> <html> <head> <style> body { background-image: url("paper.gif"); } </style> </head> <body> <h1>Hello World!</h1> <p>This page has an image as the background!</p> </body> </html> </pre>	
---	--

3. Thẻ CSS Border

a. Chức năng

Thuộc tính đường viền CSS cho phép bạn chỉ định kiểu, chiều rộng và màu của đường viền phần tử.

b. Cú pháp

```

border-style: solid; /* Viền liền */
border-style: dashed; /* Viền nét đứt */
border-style: dotted; /* Viền chấm chấm */
border-style: double; /* Viền đôi */
border-style: groove; /* Hiệu ứng khía rãnh */
border-style: ridge; /* Hiệu ứng gồ lên */
border-style: inset; /* Hiệu ứng lõm vào */
border-style: outset; /* Hiệu ứng nổi lên */
border-radius: 10px; /* Làm viền bo góc */
border-top: 2px dashed blue;
border-right: 3px dotted green;
border-bottom: 4px solid black;
border-left: 5px double purple;

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<!DOCTYPE html> <html> <head> <style> p.one { border-style: solid; border-width: 5px; } p.two { border-style: solid; border-width: medium; } p.three { border-style: dotted; border-width: 2px; } p.four { border-style: dotted; border-width: thick; } p.five { border-style: double; border-width: 15px; } p.six { border-style: double; border-width: thick; } </style> </head> <body> <p class="one">Some text.</p> <p class="two">Some text.</p> <p class="three">Some text.</p> <p class="four">Some text.</p> <p class="five">Some text.</p> <p class="six">Some text.</p> </body> </html>	Some text. Some text. Some text. Some text. Some text. Some text.
	Some text. Some text. Some text.
	Some text. Some text. Some text.
	Some text.

```

<!DOCTYPE html>
<html>
<head>
<style>
p.one {
    border-style: solid;
    border-color: red;
}
p.two {
    border-style: solid;
    border-color: green;
}
p.three {
    border-style: dotted;
    border-color: blue;
}
</style>
</head>
<body>
<p class="one">A solid red border</p>
<p class="two">A solid green border</p>
<p class="three">A dotted blue border</p>
</body>
</html>

```

A solid red border

A solid green border

A dotted blue border

```

<!DOCTYPE html>
<html>
<head>
<style>
p {
    border-top-style: dotted;
    border-right-style: solid;
    border-bottom-style: dotted;
    border-left-style: solid;
}
</style>
</head>
<body>
<p>2 different border styles.</p>
</body>
</html>

```

2 different border styles.

4. Thẻ CSS Margin

a. Chức năng

Lẽ được sử dụng để tạo khoảng trống xung quanh các thành phần, bên ngoài bất kỳ đường viền nào được xác định.

b. Cú pháp

```

margin: 10px 20px 30px 40px;
/* top right bottom left */
margin: 10px 20px;
/* top & bottom = 10px, left & right = 20px */

```

```

margin: 10px 20px 30px;
/* top = 10px, left & right = 20px, bottom = 30px */
margin: 0 auto;
/* Căn giữa theo chiều ngang (dùng với width cố định) */

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre> <!DOCTYPE html> <html> <head> <style> div { border: 1px solid black; margin-top: 100px; margin-bottom: 100px; margin-right: 480px; margin-left: 80px; background-color: lightblue; } </style> </head> <body> <div>bruh i'm tired and hungry</div> </body> </html> </pre>	

5. Thẻ CSS Padding

a. Chức năng

Khoảng đệm được sử dụng để tạo khoảng trống xung quanh nội dung của phần tử, bên trong bất kỳ đường viền nào được xác định

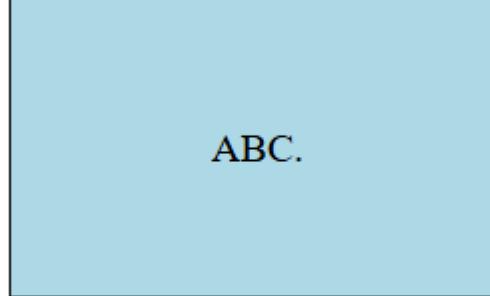
b. Cú pháp

```

padding: 10px 20px 30px 40px;
/* top right bottom left */
padding: 10px 20px;
/* top & bottom = 10px, left & right = 20px */
padding: 10px 20px 30px;
/* top = 10px, left & right = 20px, bottom = 30px */

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <head> <style> div { border: 1px solid black; background-color: lightblue; padding-top: 50px; padding-right: 30px; padding-bottom: 50px; padding-left: 80px; } </style> </head> <body> <div>ABC.</div> </body> </html></pre>	

6. Thẻ CSS Height

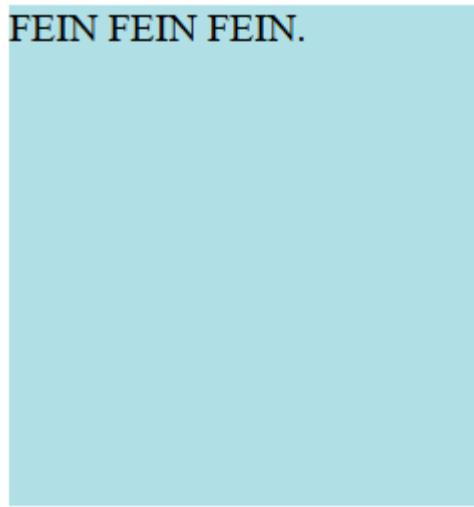
a. Chức năng

Thuộc tính CSS height và width được sử dụng để thiết lập chiều cao và chiều rộng của một phần tử.

b. Cú pháp

```
height: 200px; /* Đơn vị px, %, em, rem, vh, vw... */
height: 50%; /* Chiều cao bằng 50% thẻ cha */
height: 100vh; /* 100% chiều cao màn hình */
min-height: 100px; /* Chiều cao nhỏ nhất */
max-height: 500px; /* Chiều cao lớn nhất */
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <head> <style> div { height: 200px; width: 50%; background-color: powderblue; } </style> </head> <body> <h2></h2> <div>FEIN FEIN FEIN.</div> </body> </html></pre>	

7. Thẻ CSS Outline

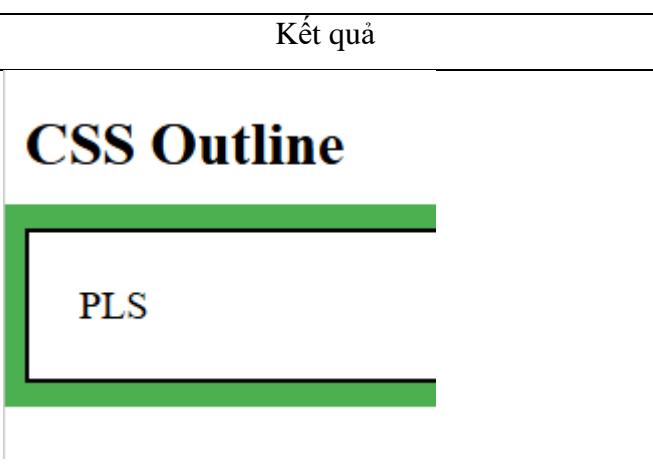
a. Chức năng

Đường viền là đường kẻ được vẽ bên ngoài đường viền của phần tử.

b. Cú pháp

```
outline-width: 3px; /* Độ dày */
outline-style: dashed; /* Kiểu viền */
outline-color: blue; /* Màu sắc */
outline-style: solid; /* Viền liền */
outline-style: dashed; /* Viền nét đứt */
outline-style: dotted; /* Viền chấm chấm */
outline-style: double; /* Viền đôi */
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <head> <style> p { border: 2px solid black; outline: #4CAF50 solid 10px; margin: auto; padding: 20px; text-align: LEFT; } </style> </head> <body> <h1>CSS Outline</h1> <p>PLS</p> </body> </html></pre>	

```

</style>
</head>
<body>

<h2>CSS Outline</h2>
<p>PLS</p>

</body>
</html>

```

```

<!DOCTYPE html>
<html>
<head>
<style>
p.ex1 {
    border: 2px solid black;
    outline-style: solid;
    outline-color: red;
}

p.ex2 {
    border: 2px solid black;
    outline-style: dotted;
    outline-color: blue;
}

p.ex3 {
    border: 2px solid black;
    outline-style: outset;
    outline-color: grey;
}
</style>
</head>
<body>

```

```

<p class="ex1">A solid red outline.</p>
<p class="ex2">A dotted blue outline.</p>
<p class="ex3">An outset grey outline.</p>

```

```

</body>
</html>

```

A solid red outline.

A dotted blue outline.

An outset grey outline.

8. Thẻ CSS Text

a. Chức năng

CSS có rất nhiều thuộc tính để định dạng văn bản.

b. Cú pháp

Thuộc tính color được sử dụng để thiết lập màu sắc cho văn bản. Màu sắc được chỉ định bởi:

- tên màu - như "đỏ"

- một giá trị HEX - như "#ff0000"
- một giá trị RGB - như "rgb(255,0,0)"

Xem Giá trị màu CSS để biết danh sách đầy đủ các giá trị màu có thể có.

Màu văn bản mặc định cho một trang được xác định trong bộ chọn nội dung.

```
body {
  color: blue;
}

h1 {
  color: green;
}
```

Trong ví dụ này, chúng tôi xác định cả thuộc tính màu nền và thuộc tính màu:

```
body {
  background-color: lightgrey;
  color: blue;
}

h1 {
  background-color: black;
  color: white;
}

div {
  background-color: blue;
  color: white;
}
```

Thuộc tính text-align được sử dụng để thiết lập căn chỉnh theo chiều ngang của văn bản.

Một văn bản có thể được căn trái hoặc phải, căn giữa hoặc căn đều.

Ví dụ sau đây hiển thị văn bản được căn giữa và căn trái và phải (căn chỉnh trái là mặc định nếu hướng văn bản là từ trái sang phải và căn chỉnh phải là mặc định nếu hướng văn bản là từ phải sang trái):

```
h1 {
  text-align: center;
}

h2 {
  text-align: left;
}

h3 {
  text-align: right;
}
```

Khi thuộc tính text-align được đặt thành "justify", mỗi dòng được kéo dài sao cho mỗi dòng có chiều rộng bằng nhau và lề trái và lề phải đều thẳng (như trong tạp chí và báo):

```
div {
  text-align: justify;
}
```

Thuộc tính text-align-last chỉ định cách căn chỉnh dòng cuối cùng của văn bản.

```
p.a {  
    text-align-last: right;  
}  
  
p.b {  
    text-align-last: center;  
}  
  
p.c {  
    text-align-last: justify;  
}
```

Các thuộc tính hướng và unicode-bidi có thể được sử dụng để thay đổi hướng văn bản của một phần tử:

```
p {  
    direction: rtl;  
    unicode-bidi: bidi-override;  
}
```

Thuộc tính Vertical-align thiết lập cách căn chỉnh theo chiều dọc của một phần tử.

```
img.a {  
    vertical-align: baseline;  
}  
  
img.b {  
    vertical-align: text-top;  
}  
  
img.c {  
    vertical-align: text-bottom;  
}  
  
img.d {  
    vertical-align: sub;  
}  
  
img.e {  
    vertical-align: super;  
}
```

Thuộc tính text-decoration-line được sử dụng để thêm dòng trang trí vào văn bản.

```
h1 {  
    text-decoration-line: overline;  
}  
  
h2 {  
    text-decoration-line: line-through;  
}  
  
h3 {  
    text-decoration-line: underline;  
}  
  
p {
```

```
text-decoration-line: overline underline;
```

```
}
```

Thuộc tính text-decoration-màu được sử dụng để đặt màu cho đường trang trí.

```
h1 {  
    text-decoration-line: overline;  
    text-decoration-color: red;  
}  
  
h2 {  
    text-decoration-line: line-through;  
    text-decoration-color: blue;  
}  
  
h3 {  
    text-decoration-line: underline;  
    text-decoration-color: green;  
}  
  
p {  
    text-decoration-line: overline underline;  
    text-decoration-color: purple;  
}
```

Thuộc tính text-decoration-style được sử dụng để thiết lập kiểu của đường trang trí.

```
h1 {  
    text-decoration-line: underline;  
    text-decoration-style: solid;  
}  
  
h2 {  
    text-decoration-line: underline;  
    text-decoration-style: double;  
}  
  
h3 {  
    text-decoration-line: underline;  
    text-decoration-style: dotted;  
}  
  
p.ex1 {  
    text-decoration-line: underline;  
    text-decoration-style: dashed;  
}  
  
p.ex2 {  
    text-decoration-line: underline;  
    text-decoration-style: wavy;  
}  
  
p.ex3 {  
    text-decoration-line: underline;  
    text-decoration-color: red;
```

```
    text-decoration-style: wavy;  
}
```

Thuộc tính text-decoration-độ dày được sử dụng để thiết lập độ dày của đường trang trí.

```
h1 {  
    text-decoration-line: underline;  
    text-decoration-thickness: auto;  
}  
  
h2 {  
    text-decoration-line: underline;  
    text-decoration-thickness: 5px;  
}  
  
h3 {  
    text-decoration-line: underline;  
    text-decoration-thickness: 25%;  
}  
  
p {  
    text-decoration-line: underline;  
    text-decoration-color: red;  
    text-decoration-style: double;  
    text-decoration-thickness: 5px;  
}
```

Thuộc tính text-trang trí là thuộc tính viết tắt của:

- dòng văn bản trang trí (bắt buộc)
- màu văn bản trang trí (tùy chọn)
- kiểu trang trí văn bản (tùy chọn)
- độ dày văn bản trang trí (tùy chọn)

```
h1 {  
    text-decoration: underline;  
}  
  
h2 {  
    text-decoration: underline red;  
}  
  
h3 {  
    text-decoration: underline red double;  
}  
  
p {  
    text-decoration: underline red double 5px;  
}
```

Thuộc tính text-transform được sử dụng để chỉ định chữ hoa và chữ thường trong văn bản.

Nó có thể được sử dụng để biến mọi thứ thành chữ hoa hoặc chữ thường hoặc viết hoa chữ cái đầu tiên của mỗi từ:

```
p.uppercase {  
    text-transform: uppercase;
```

```
}

p.lowercase {
    text-transform: lowercase;
}

p.capitalize {
    text-transform: capitalize;
}
```

Thuộc tính **thut lè** văn bản được sử dụng để chỉ định mức **thut** dòng của dòng đầu tiên của văn bản:

```
p {
    text-indent: 50px;
}
```

Thuộc tính **letter-spacing** được sử dụng để xác định khoảng cách giữa các ký tự trong văn bản.

Ví dụ sau đây minh họa cách tăng hoặc giảm khoảng cách giữa các ký tự:

```
h1 {
    letter-spacing: 5px;
}

h2 {
    letter-spacing: -2px;
}
```

Thuộc tính **line-height** được sử dụng để chỉ định khoảng cách giữa các dòng:

```
p.small {
    line-height: 0.8;
}

p.big {
    line-height: 1.8;
}
```

Thuộc tính **word-spacing** được sử dụng để chỉ định khoảng cách giữa các từ trong văn bản.

Ví dụ sau đây minh họa cách tăng hoặc giảm khoảng cách giữa các từ:

```
p.one {
    word-spacing: 10px;
}

p.two {
    word-spacing: -2px;
}
```

Thuộc tính **khoảng trắng** chỉ định cách xử lý khoảng trắng bên trong một phần tử.

Ví dụ này minh họa cách tắt tính **năng ngắn dòng** văn bản bên trong một phần tử:

```
p {
    white-space: nowrap;
}
```

Thuộc tính **text-shadow** thêm bóng vào văn bản.

Trong cách sử dụng đơn giản nhất, bạn chỉ xác định bóng ngang (2px) và bóng dọc (2px):

```
h1 {  
    text-shadow: 2px 2px;  
}
```

c. Ví dụ minh họa

```
<!DOCTYPE html>  
<html lang="vi">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport"  
content="width=device-width, initial-  
scale=1.0">  
    <title>CSS Text</title>  
    <style>  
        body {  
            font-family: Arial, sans-serif;  
            background: #f4f4f4;  
            color: #333;  
            padding: 20px;  
        }  
        h1 { text-align: center; color:  
#4CAF50; text-transform: uppercase;  
}  
        p { margin-bottom: 10px; }  
        .justify { text-align: justify; }  
        .center { text-align: center; font-  
weight: bold; }  
        .right { text-align: right; font-  
style: italic; }  
        .underline { text-decoration:  
underline; }  
        .line-through { text-decoration:  
line-through; }  
        .letter-spacing { letter-spacing:  
3px; }  
        .line-height { line-height: 2; }  
        .capitalize { text-transform:  
capitalize; }  
    </style>  
</head>  
<body>  
    <h1>Định dạng văn bản với  
CSS</h1>  
    <p class="justify">Đoạn văn bản  
được căn **justify**, đều hai bên lề  
và trông chuyên nghiệp hơn.</p>  
    <p class="center">Văn bản căn  
giữa và in đậm.</p>  
    <p class="right">Văn bản căn phải  
và in nghiêng.</p>  
    <p class="underline">Văn bản này  
có gạch chân.</p>
```

ĐỊNH DẠNG VĂN BẢN VỚI CSS

Đoạn văn bản được căn **justify**, đều hai bên lề và trông chuyên nghiệp hơn.

Văn bản căn giữa và in đậm.

Văn bản căn phải và in nghiêng.

Văn bản này có gạch chân.

Văn bản này có gạch ngang.

Khoảng cách giữa các chữ cái được tăng lên.

Văn bản có khoảng cách dòng lớn hơn để dễ đọc.

Đây Là Văn Bản Có Chữ Cái Đầu Của Mỗi Từ Được Viết Hoa.

```

<p class="line-through">Văn bản  

này có gạch ngang.</p>
<p class="letter-spacing">Khoảng  

cách giữa các chữ cái được tăng  

lên.</p>
<p class="line-height">Văn bản có  

khoảng cách dòng lớn hơn để dễ  

đọc.</p>
<p class="capitalize">đây là văn  

bản có chữ cái đầu của mỗi từ được  

viết hoa.</p>
</body>
</html>
```

9. Thẻ CSS Fonts

a. Chức năng

Việc chọn đúng phông chữ có tác động rất lớn đến cách người đọc trải nghiệm một trang web.

Phông chữ phù hợp có thể tạo ra bản sắc mạnh mẽ cho thương hiệu của bạn.

Sử dụng phông chữ dễ đọc là điều quan trọng. Phông chữ thêm giá trị cho văn bản của bạn. Điều quan trọng nữa là chọn đúng màu sắc và kích thước văn bản cho phông chữ.

Trong CSS có năm họ phông chữ chung:

Phông chữ Serif có một nét nhỏ ở rìa mỗi chữ cái. Chúng tạo cảm giác trang trọng và sang trọng.

Phông chữ Sans-serif có đường nét rõ ràng (không có nét nhỏ kèm theo). Chúng tạo ra một cái nhìn hiện đại và tối giản.

Phông chữ đơn cách - ở đây tất cả các chữ cái đều có cùng chiều rộng cố định. Họ tạo ra một cái nhìn máy móc.

Phông chữ chũ thảo bắt chước chũ viết tay của con người.

Phông chữ tưởng tượng là phông chữ trang trí/vui tươi.

Tất cả các tên phông chữ khác nhau đều thuộc về một trong các họ phông chữ chung.

b. Cú pháp

Trong CSS, chúng tôi sử dụng thuộc tính font-family để chỉ định phông chữ của văn bản.

```

.p1 {
  font-family: "Times New Roman", Times, serif;
}

.p2 {
  font-family: Arial, Helvetica, sans-serif;
}

.p3 {
  font-family: "Lucida Console", "Courier New", monospace;
}
```

Tuy nhiên, không có phông chữ nào an toàn cho web 100%. Luôn có khả năng không tìm thấy phông chữ hoặc phông chữ không được cài đặt đúng cách.

Vì vậy, điều rất quan trọng là luôn sử dụng phông chữ dự phòng.

Điều này có nghĩa là bạn nên thêm danh sách "phông chữ dự phòng" tương tự vào thuộc tính họ phông chữ. Nếu phông chữ đầu tiên không hoạt động, trình duyệt sẽ thử phông chữ tiếp theo và phông chữ tiếp theo, v.v. Luôn kết thúc danh sách bằng tên họ phông chữ chung.

```
p {  
    font-family: Tahoma, Verdana, sans-serif;  
}
```

Thuộc tính font-style chủ yếu được sử dụng để chỉ định văn bản in nghiêng.

Thuộc tính này có ba giá trị:

- normal - Văn bản được hiển thị bình thường
- in nghiêng - Văn bản được hiển thị in nghiêng
- xiên - Văn bản "nghiêng" (xiên rất giống với chữ nghiêng, nhưng ít được hỗ trợ hơn)

```
p.normal {  
    font-style: normal;  
}  
  
p.italic {  
    font-style: italic;  
}  
  
p.oblique {  
    font-style: oblique;  
}
```

Thuộc tính font-weight chỉ định độ dày của phông chữ:

```
p.normal {  
    font-weight: normal;  
}  
  
p.thick {  
    font-weight: bold;  
}
```

Thuộc tính font-variant chỉ định xem văn bản có được hiển thị bằng phông chữ viết hoa nhỏ hay không.

Trong phông chữ viết hoa nhỏ, tất cả các chữ cái viết thường đều được chuyển thành chữ in hoa. Tuy nhiên, các chữ in hoa được chuyển đổi sẽ xuất hiện ở cỡ chữ nhỏ hơn các chữ in hoa ban đầu trong văn bản.

```
p.normal {  
    font-variant: normal;  
}  
  
p.small {  
    font-variant: small-caps;  
}
```

Đặt kích thước văn bản bằng pixel sẽ cung cấp cho bạn toàn quyền kiểm soát kích thước văn bản:

```
h1 {  
    font-size: 40px;
```

```
}
```

```
h2 {  
    font-size: 30px;  
}  
  
p {  
    font-size: 14px;  
}
```

Để cho phép người dùng thay đổi kích thước văn bản (trong menu trình duyệt), nhiều nhà phát triển sử dụng em thay vì pixel.

1em bằng cỡ chữ hiện tại. Kích thước văn bản mặc định trong trình duyệt là 16px. Vì vậy, kích thước mặc định của 1em là 16px.

Kích thước có thể được tính từ pixel đến em bằng công thức sau: pixels/16=em

```
h1 {  
    font-size: 2.5em; /* 40px/16=2.5em */  
}  
  
h2 {  
    font-size: 1.875em; /* 30px/16=1.875em */  
}  
  
p {  
    font-size: 0.875em; /* 14px/16=0.875em */  
}
```

Giải pháp hoạt động trong tất cả các trình duyệt là đặt kích thước phông chữ mặc định theo phần trăm cho phần tử <body>:

```
body {  
    font-size: 100%;  
}  
  
h1 {  
    font-size: 2.5em;  
}  
  
h2 {  
    font-size: 1.875em;  
}  
  
p {  
    font-size: 0.875em;  
}
```

Kích thước văn bản có thể được đặt bằng đơn vị vw, có nghĩa là "chiều rộng khung nhìn".

Bằng cách đó, kích thước văn bản sẽ tuân theo kích thước của cửa sổ trình duyệt:

```
<h1 style="font-size:10vw">Hello World</h1>
```

Chỉ cần thêm một liên kết biểu định kiểu đặc biệt vào phần <head> rồi tham khảo phông chữ trong CSS.

```
<head>  
<link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Sofia">
```

```

<style>
body {
  font-family: "Sofia", sans-serif;
}
</style>
</head>

```

Một phông chữ nên là ông chủ. Điều này thiết lập hệ thống phân cấp cho các phông chữ trên trang của bạn. Điều này có thể đạt được bằng cách thay đổi kích thước, trọng lượng và màu sắc.

```

body {
  background-color: black;
  font-family: Verdana, sans-serif;
  font-size: 16px;
  color: gray;
}

h1 {
  font-family: Georgia, serif;
  font-size: 60px;
  color: white;
}

```

Để rút ngắn mã, cũng có thể chỉ định tất cả các thuộc tính phông chữ riêng lẻ trong một thuộc tính.

Thuộc tính phông chữ là thuộc tính viết tắt của:

- kiểu phông chữ
- biến thể phông chữ
- độ dày phông chữ
- cỡ chữ/chiều cao dòng
- họ phông chữ

```

p.a {
  font: 20px Arial, sans-serif;
}

p.b {
  font: italic small-caps bold 12px/30px Georgia, serif;
}

```

c. Ví dụ minh họa

```

<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
  <title>CSS Font</title>
  <style>
    body { font-family: Arial, sans-
serif; background: #f4f4f4; color:
#333; padding: 20px; }

```

CHÀO MỪNG ĐẾN VỚI CSS FONTS

Đây là một ví dụ tổng quát về font trong CSS.

Bạn có thể thay đổi kiểu chữ, kích thước, trọng lượng, khoảng cách chữ và nhiều thuộc tính khác.

Ví dụ khoảng cách giữa các chữ cái.

Ví dụ khoảng cách giữa các dòng giúp văn bản dễ đọc hơn.

```

    h1 { font-family: Georgia, serif;
font-size: 36px; text-align: center;
color: #4CAF50; text-transform:
uppercase; }
    p { font-size: 18px; line-height:
1.6; text-align: justify; margin-
bottom: 10px; }
.italic { font-style: italic; color:
#ff5722; }
.bold { font-weight: bold; }
.letter-spacing { letter-spacing:
2px; }
.line-height { line-height: 2; }
</style>
</head>
<body>
    <h1>Chào mừng đến với CSS
Fonts</h1>
    <p>Đây là một ví dụ tổng quát về
<span class="bold">font trong
CSS</span>.</p>
    <p class="italic">Bạn có thể thay
đổi kiểu chữ, kích thước, trọng
lượng, khoảng cách chữ và nhiều
thuộc tính khác.</p>
    <p class="letter-spacing">Ví dụ
khoảng cách giữa các chữ cái.</p>
    <p class="line-height">Ví dụ
khoảng cách giữa các dòng giúp văn
bản dễ đọc hơn.</p>
</body>
</html>

```

10. Thẻ CSS Icons

a. Chức năng

Các biểu tượng có thể dễ dàng được thêm vào trang HTML của bạn bằng cách sử dụng thư viện biểu tượng.

b. Cú pháp

Cách đơn giản nhất để thêm biểu tượng vào trang HTML của bạn là sử dụng thư viện biểu tượng, chẳng hạn như Font Awesome.

Thêm tên của lớp biểu tượng được chỉ định vào bất kỳ thành phần HTML nội tuyến nào (như `<i>` hoặc ``).

Tất cả các biểu tượng trong thư viện biểu tượng bên dưới đều là các vectơ có thể mở rộng và có thể được tùy chỉnh bằng CSS (kích thước, màu sắc, bóng, v.v.)

Font Awesome Icons

```

<!DOCTYPE html>
<html>
<head>
<title>Font Awesome Icons</title>

```

```

<meta name="viewport" content="width=device-width, initial-scale=1">
<script src="https://kit.fontawesome.com/a076d05399.js" crossorigin="anonymous"></script>
<!--Get your own code at fontawesome.com-->
</head>
<body>

<h1>Font Awesome icon library</h1>

<p>Some Font Awesome icons:</p>
<i class="fas fa-cloud"></i>
<i class="fas fa-heart"></i>
<i class="fas fa-car"></i>
<i class="fas fa-file"></i>
<i class="fas fa-bars"></i>

<p>Styled Font Awesome icons (size and color):</p>
<i class="fas fa-cloud" style="font-size:24px;"></i>
<i class="fas fa-cloud" style="font-size:36px;"></i>
<i class="fas fa-cloud" style="font-size:48px;color:red;"></i>
<i class="fas fa-cloud" style="font-size:60px;color:lightblue;"></i>

</body>
</html>

```

Bootstrap Icons

```

<!DOCTYPE html>
<html>
<head>
<title>Bootstrap Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet"
href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
</head>
<body class="container">

<h1>Bootstrap icon library</h1>

<p>Some Bootstrap icons:</p>
<i class="glyphicon glyphicon-cloud"></i>
<i class="glyphicon glyphicon-remove"></i>
<i class="glyphicon glyphicon-user"></i>
<i class="glyphicon glyphicon-envelope"></i>
<i class="glyphicon glyphicon-thumbs-up"></i>
<br><br>

<p>Styled Bootstrap icons (size and color):</p>
<i class="glyphicon glyphicon-cloud" style="font-size:24px;"></i>
<i class="glyphicon glyphicon-cloud" style="font-size:36px;"></i>
<i class="glyphicon glyphicon-cloud" style="font-size:48px;color:red;"></i>
<i class="glyphicon glyphicon-cloud" style="font-size:60px;color:lightblue;"></i>

</body>

```

```
</html>
```

Google Icons

```
<!DOCTYPE html>
<html>
<head>
<title>Google Icons</title>
<meta name="viewport" content="width=device-width, initial-scale=1">
<link rel="stylesheet" href="https://fonts.googleapis.com/icon?family=Material+Icons">
</head>
<body>

<h1>Google icon library</h1>

<p>Some Google icons:</p>
<i class="material-icons">cloud</i>
<i class="material-icons">favorite</i>
<i class="material-icons">attachment</i>
<i class="material-icons">computer</i>
<i class="material-icons">traffic</i>
<br><br>

<p>Styled Google icons (size and color):</p>
<i class="material-icons" style="font-size:24px;">cloud</i>
<i class="material-icons" style="font-size:36px;">cloud</i>
<i class="material-icons" style="font-size:48px;color:red;">cloud</i>
<i class="material-icons" style="font-size:60px;color:lightblue;">cloud</i>

</body>
</html>
```

c. Ví dụ minh họa

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
content="width=device-width, initial-scale=1.0">
  <title>CSS Icons</title>
  <script
src="https://kit.fontawesome.com/a076d05399.js"
crossorigin="anonymous"></script>
<style>
  body {
    font-family: Arial, sans-serif;
    text-align: center;
    margin-top: 50px;
  }
  .icon {
    font-size: 50px;
    color: blue;
    margin: 10px;
  }
</style>
```

Ví dụ CSS Icons với Font Awesome



```

</style>
</head>
<body>
  <h2>Ví dụ CSS Icons với Font Awesome</h2>
  <i class="fas fa-home icon"></i> <!-- Icon Home -
-&gt;
  &lt;i class="fas fa-user icon"&gt;&lt;/i&gt; <!-- Icon User --&gt;
  &lt;i class="fas fa-envelope icon"&gt;&lt;/i&gt; <!-- Icon
Email --&gt;
&lt;/body&gt;
&lt;/html&gt;
</pre>

```

11. Thẻ CSS List

a. Chức năng

CSS List cho phép bạn:

- Đặt các điểm đánh dấu mục danh sách khác nhau cho danh sách được sắp xếp
- Đặt các điểm đánh dấu mục danh sách khác nhau cho danh sách không có thứ tự
- Đặt hình ảnh làm điểm đánh dấu mục danh sách
- Thêm màu nền vào danh sách và mục danh sách

b. Cú pháp

Thuộc tính list-style-image chỉ định một hình ảnh làm điểm đánh dấu mục danh sách:

```

ul {
  list-style-image: url('sqpurple.gif');
}

```

Thuộc tính list-style-position chỉ định vị trí của các điểm đánh dấu mục danh sách (dấu đầu dòng).

```

ul.a {
  list-style-position: outside;
}

ul.b {
  list-style-position: inside;
}

```

Thuộc tính list-style-type:none cũng có thể được sử dụng để xóa điểm đánh dấu/dấu đầu dòng. Lưu ý rằng danh sách cũng có lề và phần đệm mặc định. Để loại bỏ điều này, hãy thêm lề:0 và phần đệm:0 vào hoặc :

```

ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}

```

Thuộc tính list-style là thuộc tính viết tắt. Nó được sử dụng để đặt tất cả các thuộc tính danh sách trong một khai báo:

```
ul {  
    list-style: square inside url("sqpurple.gif");  
}
```

Chúng ta cũng có thể tạo kiểu cho danh sách bằng màu sắc để làm cho chúng trông thú vị hơn một chút.

Bất cứ điều gì được thêm vào thẻ `` hoặc `` sẽ ảnh hưởng đến toàn bộ danh sách, trong khi các thuộc tính được thêm vào thẻ `` sẽ ảnh hưởng đến từng mục trong danh sách:

```
ol {  
    background: #ff9999;  
    padding: 20px;  
}  
  
ul {  
    background: #3399ff;  
    padding: 20px;  
}  
  
ol li {  
    background: #ffe5e5;  
    color: darkred;  
    padding: 5px;  
    margin-left: 35px;  
}  
  
ul li {  
    background: #cce5ff;  
    color: darkblue;  
    margin: 5px;  
}
```

c. Ví dụ minh họa

```
<!DOCTYPE html>  
<html lang="vi">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport"  
        content="width=device-width, initial-  
        scale=1.0">  
    <title>Danh Sách CSS</title>  
    <style>  
        body {  
            font-family: Arial, sans-serif;  
        }  
  
        /* Kiểu danh sách */  
        ul.custom-list {  
            list-style: none; /* Xóa kiểu mặc định */  
            padding: 0;  
        }  
  
        /* Kiểu từng mục trong danh sách */  
        ul.custom-list li {
```

Danh sách có biểu tượng tùy chỉnh



```

padding: 10px;
margin: 5px 0;
background-color: #f0f0f0;
border-radius: 5px;
display: flex;
align-items: center;
}

/* Biểu tượng tùy chỉnh */
ul.custom-list li::before {
    content: "✓"; /* Ký tự biểu tượng */
    color: green;
    font-weight: bold;
    margin-right: 10px;
}
</style>
</head>
<body>

<h2>Danh sách có biểu tượng tùy chỉnh</h2>
<ul class="custom-list">
    <li>Mục 1</li>
    <li>Mục 2</li>
    <li>Mục 3</li>
</ul>

</body>
</html>

```

12. Thẻ CSS Table

a. Chức năng

Giao diện của bảng HTML có thẻ được cải thiện đáng kể bằng CSS:

b. Cú pháp

Để chỉ định đường viền bảng trong CSS, hãy sử dụng thuộc tính border.

Ví dụ bên dưới chỉ định đường viền liền nét cho các phần tử `<table>`, `<th>` và `<td>`:

```

table, th, td {
    border: 1px solid;
}

```

Bảng trên có vẻ nhỏ trong một số trường hợp. Nếu bạn cần một bảng trải dài toàn bộ màn hình (tùy chiều rộng), hãy thêm width: 100% vào phần tử `<table>`:

```

table {
    width: 100%;
}

```

Thuộc tính border-collapse thiết lập xem các đường viền của bảng có được thu gọn thành một đường viền duy nhất hay không:

```
table {  
    border-collapse: collapse;  
}
```

Nếu bạn chỉ muốn có đường viền quanh bảng thì chỉ xác định thuộc tính đường viền cho <table>:

```
table {  
    border: 1px solid;  
}
```

Chiều rộng và chiều cao của bảng được xác định bởi thuộc tính chiều rộng và chiều cao.

Ví dụ bên dưới đặt chiều rộng của bảng thành 100% và chiều cao của các phần tử <th> thành 70px:

```
table {  
    width: 100%;  
}  
  
th {  
    height: 70px;  
}
```

Thuộc tính text-align đặt căn chỉnh theo chiều ngang (như trái, phải hoặc giữa) của nội dung trong <th> hoặc <td>.

Theo mặc định, nội dung của phần tử <th> được căn giữa và nội dung của phần tử <td> được căn trái.

```
td {  
    text-align: center;  
}
```

Thuộc tính căn chỉnh dọc đặt căn chỉnh dọc (như trên, dưới hoặc giữa) của nội dung trong <th> hoặc <td>.

Theo mặc định, căn chỉnh theo chiều dọc của nội dung trong bảng là ở giữa (đối với cả phần tử <th> và <td>).

```
td {  
    height: 50px;  
    vertical-align: bottom;  
}
```

Để kiểm soát khoảng cách giữa đường viền và nội dung trong bảng, hãy sử dụng thuộc tính đệm trên các phần tử <td> và <th>:

```
th, td {  
    padding: 15px;  
    text-align: left;  
}
```

Thêm thuộc tính border-bottom vào <th> và <td> cho dài phân cách ngang:

```
th, td {  
    border-bottom: 1px solid #ddd;  
}
```

Sử dụng bộ chọn :hover trên <tr> để đánh dấu các hàng trong bảng khi di chuột qua:

```
tr:hover {background-color: coral;}
```

Đối với các bảng sọc ngựa vằn, hãy sử dụng bộ chọn nth-child() và thêm màu nền cho tất cả các hàng trong bảng chẵn (hoặc lẻ):

```
tr:nth-child(even) {background-color: #f2f2f2;}
```

Ví dụ bên dưới chỉ định màu nền và màu văn bản của phần tử <th>:

```

th {
    background-color: #04AA6D;
    color: white;
}

```

c. Ví dụ minh họa

```

<!DOCTYPE html>
<html lang="vi">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
initial-scale=1.0">
    <title>Bảng CSS Đẹp</title>
    <style>
        table { width: 100%; border-collapse: collapse; font-
family: Arial, sans-serif; }
        th { background: #4CAF50; color: white; padding: 10px;
text-align: left; }
        td { padding: 10px; border-bottom: 1px solid #ddd; }
        tr:nth-child(even) { background: #f2f2f2; }
        tr:hover { background: #ddd; }
    </style>
</head>
<body>
    <h2>Danh sách công ty</h2>
    <table>

        <tr><th>Company</th><th>Contact</th><th>Country</th></
tr>
        <tr><td>Alfreds Futterkiste</td><td>Maria
Anders</td><td>Germany</td></tr>
        <tr><td>Berglunds snabbköp</td><td>Christina Berglund
Berglund</td><td>Sweden</td></tr>
        <tr><td>Centro comercial Moctezuma</td><td>Francisco Chang
Chang</td><td>Mexico</td></tr>
        <tr><td>Island Trading</td><td>Helen Bennett
Bennett</td><td>UK</td></tr>
    </table>
</body>
</html>

```

Danh sách công ty		
Company	Contact	Country
Alfreds Futterkiste	Maria Anders	Germany
Berglunds snabbköp	Christina Berglund	Sweden
Centro comercial Moctezuma	Francisco Chang	Mexico
Island Trading	Helen Bennett	UK

13. Thẻ CSS Navigation bar

a. Chức năng

Việc điều hướng dễ sử dụng là điều quan trọng đối với bất kỳ trang web nào.

Với CSS, bạn có thể biến các menu HTML thành chân thành các thanh điều hướng đẹp mắt.

b. Cú pháp

Thanh điều hướng cần có HTML tiêu chuẩn làm cơ sở.

Trong ví dụ của chúng tôi, chúng tôi sẽ xây dựng thanh điều hướng từ danh sách HTML tiêu chuẩn.

Thanh điều hướng về cơ bản là một danh sách các liên kết, vì vậy việc sử dụng các phần tử `` và `` là hoàn toàn hợp lý:

```
<ul>
  <li><a href="default.asp">Home</a></li>
  <li><a href="news.asp">News</a></li>
  <li><a href="contact.asp">Contact</a></li>
  <li><a href="about.asp">About</a></li>
</ul>
```

Bây giờ, hãy xóa dấu đầu dòng, lè và phần đệm khỏi danh sách:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

Để tạo thanh điều hướng dọc, bạn có thể tạo kiểu cho các phần tử `<a>` bên trong danh sách, ngoài mã từ trang trước:

```
li a {
  display: block;
  width: 60px;
}
```

Ví dụ giải thích:

Hiển thị: khối; - Hiển thị các liên kết dưới dạng các thành phần khối làm cho toàn bộ khu vực liên kết có thể nhấp vào được (không chỉ văn bản) và nó cho phép chúng ta chỉ định chiều rộng (và phần đệm, lè, chiều cao, v.v. nếu bạn muốn)

chiều rộng: 60px; - Các phần tử khối chiếm toàn bộ chiều rộng có sẵn theo mặc định. Chúng tôi muốn chỉ định chiều rộng 60 pixel

Bạn cũng có thể đặt chiều rộng của `` và xóa chiều rộng của `<a>` vì chúng sẽ chiếm toàn bộ chiều rộng có sẵn khi được hiển thị dưới dạng phần tử khối. Điều này sẽ tạo ra kết quả tương tự như ví dụ trước của chúng tôi:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
  width: 60px;
}

li a {
  display: block;
}
```

Tạo thanh điều hướng dọc cơ bản với màu nền xám và thay đổi màu nền của các liên kết khi người dùng di chuyển chuột qua chúng:

```
ul {
  list-style-type: none;
  margin: 0;
  padding: 0;
}
```

```

width: 200px;
background-color: #f1f1f1;
}

li a {
display: block;
color: #000;
padding: 8px 16px;
text-decoration: none;
}

/* Change the link color on hover */
li a:hover {
background-color: #555;
color: white;
}

```

Thêm một lớp "hoạt động" vào liên kết hiện tại để cho người dùng biết họ đang ở trang nào:

```

.active {
background-color: #04AA6D;
color: white;
}

```

Thêm text-align:center vào hoặc <a> để căn giữa các liên kết.

Thêm thuộc tính đường viền vào để thêm đường viền xung quanh thanh điều hướng. Nếu bạn cũng muốn có đường viền bên trong thanh điều hướng, hãy thêm border-bottom vào tất cả các phần tử , ngoại trừ phần tử cuối cùng:

```

ul {
border: 1px solid #555;
}

li {
text-align: center;
border-bottom: 1px solid #555;
}

li:last-child {
border-bottom: none;
}

```

Tạo điều hướng bên có chiều cao đầy đủ, "dính":

```

ul {
list-style-type: none;
margin: 0;
padding: 0;
width: 25%;
background-color: #f1f1f1;
height: 100%; /* Full height */
position: fixed; /* Make it stick, even on scroll */
overflow: auto; /* Enable scrolling if the sidenav has too much
content */
}

```

Một cách để xây dựng thanh điều hướng ngang là chỉ định các phần tử làm nội tuyến, bên cạnh mã "chuẩn" từ trang trước:

```
li {  
    display: inline;  
}
```

Ví dụ giải thích:

hiển thị: nội tuyến; - Theo mặc định, phần tử là phần tử khôi. Ở đây, chúng tôi xóa các ngắt dòng trước và sau mỗi mục danh sách để hiển thị chúng trên một dòng

Một cách khác để tạo thanh điều hướng ngang là thả nỗi các phần tử và chỉ định bối cảnh cho các liên kết điều hướng:

```
li {  
    float: left;  
}  
  
a {  
    display: block;  
    padding: 8px;  
    background-color: #dddddd;  
}
```

Ví dụ giải thích:

nỗi: trái; - Sử dụng float để các phần tử khôi nỗi cạnh nhau

hiển thị: khôi; - Cho phép chúng tôi chỉ định phần đệm (và chiều cao, chiều rộng, lề, v.v. nếu bạn muốn)

phần đệm: 8px; - Chỉ định một số khoảng đệm giữa mỗi phần tử <a> để làm cho chúng trông đẹp mắt

màu nền: #dddddd; - Thêm màu nền xám cho mỗi phần tử <a>

```
ul {  
    background-color: #dddddd;  
}
```

Tạo thanh điều hướng ngang cơ bản với màu nền tối và thay đổi màu nền của các liên kết khi người dùng di chuyển chuột qua chúng:

```
ul {  
    list-style-type: none;  
    margin: 0;  
    padding: 0;  
    overflow: hidden;  
    background-color: #333;  
}  
  
li {  
    float: left;  
}  
  
li a {  
    display: block;  
    color: white;  
    text-align: center;  
    padding: 14px 16px;  
}
```

```

    text-decoration: none;
}

/* Change the link color to #111 (black) on hover */
li a:hover {
    background-color: #111;
}

```

Căn phải các liên kết bằng cách thả nổi các mục danh sách sang bên phải (float:right);

```

<ul>
    <li><a href="#home">Home</a></li>
    <li><a href="#news">News</a></li>
    <li><a href="#contact">Contact</a></li>
    <li style="float:right"><a class="active" href="#about">About</a></li>
</ul>

```

Thêm thuộc tính border-right vào để tạo bộ chia liên kết:

```

/* Add a gray right border to all list items, except the last item
(last-child) */
li {
    border-right: 1px solid #bbb;
}

li:last-child {
    border-right: none;
}

```

Ví dụ về thanh điều hướng ngang màu xám có viền mỏng màu xám:

```

ul {
    border: 1px solid #e7e7e7;
    background-color: #f3f3f3;
}

li a {
    color: #666;
}

```

Thêm vị trí: dính; đến để tạo thanh điều hướng cố định.

Phần tử cố định chuyển đổi giữa tương đối và cố định, tùy thuộc vào vị trí cuộn. Nó được định vị tương đối cho đến khi gặp một vị trí bù nhất định trong khung nhìn - sau đó nó "cố định" tại chỗ (như vị trí: cố định).

```

ul {
    position: -webkit-sticky; /* Safari */
    position: sticky;
    top: 0;
}

```

c. Ví dụ minh họa

```
<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport"
content="width=device-width, initial-
scale=1.0">
  <title>Navbar CSS</title>
  <style>
    /* Tạo thanh điều hướng */
    .navbar {
      background-color: #333;
      overflow: hidden;
    }

    /* Kiểu cho các liên kết trong thanh điều
    hướng */
    .navbar a {
      float: left;
      display: block;
      color: white;
      text-align: center;
      padding: 14px 20px;
      text-decoration: none;
    }

    /* Hiệu ứng khi di chuột vào */
    .navbar a:hover {
      background-color: #4CAF50;
      color: white;
    }

    /* Kiểu cho mục đang active */
    .navbar a.active {
      background-color: #04AA6D;
    }
  </style>
</head>
<body>

  <h2>Thanh điều hướng CSS</h2>

  <!-- Thanh điều hướng -->
  <div class="navbar">
    <a class="active" href="#">Trang chủ</a>
    <a href="#">Giới thiệu</a>
    <a href="#">Dịch vụ</a>
    <a href="#">Liên hệ</a>
  </div>

</body>
</html>
```



14. Thủ CSS Image Gallery

a. Chức năng

CSS có thể được sử dụng để tạo một thư viện hình ảnh.

b. Cú pháp

Thư viện hình ảnh sau đây được tạo bằng CSS:

```
<html>
<head>
<style>
div.gallery {
  margin: 5px;
  border: 1px solid #ccc;
  float: left;
  width: 180px;
}

div.gallery:hover {
  border: 1px solid #777;
}

div.gallery img {
  width: 100%;
  height: auto;
}

div.desc {
  padding: 15px;
  text-align: center;
}
</style>
</head>
<body>

<div class="gallery">
  <a target="_blank" href="img_5terre.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="img_forest.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="img_lights.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>
```

```

</a>
<div class="desc">Add a description of the image here</div>
</div>

<div class="gallery">
  <a target="_blank" href="img_mountains.jpg">
    
  </a>
  <div class="desc">Add a description of the image here</div>
</div>

</body>
</html>

```

c. Ví dụ minh họa

```

<!DOCTYPE html>
<html lang="vi">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Thư viện ảnh CSS</title>
  <style>
    /* Container chính */
    .gallery {
      display: flex;
      flex-wrap: wrap;
      justify-content: center;
      gap: 10px;
    }

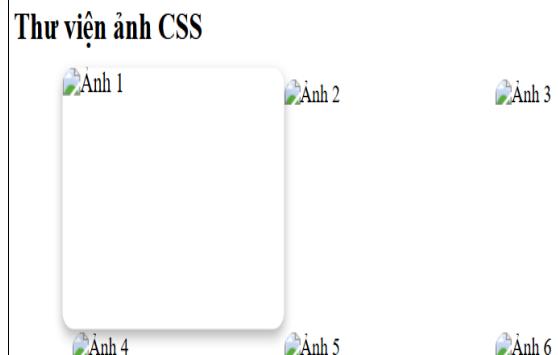
    /* Khung ảnh */
    .gallery img {
      width: 200px;
      height: 150px;
      border-radius: 10px;
      transition: transform 0.3s, box-shadow 0.3s;
    }

    /* Hiệu ứng khi di chuột vào ảnh */
    .gallery img:hover {
      transform: scale(1.1);
      box-shadow: 0 4px 8px rgba(0, 0, 0, 0.3);
    }
  </style>
</head>
<body>

  <h2>Thư viện ảnh CSS</h2>

  <!-- Thư viện ảnh -->
  <div class="gallery">

```



```







</div>

</body>
</html>

```

15. Thẻ CSS Website Layout

a. Chức năng

Một trang web thường được chia thành phần tiêu đề, menu, nội dung và chân trang.

Có rất nhiều thiết kế bố cục khác nhau để lựa chọn. Tuy nhiên, cấu trúc trên là một trong những cấu trúc phổ biến nhất và chúng ta sẽ xem xét kỹ hơn trong hướng dẫn này.

b. Cú pháp

- Tiêu đề

Tiêu đề thường nằm ở đầu trang web (hoặc ngay bên dưới menu điều hướng trên cùng). Nó thường chứa logo hoặc tên trang web:

```
.header {
background-color: #F1F1F1;
text-align: center;
padding: 20px;
}
```

- Thanh điều hướng

Thanh điều hướng chứa danh sách các liên kết giúp khách truy cập điều hướng qua trang web của bạn:

```
/* The navbar container */
.topnav {
overflow: hidden;
background-color: #333;
}
```

```

/* Navbar links */

.topnav a {
    float: left;
    display: block;
    color: #f2f2f2;
    text-align: center;
    padding: 14px 16px;
    text-decoration: none;
}

/* Links - change color on hover */

.topnav a:hover {
    background-color: #ddd;
    color: black;
}

```

- Nội dung

Bố cục trong phần này thường phụ thuộc vào người dùng mục tiêu. Bố cục phổ biến nhất là một (hoặc kết hợp chúng) trong số các bố cục sau:

- + cột (thường được sử dụng cho trình duyệt di động)
- + cột (thường được sử dụng cho máy tính bảng và máy tính xách tay)
- Bố cục 3 cột (chỉ được sử dụng cho máy tính để bàn)



Chúng tôi sẽ tạo bố cục 3 cột và thay đổi thành bố cục 1 cột trên màn hình nhỏ hơn:

```

/* Create three equal columns that float next to each other */

.column {
    float: left;
    width: 33.33%;
}

```

```

/* Clear floats after the columns */

.row:after {
    content: "";
    display: table;
    clear: both;
}

/* Responsive layout - makes the three columns stack on top of each other
instead of next to each other on smaller screens (600px wide or less) */

@media screen and (max-width: 600px) {
    .column {
        width: 100%;
    }
}

```

- Các cột không bằng nhau

Nội dung chính là phần lớn nhất và quan trọng nhất của trang web của bạn.

Thường có chiều rộng cột không bằng nhau, do đó hầu hết không gian được dành cho nội dung chính. Nội dung bên (nếu có) thường được sử dụng làm điều hướng thay thế hoặc để chỉ định thông tin có liên quan đến nội dung chính. Thay đổi chiều rộng theo ý muốn, chỉ cần nhớ rằng tổng chiều rộng phải bằng 100%:

```

.column {
    float: left;
}

/* Left and right column */

.column.side {
    width: 25%;
}

/* Middle column */

.column.middle {
    width: 50%;
}

/* Responsive layout - makes the three columns stack on top of each other */

```

```

instead of next to each other */

@media screen and (max-width: 600px) {
  .column.side, .column.middle {
    width: 100%;
  }
}

```

- Chân trang

Chân trang được đặt ở cuối trang của bạn. Nó thường chứa thông tin như bản quyền và thông tin liên hệ:

```

.footer {
  background-color: #F1F1F1;
  text-align: center;
  padding: 10px;
}

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre> <!DOCTYPE html> <html lang="en"> <head> <title>CSS Website Layout</title> <meta charset="utf-8"> <meta name="viewport" content="width=device- width, initial-scale=1"> <style> body { margin: 0; } /* Style the header */ .header { background-color: #f1f1f1; padding: 20px; } </pre>	 <p style="text-align: center;">Tiêu đề</p>

```
    text-align: center;  
}  
</style>  
</head>  
<body>  
  
<div class="header">  
    <h1>Tiêu đề</h1>  
</div>  
  
</body>  
</html>
```

16. Thẻ CSS Rounded Corners

a. Chức năng

Với thuộc tính border-radius của CSS, bạn có thể làm cho bất kỳ phần tử nào có "góc bo tròn".

b. Cú pháp

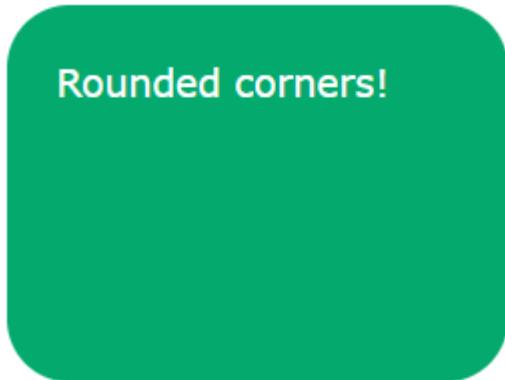
- Thuộc tính border-radius của CSS

Thuộc tính border-radius của CSS xác định bán kính của các góc của một phần tử.

Mẹo: Thuộc tính này cho phép bạn thêm các góc bo tròn vào các phần tử!

Sau đây là ba ví dụ:

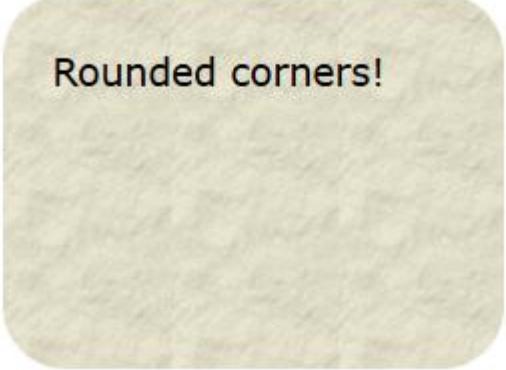
+ Các góc bo tròn cho một phần tử có màu nền được chỉ định:



+ Các góc bo tròn cho phần tử có đường viền:

Rounded corners!

- + Các góc bo tròn cho phần tử có hình ảnh nền:



Rounded corners!

```
#rcorners1 {  
    border-radius: 25px;  
    background: #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}  
  
#rcorners2 {  
    border-radius: 25px;  
    border: 2px solid #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}  
  
#rcorners3 {  
    border-radius: 25px;  
    background: url(paper.gif);
```

```
background-position: left top;  
background-repeat: repeat;  
padding: 20px;  
width: 200px;  
height: 150px;  
}
```

Mèo: Thuộc tính border-radius thực chất là thuộc tính viết tắt của các thuộc tính border-top-left-radius, border-top-right-radius, border-bottom-right-radius và border-bottom-left-radius.

- CSS border-radius - Chỉ định từng góc

Thuộc tính border-radius có thể có từ một đến bốn giá trị. Sau đây là các quy tắc:

+ Bốn giá trị - border-radius: 15px 50px 30px 5px; (giá trị đầu tiên áp dụng cho góc trên cùng bên trái, giá trị thứ hai áp dụng cho góc trên cùng bên phải, giá trị thứ ba áp dụng cho góc dưới cùng bên phải và giá trị thứ tư áp dụng cho góc dưới cùng bên trái):



+ Ba giá trị - border-radius: 15px 50px 30px; (giá trị đầu tiên áp dụng cho góc trên cùng bên trái, giá trị thứ hai áp dụng cho góc trên cùng bên phải và góc dưới cùng bên trái, và giá trị thứ ba áp dụng cho góc dưới cùng bên phải):



+ Hai giá trị - border-radius: 15px 50px; (giá trị đầu tiên áp dụng cho góc trên bên trái và góc dưới bên phải, và giá trị thứ hai áp dụng cho góc trên bên phải và góc dưới bên trái):



+ Một giá trị - border-radius: 15px; (giá trị áp dụng cho cả bốn góc được bo tròn như nhau:



```
#rcorners1 {  
    border-radius: 15px 50px 30px 5px;  
    background: #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}  
  
#rcorners2 {  
    border-radius: 15px 50px 30px;  
    background: #73AD21;  
    padding: 20px;  
    width: 200px;  
    height: 150px;  
}  
  
#rcorners3 {  
    border-radius: 15px 50px;  
    background: #73AD21;
```

```

padding: 20px;
width: 200px;
height: 150px;
}

#rcorners4 {
border-radius: 15px;
background: #73AD21;
padding: 20px;
width: 200px;
height: 150px;
}

```

c. Ví dụ minh họa

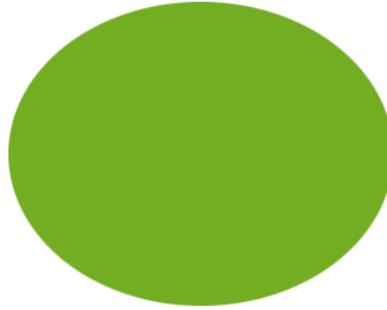
Code mẫu	Kết quả
#rcorners1 { border-radius: 50px / 15px; background: #73AD21; padding: 20px; width: 200px; height: 150px; }	The border-radius Property Elliptical border - border-radius: 50px / 15px: 
#rcorners2 { border-radius: 15px / 50px; background: #73AD21; padding: 20px; width: 200px; height: 150px; }	
#rcorners3 { border-radius: 50%; background: #73AD21; padding: 20px;	

```
width: 200px;  
height: 150px;  
}
```

Elliptical border - border-radius: 15px / 50px:



Ellipse border - border-radius: 50%:



17. Thủ CSS Multiple

a. Chức năng

CSS Multiple Selector (Bộ chọn CSS nhiều phần tử) được sử dụng để áp dụng cùng một kiểu cho nhiều phần tử HTML khác nhau mà không cần viết lại nhiều lần các dòng CSS.

b. Cú pháp

```
selector1, selector2, selector3 {  
    property: value;  
}
```

Các bộ chọn được ngăn cách bởi dấu phẩy ,

Mỗi bộ chọn đều có thể là một phần tử HTML, một class, một id hoặc một tổ hợp bộ chọn phức tạp.

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <head> <style> /* Ví dụ về Multiple Selector */ h1, h2, p {</pre>	

```

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

    .box, #mainContent {
        border: 2px solid red;
        padding: 10px;
        margin: 5px;
    }

```

</style>

</head>

<body>

<h1>Tiêu đề cấp 1</h1>

<h2>Tiêu đề cấp 2</h2>

<p>Đây là một đoạn văn được định dạng bởi Multiple Selector.</p>

<div class="box">Div với class "box"</div>

<div id="mainContent">Div với id "mainContent"</div>

</body>

</html>

Tiêu đề cấp 1

Tiêu đề cấp 2

Đây là một đoạn văn được định dạng bởi Multiple Selector.

Div với class "box"

Div với id "mainContent"

18. Thẻ CSS Buttons

a. Chức năng

- CSS Buttons được sử dụng để tạo và tùy chỉnh các nút (buttons) trên trang web. Với CSS, bạn có thể định dạng và thay đổi giao diện nút như:

- + Màu nền, màu chữ.
- + Kích thước, đường viền.
- + Hiệu ứng hover (khi rê chuột qua).

- + Hiệu ứng active (khi nhấn giữ).
- + Hiệu ứng transition (chuyển động mượt mà).
- + Bo tròn góc, đổ bóng, hoạt ảnh, v.v.

b. Cú pháp

```
button {
    background-color: value;
    color: value;
    padding: value;
    border: value;
    border-radius: value;
    transition: value;
}
```

- Button colors: Sử dụng thuộc tính background-color để thay đổi màu nền của nút

```
.button1 {background-color: #04AA6D;} /* Green */
.button2 {background-color: #008CBA;} /* Blue */
.button3 {background-color: #f44336;} /* Red */
.button4 {background-color: #e7e7e7; color: black;} /* Gray */
.button5 {background-color: #555555;} /* Black */
```

- Button sizes: sử dụng thuộc tính font-size để thay đổi kích thước phông chữ của nút

```
.button1 {font-size: 10px;}
.button2 {font-size: 12px;}
.button3 {font-size: 16px;}
.button4 {font-size: 20px;}
.button5 {font-size: 24px;}
```

- Sử dụng thuộc tính padding để thay đổi phần đệm của nút:

```
.button1 {padding: 10px 24px;}
.button2 {padding: 12px 28px;}
.button3 {padding: 14px 40px;}
.button4 {padding: 32px 16px;}
.button5 {padding: 16px;}
```

- Rounded buttons: Sử dụng thuộc tính border-radius để thêm các góc bo tròn vào nút

```
.button1 {border-radius: 2px;}
.button2 {border-radius: 4px;}
.button3 {border-radius: 8px;}
```

```
.button4 {border-radius: 12px;}  
.button5 {border-radius: 50%;}
```

- Colored button borders: Sử dụng thuộc tính border để thêm đường viền màu vào nút

```
.button1 {  
background-color: white;  
color: black;  
border: 2px solid #04AA6D; /* Green */  
}
```

- Hoverable buttons: Sử dụng bộ chọn :hover để thay đổi kiểu của nút khi bạn di chuyển chuột qua nút đó.

Mẹo: Sử dụng thuộc tính transition-duration để xác định tốc độ của hiệu ứng "hover":

```
.button {  
transition-duration: 0.4s;  
}  
  
.button:hover {  
background-color: #04AA6D; /* Green */  
color: white;  
}
```

- Shadow buttons: Sử dụng thuộc tính box-shadow để thêm bóng cho nút

```
.button1 {  
box-shadow: 0 8px 16px 0 rgba(0,0,0,0.2), 0 6px 20px 0 rgba(0,0,0,0.19);  
}  
  
.button2:hover {  
box-shadow: 0 12px 16px 0 rgba(0,0,0,0.24), 0 17px 50px 0 rgba(0,0,0,0.19);  
}
```

- Disabled buttons: Sử dụng thuộc tính opacity để thêm độ trong suốt cho một nút (tạo giao diện "vô hiệu hóa").

Mẹo: Bạn cũng có thể thêm thuộc tính con trỏ với giá trị "not-allowed", sẽ hiển thị "biển báo cấm đỗ xe" khi bạn di chuột qua nút:

```
.disabled {  
opacity: 0.6;  
cursor: not-allowed;  
}
```

- Button width: Theo mặc định, kích thước của nút được xác định bởi nội dung văn bản của nó (rộng bằng nội dung của nó). Sử dụng thuộc tính width để thay đổi chiều rộng của nút:

```
.button1 {width: 250px;}  
.button2 {width: 50%;}  
.button3 {width: 100%;}
```

- Button groups: Xóa lè và thêm float:left vào mỗi nút để tạo một nhóm nút:

```
.button {  
    float: left;  
}
```

- Bordered button group: Sử dụng thuộc tính border để tạo nhóm nút có viền:

```
.button {  
    float: left;  
    border: 1px solid green;  
}
```

- Vertical button group: Sử dụng display:block thay vì float:left để nhóm các nút bên dưới nhau, thay vì đặt cạnh nhau:

```
.button {  
    display: block;  
}
```

- Button on image

- Animated buttons

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <head> <style> .button { background-color: #04AA6D; border: none; color: white; padding: 15px 32px; text-align: center; text-decoration: none; display: inline-block; font-size: 16px; margin: 4px 2px; }</pre>	<p>CSS Buttons</p> 

```
        cursor: pointer;  
    }  
  
</style>  
</head>  
<body>  
  
<h2>CSS Buttons</h2>  
  
<button>Default Button</button>  
<a href="#" class="button">Link  
Button</a>  
<button  
class="button">Button</button>  
<input type="button" class="button"  
value="Input Button">  
  
</body>  
</html>
```

19. Thẻ CSS object-position

a. Chức năng

Thuộc tính object-position của CSS được sử dụng để chỉ định cách định vị `` hoặc `<video>` trong vùng chứa của nó.

b. Cú pháp

- Hình ảnh

+ Hãy xem hình ảnh sau đây từ Paris, có kích thước 400x300 pixel:



+ Tiếp theo, chúng ta sử dụng object-fit: cover; để giữ nguyên tỷ lệ khung hình và lấp đầy kích thước đã cho. Tuy nhiên, hình ảnh sẽ được cắt bớt để vừa vặn, như thế này:



```
img {  
    width: 200px;  
    height: 300px;
```

```
object-fit: cover;  
}
```

- Sử dụng Thuộc tính object-position
- + Giả sử phần hình ảnh được hiển thị không được định vị như chúng ta muốn. Để định vị hình ảnh, chúng ta sẽ sử dụng thuộc tính object-position.
- + Ở đây, chúng ta sẽ sử dụng thuộc tính object-position để định vị hình ảnh sao cho tòa nhà cổ lớn nằm ở giữa:



```
img {  
    width: 200px;  
    height: 300px;  
    object-fit: cover;  
    object-position: 80% 100%;  
}
```

- + Ở đây chúng ta sẽ sử dụng thuộc tính object-position để định vị hình ảnh sao cho Tháp Eiffel nổi tiếng nằm ở giữa:



```
img {  
    width: 200px;  
    height: 300px;  
    object-fit: cover;  
    object-position: 15% 100%;  
}
```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre><!DOCTYPE html> <html> <head> <style> img { width: 200px; height: 300px; object-fit: cover; object-position: 80% 100%; } </style></pre>	<p>Using object-position</p> <p>Here we will use the object-position property to position the image so that the great old building is in center:</p>

```

</head>

<body>

<h2>Using object-position</h2>

<p>Here we will use the object-position
property to position the image so that the
great old building is in center:</p>



</body>
</html>

```



20. Thủ CSS Multiple Columns

a. Chức năng

- Bố cục nhiều cột CSS

+ Bố cục nhiều cột CSS cho phép định nghĩa dễ dàng nhiều cột văn bản - giống như trong báo:

Daily Ping

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci

tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui

blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

- Thuộc tính nhiều cột CSS

- Trong chương này, bạn sẽ tìm hiểu về các thuộc tính nhiều cột sau:

+ column-count

+ column-gap

+ column-rule-style

```
+ column-rule-width  
+ column-rule-color  
+ column-rule  
+ column-span  
+ column-width
```

b. Cú pháp

- CSS Tạo Nhiều Cột

Thuộc tính column-count chỉ định số cột mà một phần tử nên được chia thành.

Ví dụ sau sẽ chia văn bản trong phần tử <div> thành 3 cột:

```
div {  
    column-count: 3;  
}
```

- CSS Chỉ định Khoảng cách Giữa Các Cột

Thuộc tính column-gap chỉ định khoảng cách giữa các cột.

Ví dụ sau chỉ định khoảng cách 40 pixel giữa các cột:

```
div {  
    column-gap: 40px;  
}
```

- Quy tắc cột CSS

Thuộc tính column-rule-style chỉ định kiểu của quy tắc giữa các cột:

```
div {  
    column-rule-style: solid;  
}
```

- Thuộc tính column-rule-width chỉ định chiều rộng của quy tắc giữa các cột:

```
div {  
    column-rule-width: 1px;  
}
```

- Thuộc tính column-rule-color chỉ định màu của quy tắc giữa các cột:

```
div {  
    column-rule-color: lightblue;  
}
```

- Thuộc tính column-rule là thuộc tính viết tắt để thiết lập tất cả các thuộc tính column-rule-* ở trên.

Ví dụ sau thiết lập chiều rộng, kiểu và màu của quy tắc giữa các cột:

```

div {
    column-rule: 1px solid lightblue;
}

```

- Chỉ định số lượng cột mà một phần tử nên trải dài

Thuộc tính column-span chỉ định số lượng cột mà một phần tử nên trải dài.

Ví dụ sau đây chỉ định rằng phần tử <h2> nên trải dài trên tất cả các cột:

```

h2 {
    column-span: all;
}

```

- Chỉ định Chiều rộng Cột

Thuộc tính column-width chỉ định chiều rộng tối ưu được đề xuất cho các cột.

Ví dụ sau chỉ định rằng chiều rộng tối ưu được đề xuất cho các cột phải là 100px:

```

div {
    column-width: 100px;
}

```

c. Ví dụ minh họa

Code mẫu	Kết quả
<pre> <!DOCTYPE html> <html> <head> <style> .newspaper { column-count: 2; column-gap: 40px; column-rule: 1px solid lightblue; } h2 { column-span: all; } </style> </head> <body> <div class="newspaper"> <h2>TT IOTs</h2> Lorem ipsum dolor sit amet, consectetuer adipiscing elit, sed diam nonummy nibh euismod </pre>	<p>TT IOTs</p> <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed diam nonummy nibh euismod tincidunt ut laoreet dolore magna aliquam erat volutpat. Ut wisi enim ad minim veniam, quis nostrud exerci tation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.</p>

tincidunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamcorper suscipit lobortis nisl ut aliquip ex ea commodo consequat. Duis autem vel eum iriure dolor in hendrerit in vulputate velit esse molestie consequat, vel illum dolore eu feugiat nulla facilisis at vero eros et accumsan et iusto odio dignissim qui blandit praesent luptatum zzril delenit augue duis dolore te feugait nulla facilisi. Nam liber tempor cum soluta nobis eleifend option congue nihil imperdiet doming id quod mazim placerat facer possim assum.

</div>

</body>

</html>

B. CƠ SỞ DỮ LIỆU FIREBASE

1. Giới thiệu về Firebase

Trong quá trình phát triển các ứng dụng web và mobile hiện đại, việc xây dựng backend (máy chủ, cơ sở dữ liệu, xác thực, lưu trữ, xử lý server-side...) là một trong những khía cạnh quan trọng nhưng cũng tốn kém và phức tạp. Để đơn giản hóa quy trình này, nhiều nền tảng Backend-as-a-Service (BaaS) đã ra đời, trong đó Firebase là một trong những nền tảng phổ biến nhất do Google cung cấp.

Firebase ban đầu là một công ty độc lập ra mắt vào năm 2011, chuyên cung cấp các dịch vụ thời gian thực cho ứng dụng. Đến năm 2014, Firebase được Google mua lại và phát triển thành một nền tảng toàn diện hỗ trợ nhiều dịch vụ backend cần thiết cho cả ứng dụng web lẫn mobile. Firebase giúp các lập trình viên tiết kiệm thời gian phát triển, dễ dàng tích hợp các tính năng quan trọng như lưu trữ dữ liệu, xác thực người dùng, gửi thông báo đẩy, phân tích hành vi người dùng và nhiều hơn nữa, tất cả đều thông qua các SDK và API mạnh mẽ do Google duy trì.

Firebase hướng tới việc cung cấp một hệ sinh thái đồng bộ và mở rộng, giúp cả những lập trình viên cá nhân cho đến các doanh nghiệp khởi nghiệp có thể dễ dàng xây dựng ứng dụng chất lượng cao, có thể mở rộng và phát triển trong thời gian ngắn.



Firebase

Logo Firebase

2. Các thành phần chính trong hệ sinh thái Firebase

Firebase cung cấp nhiều dịch vụ backend phục vụ cho việc xây dựng ứng dụng toàn diện. Dưới đây là mô tả chi tiết về các thành phần cốt lõi trong nền tảng này:

a. Firebase Realtime Database

Firebase Realtime Database là dịch vụ cơ sở dữ liệu NoSQL đầu tiên của Firebase, cho phép lưu trữ và đồng bộ dữ liệu theo thời gian thực giữa nhiều thiết bị client. Dữ liệu được tổ chức dưới dạng cây JSON, trong đó mỗi nút có thể được truy cập trực tiếp hoặc lắng nghe để theo dõi các thay đổi một cách tức thì. Khi một client

thực hiện thay đổi dữ liệu (ví dụ như gửi tin nhắn), các thay đổi đó sẽ ngay lập tức được cập nhật đến tất cả các client khác đang kết nối và lắng nghe cùng một phần dữ liệu. Nhờ khả năng đồng bộ nhanh chóng này, Firebase Realtime Database đặc biệt phù hợp cho các ứng dụng cần cập nhật liên tục như ứng dụng chat, hệ thống theo dõi đơn hàng, hoặc bảng điều khiển trực tuyến.

b. Cloud Firestore

Cloud Firestore là phiên bản nâng cấp của Realtime Database, cũng thuộc nhóm cơ sở dữ liệu NoSQL của Firebase. Khác với Realtime Database lưu trữ dữ liệu dưới dạng cây JSON, Firestore tổ chức dữ liệu theo cấu trúc document và collection, tương tự như cách hoạt động của MongoDB. Nhờ đó, Firestore hỗ trợ các truy vấn phức tạp hơn, cho phép phân quyền truy cập chi tiết và khả năng mở rộng cao hơn để đáp ứng nhu cầu của các ứng dụng quy mô lớn. Một điểm nổi bật của Firestore là tính năng offline-first – người dùng vẫn có thể thao tác và lưu trữ dữ liệu ngay cả khi thiết bị không có kết nối mạng, và các thay đổi sẽ được tự động đồng bộ hóa khi kết nối được khôi phục. Với những ưu điểm này, Firestore rất phù hợp cho các ứng dụng có cấu trúc dữ liệu rõ ràng, cần khả năng truy vấn linh hoạt và hiệu năng ổn định trong môi trường có kết nối không liên tục.

c. Firebase Authentication

Firebase Authentication là một dịch vụ xác thực người dùng mạnh mẽ, giúp lập trình viên dễ dàng tích hợp các phương thức đăng nhập phổ biến vào ứng dụng của mình. Dịch vụ này hỗ trợ đa dạng các hình thức đăng nhập như email/mật khẩu, tài khoản Google, Facebook, Twitter, Apple, số điện thoại thông qua SMS, và thậm chí cho phép sử dụng các provider tùy chỉnh để phù hợp với nhu cầu đặc thù của từng hệ thống. Bên cạnh đó, Firebase Authentication còn đi kèm với nhiều tính năng bảo mật quan trọng như xác minh email, xác thực đa yếu tố (MFA), và quản lý phiên đăng nhập nhằm nâng cao mức độ an toàn cho người dùng. Một ưu điểm lớn khác là khả năng tích hợp trực tiếp với các dịch vụ khác của Firebase như Realtime Database, Cloud Firestore và Cloud Storage, giúp kiểm soát và bảo vệ quyền truy cập dữ liệu hiệu quả hơn trong toàn bộ hệ sinh thái Firebase.

d. Firebase Cloud Storage

Firebase Storage là dịch vụ lưu trữ đám mây mạnh mẽ, được thiết kế để xử lý các loại tệp có dung lượng lớn như hình ảnh, video, âm thanh và tài liệu. Dữ liệu được lưu trữ trên nền tảng Google Cloud Storage, mang lại độ tin cậy cao, tốc độ truy xuất nhanh và đảm bảo an toàn trong quá trình truyền tải và lưu giữ. Firebase Storage đặc biệt phù hợp cho các ứng dụng cần chia sẻ hoặc quản lý nội dung đa phương tiện với số lượng lớn người dùng. Ngoài ra, dịch vụ này còn hỗ trợ kiểm soát truy cập chi tiết thông qua Firebase Authentication, cho phép

giới hạn quyền truy cập dựa trên danh tính người dùng, từ đó nâng cao tính bảo mật và cá nhân hóa trong việc quản lý dữ liệu.

e. Firebase Hosting

Firebase Hosting là dịch vụ lưu trữ dành riêng cho nội dung web tĩnh như các tệp HTML, CSS, JavaScript, hình ảnh và các tài nguyên frontend khác. Dịch vụ này cung cấp môi trường lưu trữ an toàn với hỗ trợ HTTPS mặc định, sử dụng mạng phân phối nội dung (CDN) toàn cầu để đảm bảo tốc độ tải trang nhanh và ổn định cho người dùng ở mọi khu vực. Quá trình triển khai trang web được thực hiện nhanh chóng và dễ dàng thông qua dòng lệnh, giúp rút ngắn thời gian phát hành sản phẩm. Firebase Hosting đặc biệt phù hợp cho các dự án như Single Page Application (SPA), trang landing page, portfolio cá nhân, hoặc bất kỳ trang web nào vận hành hoàn toàn ở phía client mà không cần backend phức tạp.

f. Firebase Cloud Functions

Firebase Cloud Functions là dịch vụ cho phép lập trình viên viết các đoạn mã xử lý backend mà không cần vận hành hoặc quản lý máy chủ, theo mô hình serverless. Các hàm này được viết bằng JavaScript hoặc TypeScript và chạy trên môi trường Node.js. Chúng được kích hoạt tự động bởi các sự kiện như thay đổi dữ liệu trong Firestore, người dùng mới đăng ký tài khoản, yêu cầu HTTP từ client, hoặc các sự kiện đến từ dịch vụ bên thứ ba. Điều này giúp tự động hóa nhiều quy trình phía sau ứng dụng một cách hiệu quả và linh hoạt. Một số ứng dụng tiêu biểu của Firebase Cloud Functions bao gồm: gửi email xác nhận sau khi người dùng đăng ký, xử lý đơn hàng, tạo thông báo đẩy, ghi log bảo mật, và thực hiện các thao tác xử lý dữ liệu tùy chỉnh khác.

g. Firebase Cloud Messaging (FCM)

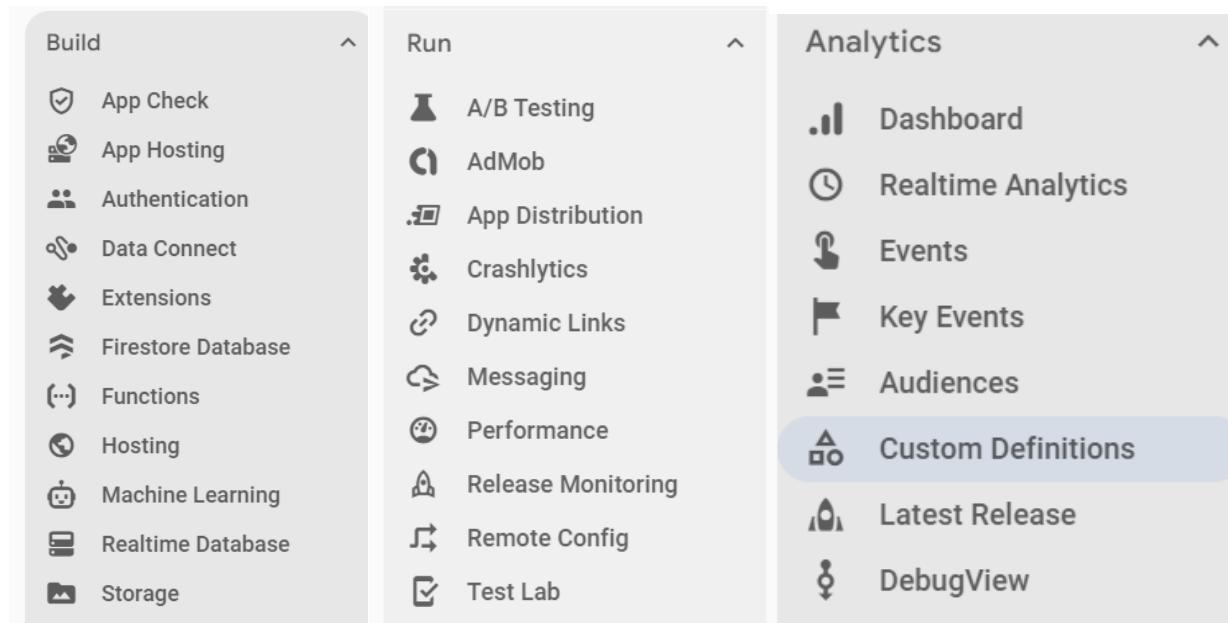
Firebase Cloud Messaging (FCM) là dịch vụ cho phép gửi thông báo đẩy đến các ứng dụng trên nền tảng Android, iOS cũng như các trình duyệt web. Với FCM, lập trình viên có thể dễ dàng gửi thông báo đến một người dùng cụ thể, một nhóm người dùng, hoặc toàn bộ hệ thống chỉ với vài thao tác cấu hình. Dịch vụ này hỗ trợ nhiều tính năng hữu ích như lập lịch gửi thông báo, phân phối thông báo theo múi giờ của người nhận, và cá nhân hóa nội dung để tăng mức độ tương tác. Nhờ khả năng hoạt động linh hoạt và hiệu quả, FCM là công cụ lý tưởng để duy trì kết nối với người dùng trong thời gian thực và nâng cao trải nghiệm ứng dụng.

h. Firebase Analytics (Google Analytics for Firebase)

Firebase Analytics (hay còn gọi là Google Analytics for Firebase) là một công cụ phân tích dữ liệu mạnh mẽ, được tích hợp chặt chẽ với các dịch vụ khác trong hệ sinh thái Firebase. Công cụ này cho phép lập trình viên và nhà phát triển theo dõi hành vi người dùng trong ứng dụng, đo lường hiệu quả của các chiến dịch marketing, đánh giá tỷ lệ giữ chân người dùng, cũng như ghi nhận các sự kiện tùy chỉnh theo nhu cầu cụ thể. Ngoài ra, Firebase Analytics có thể kết hợp với Firebase Remote Config để điều chỉnh giao diện hoặc chức năng của ứng

dụng dựa trên hành vi và đặc điểm của từng nhóm người dùng, từ đó tối ưu hóa trải nghiệm sử dụng và tăng hiệu quả tương tác.

Hình ảnh bên dưới là các tính năng của Firebase:



3. Ưu điểm và nhược điểm của Firebase

a. Ưu điểm

Triển khai nhanh: Firebase giúp rút ngắn đáng kể thời gian phát triển ứng dụng nhờ cung cấp sẵn nhiều công cụ và dịch vụ tiện ích. Lập trình viên có thể nhanh chóng xây dựng, thử nghiệm và triển khai các tính năng mà không cần cấu hình phức tạp về hạ tầng.

Dễ học, dễ tích hợp: Firebase cung cấp các bộ SDK có tài liệu hướng dẫn chi tiết, dễ tiếp cận cho cả người mới bắt đầu lẫn lập trình viên có kinh nghiệm. Các SDK này hỗ trợ tích hợp mượt mà trên nhiều nền tảng như Android, iOS và Web, giúp quá trình phát triển trở nên đơn giản và nhất quán.

Khả năng realtime mạnh mẽ: Các dịch vụ Realtime Database và Cloud Firestore của Firebase hỗ trợ khả năng cập nhật dữ liệu theo thời gian thực giữa các thiết bị client. Điều này đặc biệt hữu ích với những ứng dụng yêu cầu tính tương tác cao như chat, bảng điều khiển trực tuyến, hoặc theo dõi trạng thái người dùng.

Gói miễn phí hấp dẫn: Firebase cung cấp một gói miễn phí với đầy đủ các tính năng cơ bản, rất phù hợp cho các dự án cá nhân, ứng dụng nhỏ hoặc sản phẩm mẫu (MVP). Đây là lựa chọn lý tưởng cho sinh viên và các startup đang trong giai đoạn thử nghiệm, giúp tiết kiệm đáng kể chi phí phát triển ban đầu.

b. Nhược điểm

Chi phí tăng nhanh: Khi số lượng người dùng tăng lên hoặc dữ liệu lưu trữ ngày càng lớn, chi phí sử dụng Firebase có thể tăng nhanh và vượt mức ngân sách cho phép. Điều này đặc biệt xảy ra với dịch vụ Realtime Database, do tính chất đồng bộ dữ liệu liên tục và lưu lượng truy cập cao.

Giới hạn khả năng tùy biến backend: Firebase không phải là lựa chọn phù hợp cho các ứng dụng có logic xử lý backend phức tạp hoặc yêu cầu tính toán chuyên sâu, chẳng hạn như các hệ thống phân tích AI hoặc xử lý dữ liệu thời gian thực đòi hỏi hiệu năng cao. Các giới hạn này có thể cần拓展 khả năng mở rộng và phát triển ứng dụng theo nhu cầu chuyên sâu.

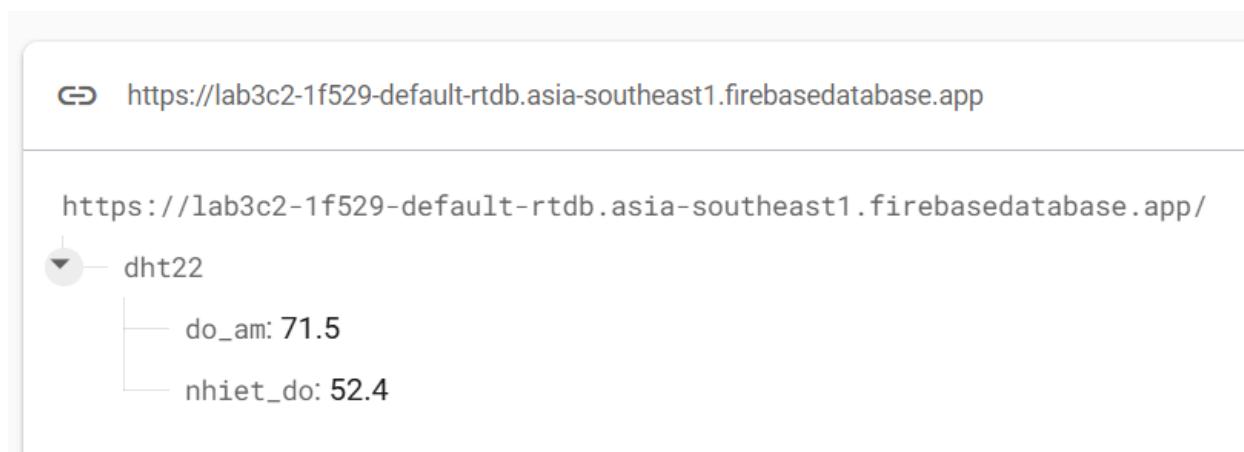
Phụ thuộc vào hệ sinh thái Google: Việc sử dụng Firebase rộng rãi trong dự án có thể khiến nhà phát triển bị “lock-in” vào hệ sinh thái của Google. Khi dự án mở rộng quy mô hoặc cần thay đổi chiến lược công nghệ, việc di chuyển sang nền tảng khác có thể gặp nhiều khó khăn và chi phí cao.

4. Ứng dụng của Firebase

Firebase hiện nay được ứng dụng rộng rãi trong cả môi trường học thuật lẫn thực tiễn công nghiệp nhờ tính linh hoạt và khả năng triển khai nhanh chóng. Trong lĩnh vực phát triển ứng dụng di động (Android/iOS), Firebase đặc biệt phù hợp với các ứng dụng có tính năng thời gian thực như trò chuyện, gửi thông báo hay

đồng bộ hóa dữ liệu người dùng. Đối với các ứng dụng web cần xử lý dữ liệu theo thời gian thực, như các hệ thống giám sát vận chuyển hay theo dõi thiết bị IoT, Firebase cung cấp giải pháp hiệu quả để đồng bộ và hiển thị dữ liệu trực tiếp. Ngoài ra, Firebase còn là lựa chọn lý tưởng cho các sản phẩm MVP hoặc startup nhờ khả năng triển khai nhanh mà không cần đầu tư hạ tầng phức tạp. Trong môi trường học thuật, sinh viên thường sử dụng Firebase cho các dự án môn học, đồ án tốt nghiệp hoặc các ứng dụng cá nhân, nhờ tài liệu đầy đủ và tính dễ sử dụng.

Hình ảnh bên dưới là ví dụ về liên kết Firebase Realtime Database cập nhật dữ liệu từ cảm biến



5. Kết luận

Firebase là một nền tảng backend mạnh mẽ, linh hoạt và dễ sử dụng, được thiết kế để hỗ trợ quá trình phát triển ứng dụng một cách nhanh chóng và hiệu quả — điều đặc biệt quan trọng trong bối cảnh công nghệ thay đổi liên tục như hiện nay. Với các dịch vụ tích hợp sẵn như cơ sở dữ liệu thời gian thực (Realtime Database), Firestore, xác thực người dùng (Firebase Authentication), lưu trữ tệp (Cloud Storage), và hệ thống gửi thông báo (Cloud Messaging), Firebase giúp rút ngắn đáng kể thời gian xây dựng và triển khai ứng dụng.

Một trong những điểm mạnh nổi bật của Firebase là khả năng tích hợp mượt mà với các công nghệ frontend phổ biến như React, Angular, Flutter, hay các nền tảng di động Android và iOS. Ngoài ra, việc được Google duy trì và phát triển không chỉ đảm bảo tính ổn định, mà còn giúp hệ sinh thái của Firebase liên tục được cập nhật và mở rộng với các công cụ hỗ trợ mạnh mẽ như Google Analytics, App Distribution, hay Machine Learning Kit.

Firebase là lựa chọn lý tưởng cho các lập trình viên ở mọi cấp độ, từ người mới bắt đầu đến những nhà phát triển chuyên nghiệp, đặc biệt là trong các dự án cần phản hồi thời gian thực hoặc cần triển khai sản phẩm thử nghiệm (MVP) nhanh chóng. Tuy nhiên, khi ứng dụng trong các hệ thống lớn hơn hoặc có yêu cầu xử lý backend phức tạp, một số hạn chế của Firebase bắt đầu bộc lộ rõ. Cụ thể, mô hình tính phí dựa trên mức sử dụng (usage-based pricing) có thể dẫn đến chi phí cao nếu không được tối ưu hợp lý. Đồng thời, việc phụ thuộc

quá nhiều vào hạ tầng của Firebase cũng có thể gây khó khăn trong việc di chuyển sang nền tảng khác (vendor lock-in) khi cần mở rộng quy mô hoặc tùy chỉnh sâu về backend.

Do đó, trước khi lựa chọn Firebase cho các dự án lớn hoặc dài hạn, các nhà phát triển nên đánh giá kỹ lưỡng về nhu cầu thực tế, khả năng mở rộng, chi phí vận hành, cũng như chiến lược phát triển lâu dài để đảm bảo sự phù hợp và bền vững.

C. THIẾT KẾ HỆ THỐNG

3.1 Yêu cầu hệ thống

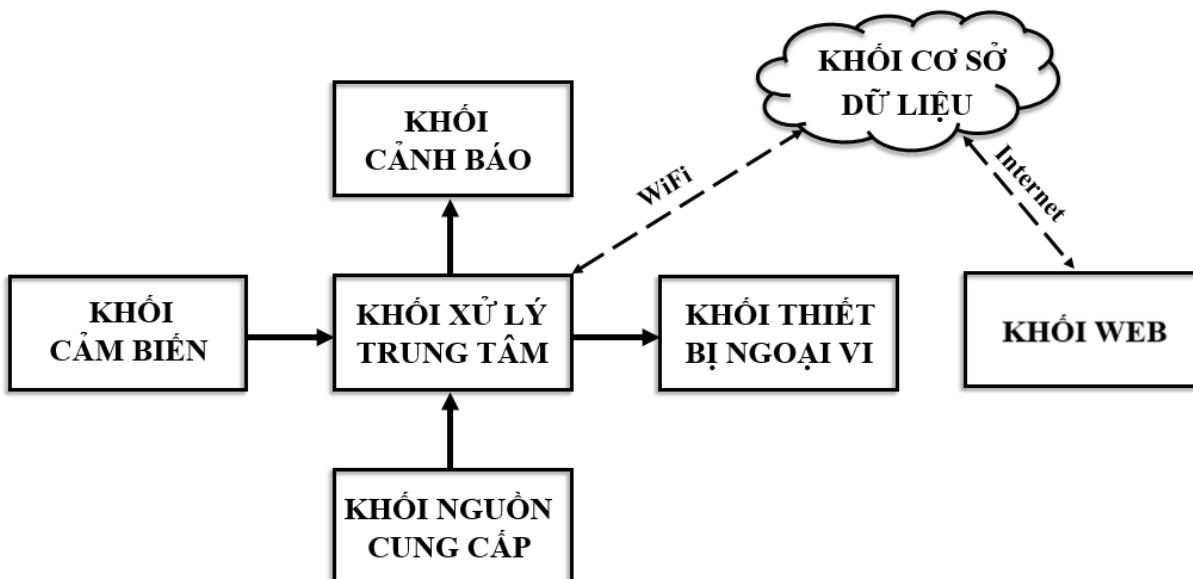
Phải đảm bảo được một số yêu cầu sau đây:

- **Thứ nhất**, hệ thống thu thập các thông số dữ liệu về môi trường từ các cảm biến như: nhiệt độ, độ ẩm, khí gas,... Tất cả các thông số được cập nhật liên tục lên cơ sở dữ liệu Firebase khi có sự thay đổi.
- **Thứ hai**, hệ thống có khả năng theo dõi các thông số về thời tiết tại thành phố HCM như biểu đồ lượng mưa, biểu đồ nhiệt độ, biểu đồ độ ẩm, cảnh báo khi quá ngưỡng và hiển thị các quận khác nhau :quận 1, thủ đức, bình thuận, quận 7. Đồng thời đo các thông số nhiệt độ, độ ẩm, khí gas trong các phòng bếp, phòng khách, phòng ngủ trong nhà. Ở đây nhóm sẽ thiết kế đặt cảm biến ở phòng khách. Tất cả sẽ được hiển thị trên giao diện web mà nhóm xây dựng.
- **Thứ ba**, hệ thống điều khiển được các thiết bị ngoại vi trong nhà như tivi, máy hút bụi, quạt trong phòng khách, bếp ga, lò nướng, tủ lạnh trong phòng ăn và máy lạnh, đèn, đồng hồ báo thức trong phòng ngủ thông qua các nút nhấn trên giao diện điều khiển.

3.2 Đặc tả hệ thống

3.2.1 Sơ đồ khái

Dựa vào yêu cầu thiết kế hệ thống đã đưa ra, nhóm đã thiết kế sơ đồ khái của hệ thống như Hình 3.2.1 bên dưới.



Hình 3.2. 1: Sơ đồ khái

Sơ đồ khái của hệ thống bao gồm 7 khái chính.

- + **Khái cảm biến:** Khái cảm biến sẽ thu thập các thông tin về môi trường như nhiệt độ, độ ẩm không khí và nồng độ khí gas để gửi về cho khái xử lý.
- + **Khái thiết bị ngoại vi:** Bao gồm các thiết bị ngoại vi trong nhà như tivi, quạt và máy hút bụi. Các thiết bị này được điều khiển thông qua khái xử lý trung tâm.
- + **Khái xử lý trung tâm:** Khái xử lý trung tâm sử dụng phần cứng ESP32, đóng vai trò điều phối toàn bộ hoạt động của hệ thống từ việc điều khiển đến giao tiếp Server.
- + **Khái cảnh báo:** Bao gồm một buzzer và một led sẽ nhấp nháy và hú khi quá ngưỡng.
- + **Khái nguồn:** Cung cấp năng lượng của toàn bộ hệ thống hoạt động, nguồn điện được lấy từ USB và cung cấp cho khái xử lý trung tâm. Các khái còn lại sử dụng nguồn điện trực tiếp từ các chân điện áp 5V có sẵn trên ESP32.
- + **Khái cơ sở dữ liệu:** Khái cơ sở dữ liệu sử dụng dịch vụ Realtime Database được hỗ trợ bởi Firebase, cung cấp một kho lưu trữ dữ liệu theo thời gian thực và tổ chức dưới dạng thư mục. Khái này sẽ làm cầu nối trung gian giữa khái giao diện Web/App và khái xử lý trung tâm.
- + **Khái Web:** Các giao diện điều khiển và quản lý được hiển thị trên cả giao diện Web người dùng thông qua dữ liệu từ khái cơ sở dữ liệu Firebase.

3.2.2 Giải thích hoạt động của hệ thống

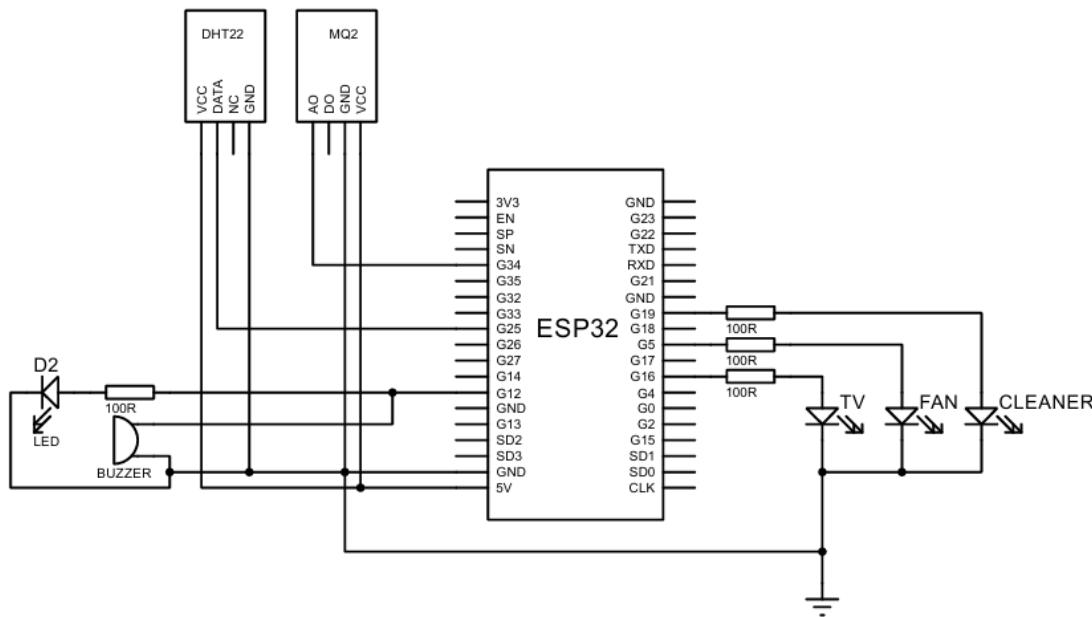
Hoạt động của hệ thống có thể bao gồm ba hoạt động chính: Thứ nhất là theo dõi và giám sát các thông số môi trường như nhiệt độ và độ ẩm không khí, nồng độ khí gas; Thứ hai là điều khiển các thiết bị ngoại vi thông qua giao diện người dùng trên Web giao tiếp với cơ sở dữ liệu. Thứ ba là hệ thống cảnh báo âm thanh kèm kẽm với led đơn nhấp nháy khi có quá ngưỡng xảy ra.

Với hoạt động thứ nhất, đầu tiên hệ thống sẽ tiến hành thu thập dữ liệu từ môi trường thông qua khái cảm biến, sau đó các thông số này được đưa đến khái xử lý trung tâm để tiến hành gửi dữ liệu lên cơ sở dữ liệu sau một khoảng thời gian cố định thông qua mạng WiFi. Tại đây các thông số từ cảm biến sẽ được lấy về để hiển thị trên giao diện người dùng thông qua Web một cách trực quan. Nếu có một hay nhiều thông số quá ngưỡng đã đặt ra thì khái cảnh báo bao gồm buzzer sẽ kêu và led nhấp nháy.

Với hoạt động thứ hai, hệ thống cho phép người dùng điều khiển các thiết bị ngoại vi thông qua các nút nhấn trên giao diện, khi có sự kiện nhấn nút xảy ra khái xử lý trung tâm sẽ tiến hành đọc giá trị các nút nhấn đó về và thực thi các công việc cần thiết. Hệ thống cho phép cài đặt thời gian tắt mở các thiết bị dựa trên thời gian thực của hệ thống.

3.3 Thiết kế hệ thống

Sau đây là sơ đồ nguyên lý của toàn hệ thống. Hệ thống được thiết kế bao gồm 3 cảm biến: cảm biến DHT22, cảm biến khí gas MQ-2 và khói cảnh báo bao gồm buzzer và led đơn. Ngoài ra, để minh họa cho bốn thiết bị ngoại vi, nhóm đã sử dụng các đèn LED thay thế.



Hình 3.3.1: Sơ đồ nguyên lý

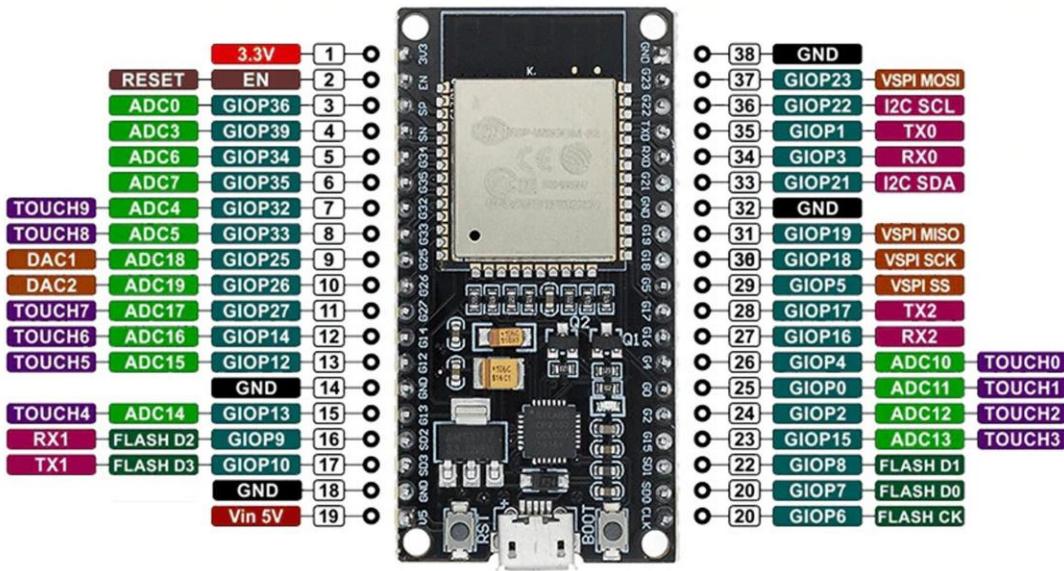
3.3.1 Thiết kế phần cứng

3.3.1.1 Khối xử lý trung tâm

Theo yêu cầu của đè tài, vi điều khiển được sử dụng phải có khả năng kết nối với WiFi. Trong trường hợp các vi điều khiển không tích hợp sẵn kết nối WiFi, cần sử dụng các module bổ sung như WiFi, 3G, 4G... để thực hiện kết nối với vi điều khiển. Tuy nhiên, việc này sẽ làm cho hệ thống trở nên phức tạp và tăng chi phí. Vì vậy, nhóm quyết định sử dụng vi điều khiển ESP32, có sẵn chuẩn kết nối WiFi bên trong module cùng với các chân I/O và các chuẩn truyền dữ liệu như SPI, UART, I2C.

Vi điều khiển ESP32, như được minh họa trong Hình 6.3.3, đã được nghiên cứu và phát triển bởi hãng Espressif Systems. Đây là dòng vi điều khiển giá rẻ dựa trên chip SoC, được phát triển từ vi điều khiển ESP8266. Sử dụng phần mềm Arduino IDE để lập trình, ESP32 rất phù hợp cho người mới tiếp cận với dòng vi điều khiển này. ESP32 hỗ trợ kết nối các chuẩn truyền thông không dây như WiFi, Bluetooth, cũng như mở rộng số lượng chân I/O so với ESP8266 với bộ chuyển đổi ADC 12 bits có 12 kênh và bộ chuyển đổi DAC 8 bits 2 kênh. Ngoài ra, ESP32 cũng hỗ trợ giao tiếp với các ngoại vi thông qua các chuẩn giao tiếp như UART, I2C, SPI, cùng với các kênh PWM để điều khiển động cơ. Thông số kỹ thuật của vi điều khiển ESP32 được mô tả trong Bảng 6.3.1.

PINOUT ESP32 38 PINES ESP WROOM 32



Hình 3.3.2: Sơ đồ chân ESP32

Bảng 3.3. 1: Bảng thông số kỹ thuật của ESP32

STT	Thông số	Giá trị
1	Điện áp hoạt động	3.3 – 5V DC
2	Dòng điện hoạt động	80 mA
3	Điện áp trên chân GPIO	3.3 V
4	Bộ nhớ Flash	4 MB
5	SRAM	128KB
6	Tốc độ xung clock	80 MHz/160 MHz
7	WiFi	802.11b/g/n WiFi transceiver
8	Giao tiếp máy tính	Cable Micro USB
9	Nhiệt độ hoạt động ổn định	-40°C đến 85°C

3.3.1.2 Khối cảm biến

a. Cảm biến nhiệt độ - độ ẩm DHT22

Nhiệt độ và độ ẩm là hai yếu tố quan trọng của môi trường, có tác động trực tiếp đến các hoạt động hàng ngày và cũng ảnh hưởng đến sức khỏe của con người. Trong thành phố Hồ Chí Minh, nhiệt độ thường dao động từ 20°C đến 45°C, và độ ẩm trung bình nằm trong khoảng từ 40% đến 90%. Để giám sát và điều chỉnh

được môi trường xung quanh, có nhiều loại cảm biến phổ biến hiện nay có khả năng đo và ghi lại thông tin về nhiệt độ và độ ẩm. Bảng 3.3.2 dưới đây liệt kê một số loại cảm biến nhiệt độ và độ ẩm được sử dụng rộng rãi:

Bảng 3.3. 2: Bảng thông số kỹ thuật của các cảm biến: DHT22, AM2305, SHT10

Thông số kỹ thuật	DHT22	AM2305	SHT10
Điện áp hoạt động	3.3 – 6V DC	3.3 – 5V DC	3.3 – 5.5V DC
Tầm đo nhiệt độ	-40°C - +80°C	-40°C - +80°C	-40°C - +125°C
Tầm đo độ ẩm	0% - 100%	0% - 100%	0% - 100%
Sai số nhiệt độ	± 0.5 °C	± 0.3 °C	± 0.5 °C
Sai số độ ẩm	± 5%	± 2%	± 4.5%
Giá cả	65.000 VNĐ	420.000 VNĐ	200.000 VNĐ

Dựa vào Bảng 3.3.2, cảm biến AM2305 và cảm biến SHT10 có sai số ít hơn so với cảm biến DHT22, tuy nhiên giá trị chênh lệch không quá lớn, ngoài ra hai cảm biến trên có giá thành quá cao so với DHT22, nên nhóm quyết định sử dụng DHT22 với thiết kế nhỏ gọn, giao tiếp với mọi dòng vi điều khiển, cũng như thông số cảm biến phù hợp với yêu cầu của đề tài.

Cảm biến DHT22 là một cảm biến kỹ thuật số đo độ ẩm và nhiệt độ môi trường. Cảm biến DHT22 có độ chính xác cao và thời gian lấy mẫu nhanh. DHT22 được sử dụng phổ biến trong việc giám sát môi trường, hệ thống điều khiển tự động và ứng dụng IoTs. Sơ đồ kết nối chân giữa cảm biến DHT22 và Board ESP32 được trình bày như Bảng 6.3.3 bên dưới.

Bảng 3.3. 3: Thứ tự kết nối chân giữa cảm biến DHT22 và Board ESP32 Wroom 32

DHT22	Board ESP32
VCC	5 V
DATA	IO25
GND	GND

b. Cảm biến khí gas MQ-2

Cảm biến khí gas MQ-2 được sử dụng để kiểm tra nồng độ khí gas (như LPG, butan, propane, methane, cồn, khói, và hydrogen) trong môi trường. Cảm biến hoạt động dựa trên nguyên lý thay đổi độ dẫn điện của các lớp vật liệu bán dẫn: khi nồng độ khí mục tiêu tăng lên, độ dẫn điện của cảm biến cũng tăng. Sự thay đổi này được chuyển đổi thành tín hiệu điện tại ngõ ra. MQ-2 có độ nhạy cao, khả năng phản hồi nhanh, tiêu thụ điện năng thấp, hoạt động ổn định, tuổi thọ cao và độ nhạy có thể điều chỉnh thông qua biến trở tích hợp.



Hình 3.3.3: Cảm biến khí gas MQ-2

Thông tin chân và thông số kỹ thuật của cảm biến MQ-2 được mô tả ở bảng 3.3.4 và bảng 3.3.5 bên dưới.

Bảng 3.3.4 Chức năng các chân của cảm biến MQ-2

STT	Tên chân	Chức năng
1	VCC	Chân cấp nguồn 5V DC
2	GND	Chân nối đất
3	DO	Tín hiệu số ngõ ra
4	AO	Tín hiệu tương tự ngõ ra

Bảng 3.3.5 Thông số kỹ thuật của cảm biến MQ-2

STT	Thông số kỹ thuật	Giá trị
1	Điện áp hoạt động	5V DC
2	Công suất tiêu thụ	Max 800 mW
3	Nhiệt độ hoạt động	-20°C – 50°C
4	Phạm vi đo	200 ppm đến 20000 ppm, tùy theo loại khí.
5	Loại đầu ra	Số/Tương tự

3.3.1.3 Khối cảnh báo

Khi một trong các thông số môi trường đạt đến giá trị ngưỡng cho phép, hệ thống sẽ tiến hành cảnh báo thông qua âm thanh từ còi Buzzer.

Do dòng hoạt động của Buzzer là 25mA mà dòng trên các chân I/O của ESP32 là 40mA nên đã đủ dòng hoạt động cho Buzzer nên không cần sử dụng thêm transistor đệm dòng.

Bảng 3.3.4: Thứ tự kết nối chân giữa khối cảnh báo và Board ESP32

Buzzer	ESP32 Board
VCC	IO12
GND	GND

3.3.1.3 Khối thiết bị ngoại vi

Đối với các thiết bị ngoại vi, thay vì sử dụng các thiết bị thực tế với công suất lớn thì trong đề tài này, nhóm sử dụng các đèn LED để minh họa cho các thiết bị với mục đích nghiên cứu và phát triển trước khi đưa vào sử dụng thực tế. Đối với LED, dòng điện định mức cho LED từ 10mA đến 20mA, điện áp rơi trên LED từ 1.8V đến 2.2V. Điện áp ngõ ra số của ESP32 mức cao là 3.3V. Từ đó, áp dụng định luật Ohm ta tính ra được giá trị điện trở hạn dòng cho LED là:

$$R = \frac{U}{I} = \frac{3.3V - 1.8V}{0.015A} = 100\Omega$$

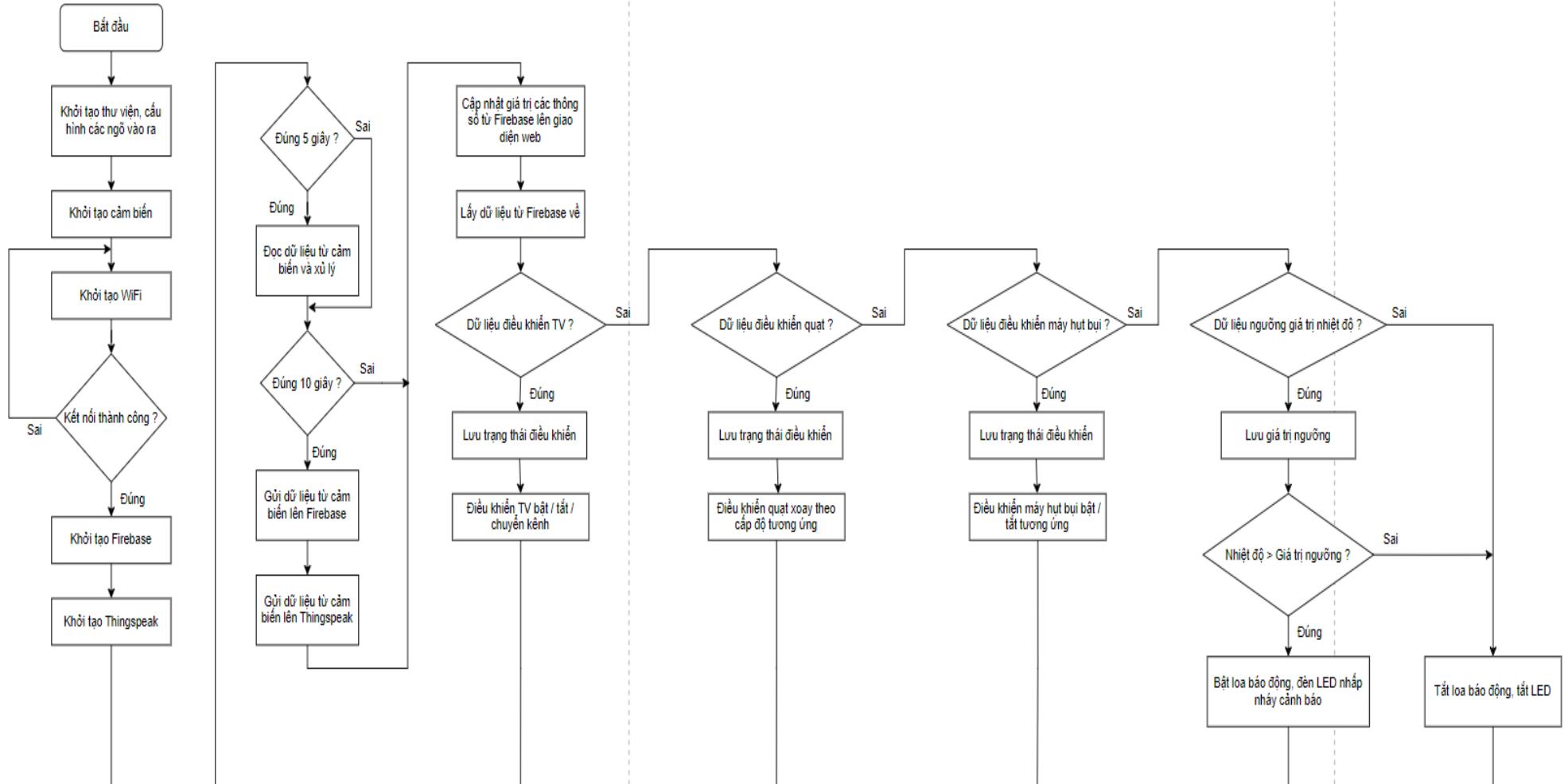
Do đó, nhóm lựa chọn điện trở 100Ω làm điện trở hạn dòng.

3.3.2 Thiết kế phần mềm

Đối với phần mềm, yêu cầu đặt ra là giao diện hiển thị một cách trực quan, thân thiện và dễ dàng tiếp cận với mọi người dùng. Các phím nhấn, biểu đồ được bố trí cân đối và hợp lý đảm bảo đầy đủ các tính năng mà không làm phức tạp hóa giao diện. Đồng thời, quá trình tương tác giữa phần cứng và giao diện phải được đáp ứng kịp thời, không quá chậm trễ cũng như là độ chính xác trong suốt quá trình điều khiển và hoạt động. Các quá trình lấy mẫu dữ liệu, chuyển tiếp dữ liệu phải được xử lý định thời, tránh tình trạng quá tải trong suốt quá trình xử lý.

3.3.2.1 Lưu đồ giải thuật hệ thống

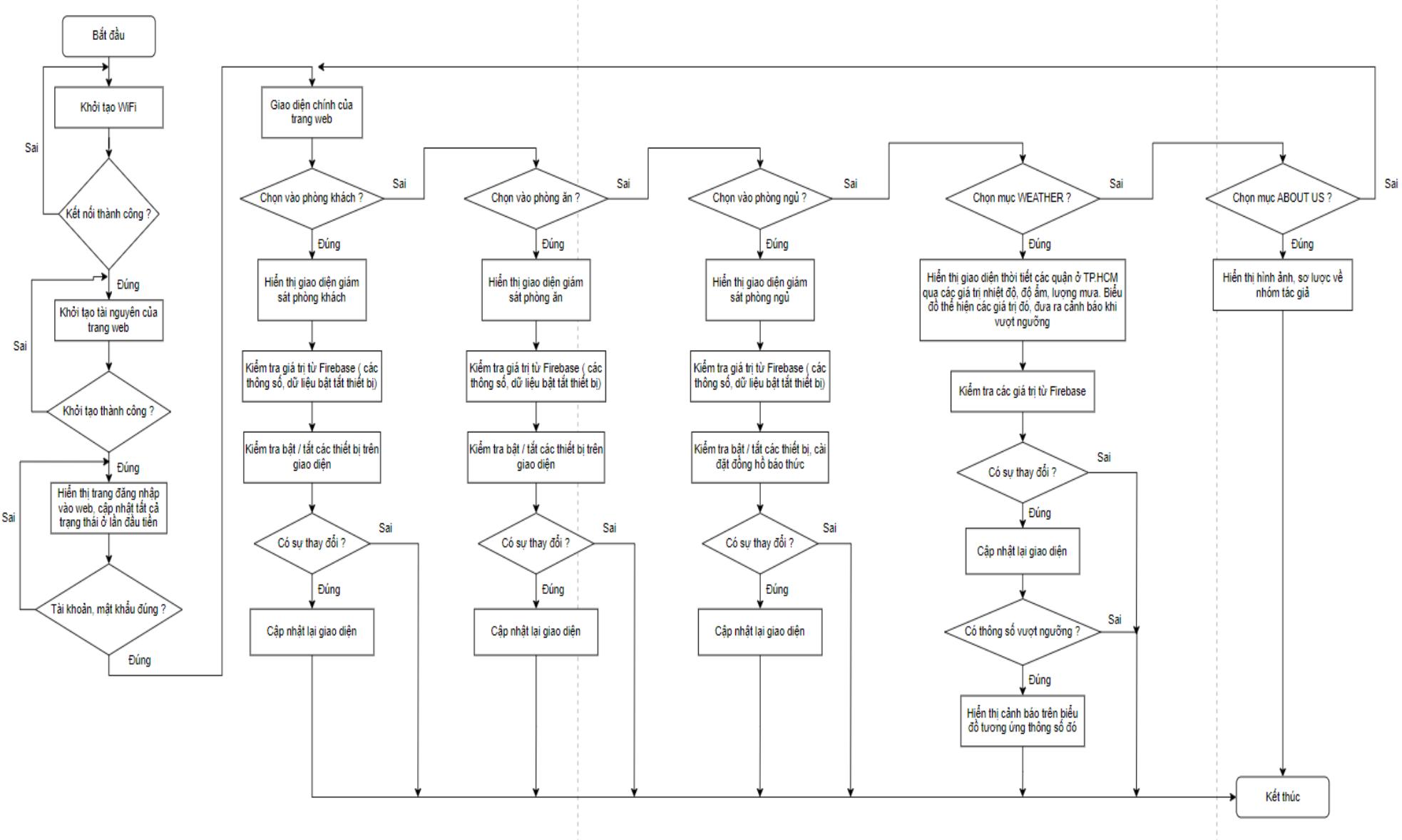
Để thực hiện được yêu cầu trên, nhóm đã thiết kế hệ thống với lưu đồ giải thuật. Hoạt động của hệ thống dựa trên lưu đồ giải thuật như sau. Đầu tiên, khi hệ thống vừa được cấp nguồn, hệ thống sẽ khởi tạo các thư viện cần thiết để kết nối với WiFi, khởi tạo các ngõ vào ra của hệ thống và các cảm biến, sau đó sẽ tiến hành khởi tạo Firebase và ThingSpeak. Sau khi hoàn tất việc khởi tạo các tài nguyên cần thiết, hệ thống sẽ bắt đầu hoạt động theo chu trình. Cứ mỗi 5 giây, hệ thống sẽ đọc dữ liệu từ các cảm biến một lần để đáp ứng được thời gian lấy mẫu của cảm biến cũng như tránh việc lấy mẫu quá nhanh dẫn đến dữ liệu không chính xác. Và cứ sau mỗi 10 giây, thì tất cả dữ liệu từ các cảm biến sẽ được gửi lên cơ sở dữ liệu Firebase để lưu trữ và đồng bộ với Web cũng như gửi lên ThingSpeak để trực quan hóa dữ liệu. Tiếp theo, hệ thống sẽ lấy dữ liệu điều khiển thiết bị ngoại vi từ Firebase về, tương ứng với từng dữ liệu hệ thống sẽ điều khiển thiết bị ngoại vi tương ứng. Cũng như đối với giá trị ngưỡng thì hệ thống cũng sẽ so sánh giá trị của cảm biến ở hiện tại, khi có sự quá nhiệt xảy ra thì hệ thống sẽ cảnh báo bằng còi và led.



Hình 3.3.4: Lưu đồ giải thuật toàn hệ thống

3.3.2.2 Lưu đồ giải thuật giao diện Web

Lưu đồ giải thuật cho giao diện Web được trình bày bên dưới. Ban đầu tiến hành khởi tạo WiFi để có kết nối với Internet để thực thi các tác vụ giao tiếp với Firebase. Các tài nguyên của trang Web cũng được khởi tạo trước khi khởi động. Sau khi khởi tạo xong các tài nguyên cần thiết, trang Web sẽ cập nhật các giá trị lần đầu tiên khi vừa tải trang từ trên Firebase về để đồng bộ trạng thái với Firebase và sau đó hiển thị các giao diện cho người dùng đăng nhập. Sau khi đăng nhập sẽ tiến vào giao diện chính, ở đây sẽ có 3 mục chính tương ứng với 3 phòng là phòng khách, phòng ăn, phòng ngủ để người dùng có thể truy cập, ngoài ra sẽ có 2 phần nữa là WEATHER và ABOUT US. Đối với phòng khách, khi ta truy cập vào sẽ hiển thị giao diện đo thông số của nhiệt độ, độ ẩm, khí gas và các thông số này sẽ liên tục được cập nhật khi có sự thay đổi, ngoài ra phòng khách còn điều khiển 3 thiết bị là TV, quạt và máy hút bụi với cái nút nhấn on/off và các mức quay của quạt. Phòng ăn cũng sẽ tương tự nhưng nó sẽ điều khiển 3 thiết bị là bếp, lò quay và tủ lạnh. Phòng ngủ cũng sẽ đo thông số tương tự nhưng sẽ điều khiển máy lạnh, đèn và một đồng hồ thời gian thực (ta có thể cài đặt báo thức cho đồng hồ này). Giao diện mục WEATHER sẽ hiển thị thời tiết ở quận 1, quận 7, Thủ Đức và Bình Thạnh ở TP.HCM bao gồm đo các thông số nhiệt độ, độ ẩm và lượng mưa. Đồng thời sẽ có 3 biểu đồ đại diện cho 3 thông số đó, mỗi khi có thông số vượt ngưỡng quy định thì sẽ cảnh báo trên biểu đồ. Mục cuối cùng là ABOUT US, đây là giao diện tổng quan nói về nhóm tác giả.

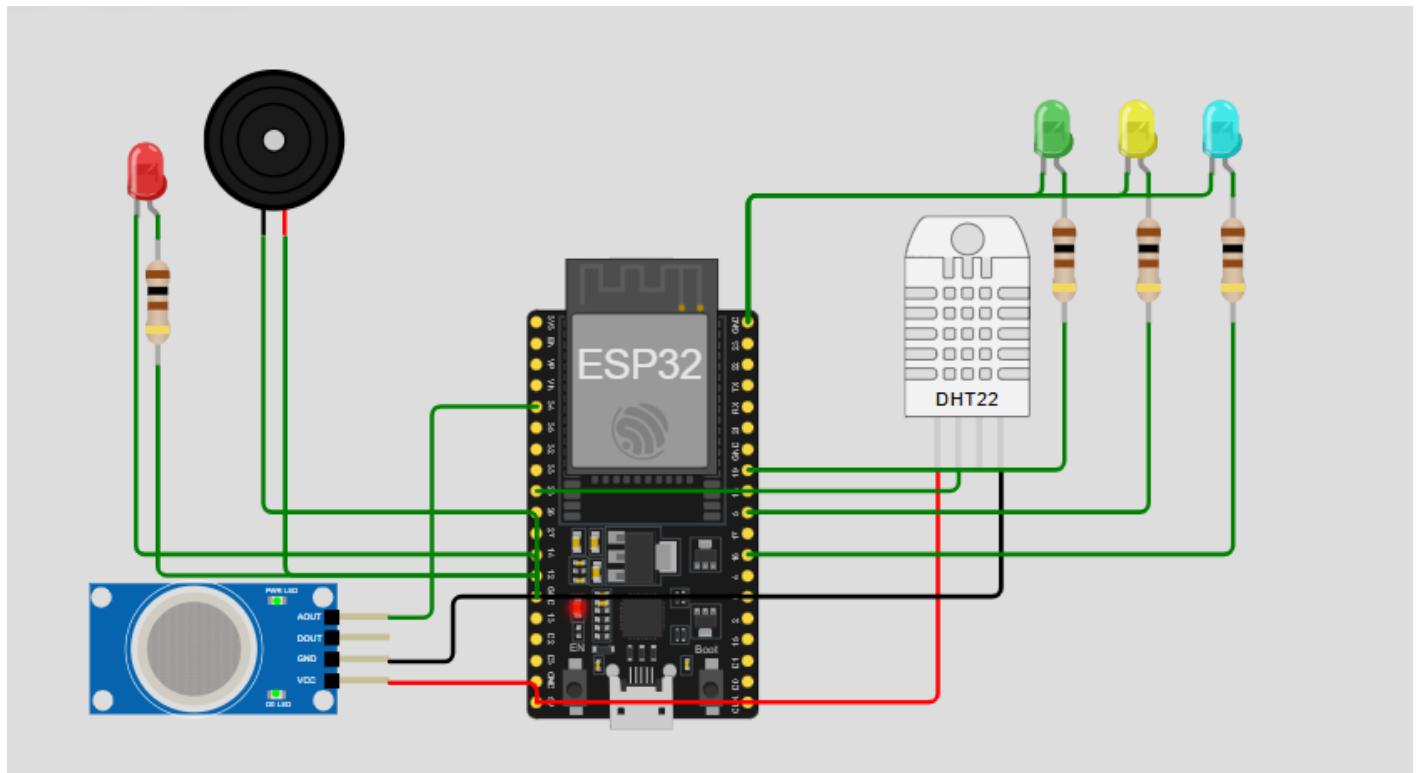


Hình 3.3.5: Lưu đồ giải thuật giao diện WEB

3.4 Kết quả

3.4.1 Kết quả mô phỏng phần cứng trên WOKWI

Sau khi đã kết nối các khối lại với nhau, nhóm thu được phần cứng hoàn chỉnh khi mô phỏng trên wokwi như hình bên dưới.

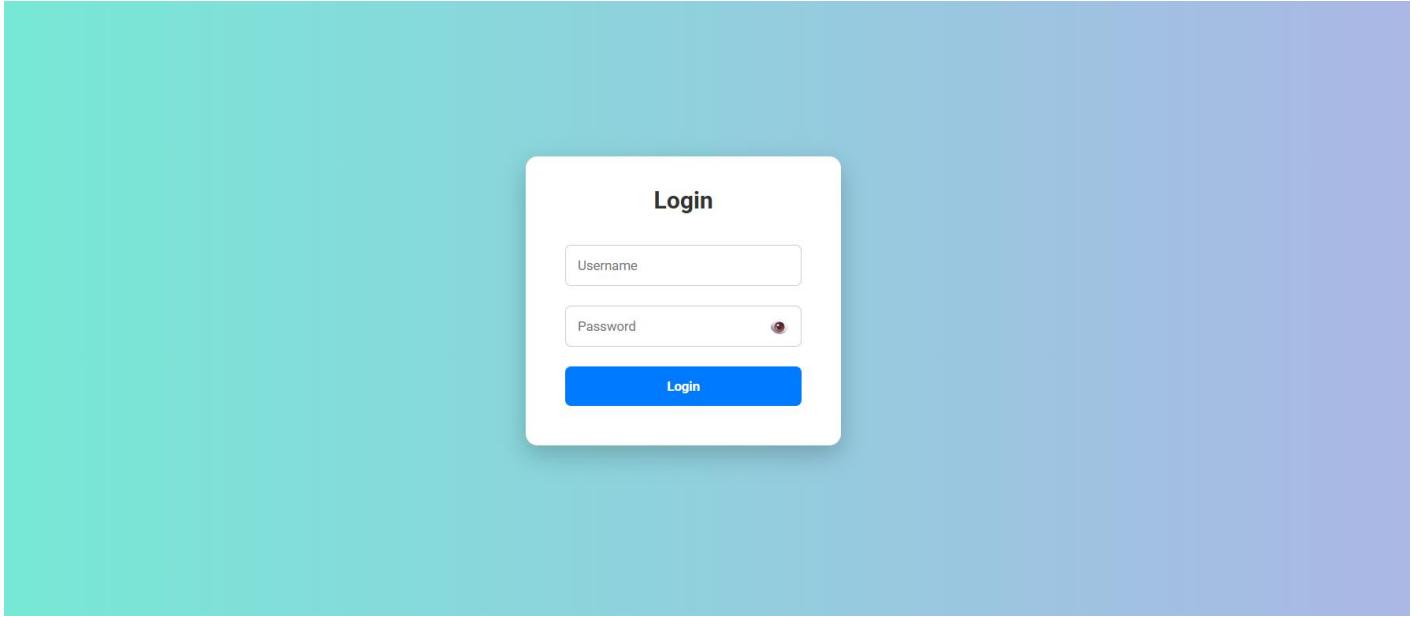


Hình 3.4.1: Phần cứng mô phỏng trên WOKWI

Phần cứng mô phỏng gồm ESP32 là vi điều khiển chính, cảm biến DHT22 (nhiệt độ, độ ẩm) và cảm biến MQ2 (nồng độ khí gas) được kết nối vi điều khiển, một led đơn đỏ và còi buzzer để cảnh báo, 3 led với 3 màu khác nhau đại diện cho 3 thiết bị trong phòng khách.

3.4.2 Kết quả phần mềm (giao diện web)

Giao diện đăng nhập đầu tiên được mô tả như hình bên dưới, ta sẽ tiến hành nhập tài khoản và mật khẩu vào sau đó nhấn Login để vào giao diện chính. Nếu nhập sai thì sẽ thông báo sai tài khoản mật khẩu và không thể đăng nhập vào giao diện chính cho đến khi nhập đúng. Ngoài ra nó còn có chế độ ẩn và hiện mật khẩu khi đăng nhập. Username sẽ là “admin”, còn Password là “1234”.



Hình 3.4.2: Giao diện đăng nhập

Giao diện sau khi đăng nhập sẽ là giao diện chính, ở đây sẽ bao gồm mục phòng ăn, phòng khách, phòng ngủ, mục Weather, About Us và nút đăng xuất tài khoản. Ta muốn truy cập vào phần nào thì ta chỉ cần chọn và phần đó

The main interface of the IoT Smart Monitor web application. At the top, there's a header bar with the HCMUTE logo on the left and a UTC logo on the right. The title "IoT Smart Monitor" is centered. Below the header, a navigation bar includes "Home" (highlighted in yellow), "Weather", "About Us", and a "Đăng xuất" (Logout) button. The main content area displays three cards representing different rooms: "Phòng khách" (Living Room) with an image of a room with white walls and two small potted plants; "Phòng ăn" (Kitchen) with an image of a modern kitchen; and "Phòng ngủ" (Bedroom) with an image of a bedroom. Each card has a "Read More" button at the bottom.

Hình 3.4.3: Giao diện chung của trang web

Ta sẽ tiến hành truy cập vào giao diện đầu tiên là phòng khách, ở đây sẽ hiển thị giá trị các thông số được cập nhật từ dữ liệu cảm biến mà ta đã mô phỏng trên Wokwi, ngoài ra ta cũng sẽ điều khiển 3 thiết bị là TV, máy hút bụi và quạt trong giao diện này. Ngoài ra khi ta bật/tắt các thiết bị tương ứng ở đây thì những con led đại diện cho các thiết bị này trong mô phỏng Wokwi cũng sẽ sáng/tắt tương ứng.



Hình 3.4.4: Giao diện phòng khách

Đây là giao diện khi ta chọn vào phòng ăn, ở đây cũng sẽ hiển thị giá trị các thông số nhưng các thông số này được nhập từ Firebase qua và ở đây ta sẽ điều khiển 3 thiết bị là bếp gas, lò vi sóng và tủ lạnh.



Hình 3.4.5: Giao diện phòng ăn

Tiếp đến là giao diện phòng ngủ, , ở đây cũng sẽ hiển thị giá trị các thông số nhưng các thông số này được nhập từ Firebase qua và ta sẽ điều khiển được máy lạnh, đèn, ngoài ra phòng ngủ sẽ có một đồng hồ thời gian thực cùng với chức năng đặt báo thức.



Hình 3.4.6: Giao diện phòng ngủ

Tiếp theo là giao diện khi ta chọn vào mục Weather, ở đây sẽ thể hiện thời tiết quận 1, quận 7, Bình Thạnh và Thủ Đức ở TP.HCM. Nó gồm các thông số nhiệt độ, độ ẩm và lượng mưa được nhập trực tiếp từ Firebase. Ngoài ra ta sẽ có 3 biểu đồ thể hiện giá trị của các thông số này, khi có giá trị vượt ngưỡng thì sẽ đưa ra cảnh báo bên dưới biểu đồ.



Hình 3.4.7: Giao diện mục Weather

Cuối cùng là giao diện About Us, đây là giao diện tổng quan về dự án này và hình ảnh của nhóm tác giả.

Về Nhóm & Đề Tài



IoT Smart Monitor là dự án được thực hiện bởi nhóm sinh viên đam mê công nghệ, với mục tiêu xây dựng hệ thống giám sát và điều khiển nhà thông minh dựa trên nền tảng IoT.

Đề tài tập trung vào việc thu thập, hiển thị và phân tích các thông số môi trường (nhiệt độ, độ ẩm, lượng mưa,...) tại các khu vực trong nhà và ngoài trời, giúp người dùng dễ dàng theo dõi và điều khiển thiết bị từ xa.

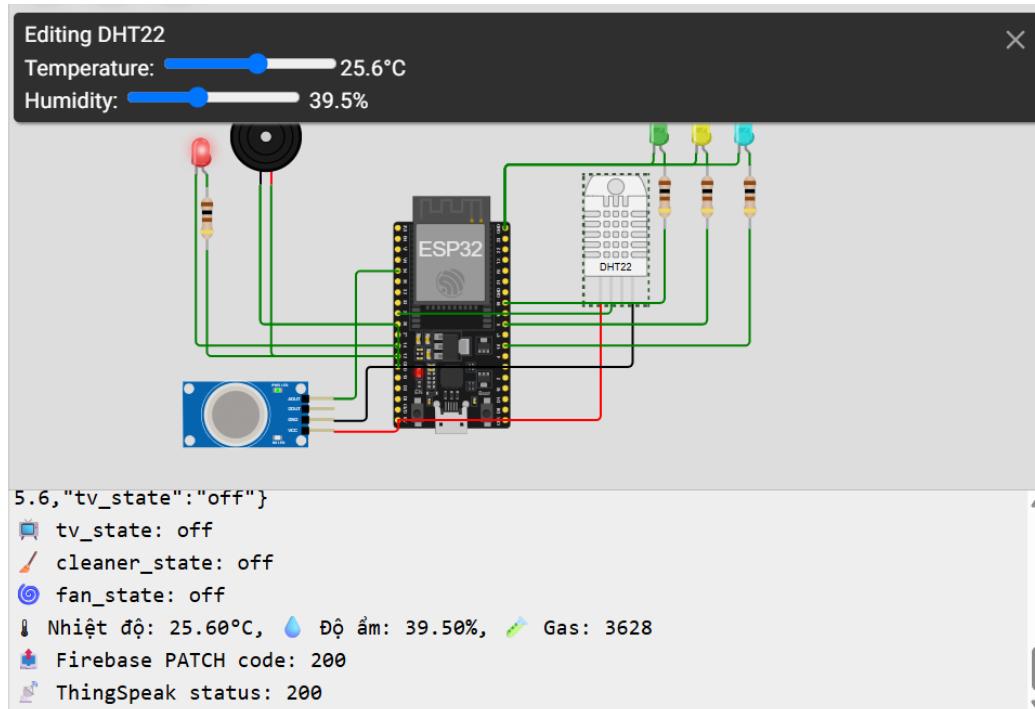
Sản phẩm hướng đến sự tiện nghi, an toàn và tiết kiệm năng lượng cho ngôi nhà hiện đại.

[Lê Anh Quất](#) [Khuê Chí Khang](#) [Lê Quốc Đạt](#)

Hình 3.4.8: Giao diện mục *About Us*

3.4.3 Kết quả hoạt động của toàn hệ thống

Ở đây nhóm đã thống nhất chọn ra phòng khách là nơi đặt phần cứng mô phỏng. Khi bắt đầu chạy mô phỏng trên Wokwi thì các thông số đo được từ cảm biến sẽ được gửi lên Firebase, Thingspeak từ đó cũng sẽ hiển thị qua giao diện web tương ứng, ta có thể quan sát ở hình bên dưới.



Hình 3.4.9: Cập nhật dữ liệu từ cảm biến và gửi lên firebase, thingspeak

```

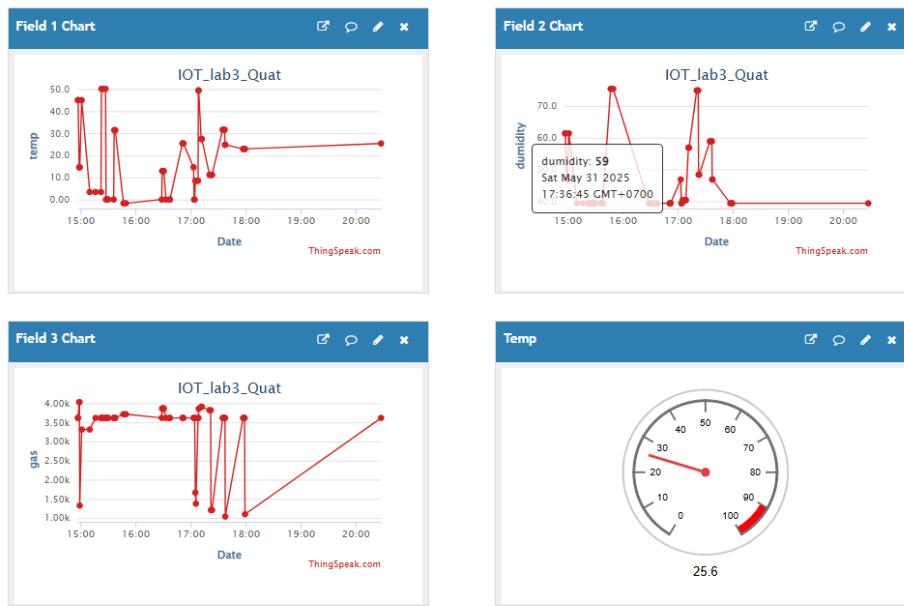
    ▼ -- livingroom
        cleaner_state: "off"
        fan_state: "off"  
        gas: 3628
        humidity: 39.5
        temperature: 25.6
        tv_state: "off"
    
```

Hình 3.4.9: Dữ liệu trên firebase



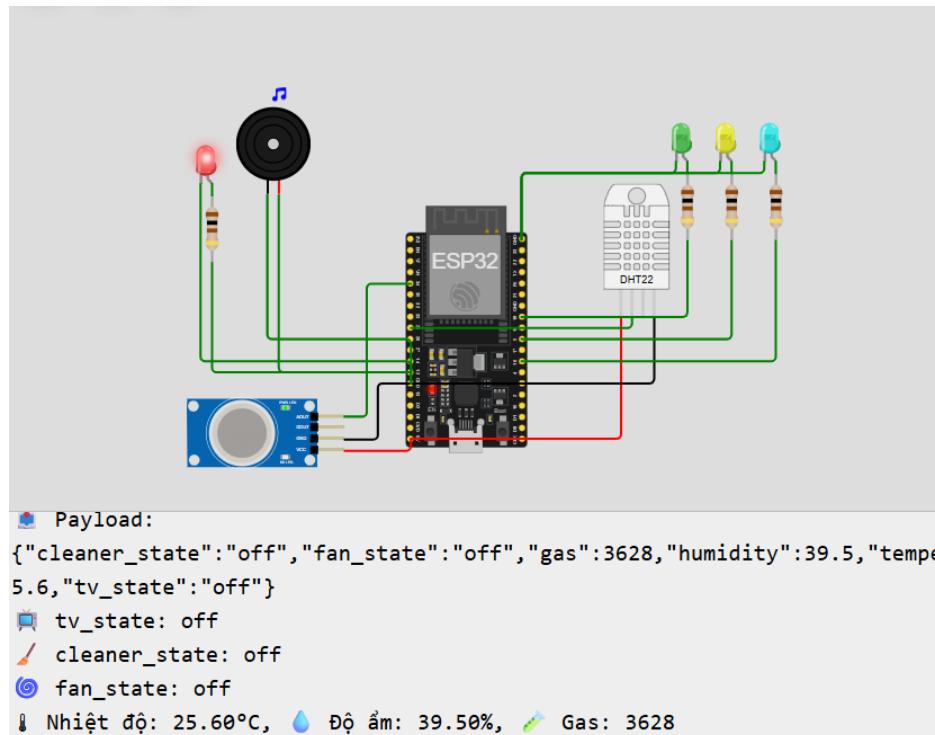
Hình 3.4.10: Các giá trị trên giao diện web

Ngoài ra các giá trị này cũng sẽ được gửi lên Thingspeak để ta có thể quan sát một cách trực quan hơn, nó thể hiện cho ta thấy các biểu đồ về sự thay đổi của các thông số theo thời gian thực. Chỉ khi phần cứng được chạy thì mới cập nhật, hình bên dưới gồm 3 biểu đồ cho 3 thông số, mỗi chấm đỏ là giá trị thông số được gửi đi, ở đây nhóm tác giả đã thay đổi nhiều giá trị khác nhau để có thể dễ dàng quan sát thấy sự thay đổi.



Hình 3.4.11: Dữ liệu được gửi lên Thingspeak

Mỗi khi đo được thông số quá ngưỡng thì đèn led sẽ nhấp nháy, còi báo động sẽ liên tục hú đèn khi không còn thông số nào vượt ngưỡng. Các giá trị ngưỡng đã được nhóm đặt ra gồm nhiệt độ $> 40^{\circ}\text{C}$ hoặc độ ẩm $> 60\%$ hoặc nồng độ khí gas $> 1500\text{ppm}$. Chỉ cần có 1 trong 3 thông số trên cao hơn giá trị ngưỡng thì sẽ phát cảnh báo. Hình bên dưới là led đơn đỏ và còi cảnh báo khi có thông số vượt ngưỡng (nồng độ khí gas 3628ppm $> 1500\text{ppm}$).

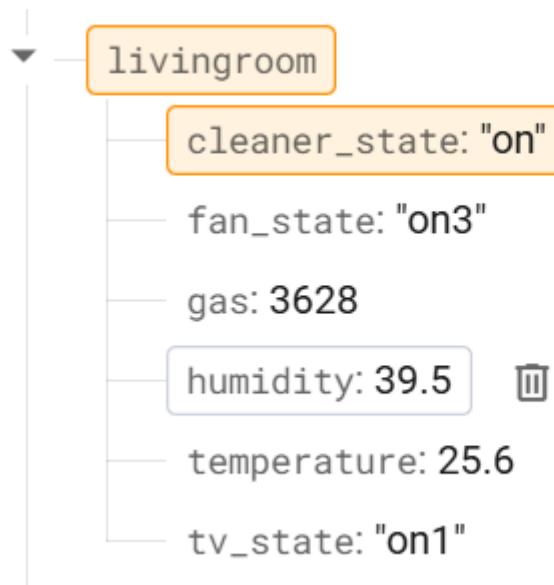


Hình 3.4.12: Cảnh báo khi có giá trị vượt ngưỡng

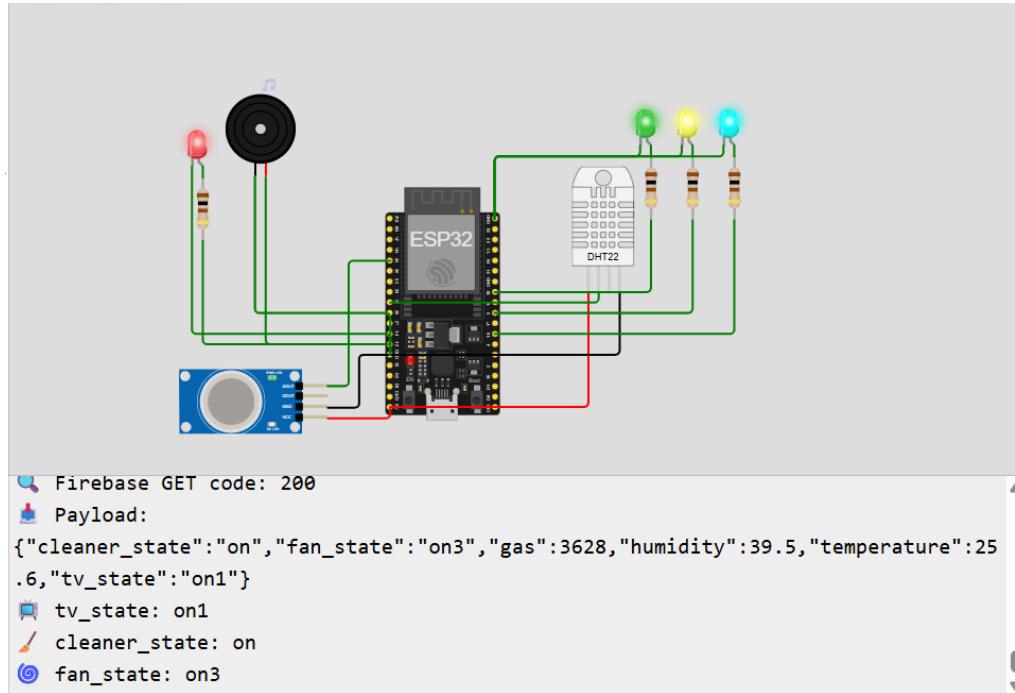
Tiếp đến nhóm dùng 3 con led để thay thế đại diện cho 3 thiết bị ở phòng khách, cụ thể led xanh lá sẽ đại diện cho TV, led vàng sẽ đại diện cho máy hút bụi, led xanh dương sẽ đại diện cho quạt. Chỉ cần trên giao diện ta bật / tắt thiết bị tương ứng thì trạng thái on / off sẽ được cập nhật vào các biến tương ứng trong Firebase, đối với TV sẽ là biến tv_state (gồm 3 trạng thái: off, on1 và on2), máy hút bụi sẽ là biến cleaner_state (gồm 2 trạng thái: on và off), cuối cùng là quạt với biến fan_state (gồm 4 trạng thái: off, on1, on2 và on3), từ đó ESP32 sẽ đọc dữ liệu đó và xử lý bật/ tắt led tương ứng. Ở đây nhóm sẽ bật cả 3 thiết bị, ta sẽ quan sát hình bên dưới để thấy kết quả,



Hình 3.4.13: Giao diện web khi bật 3 thiết bị



Hình 3.4.14: Giá trị trạng thái được cập nhật cho Firebase



Hình 3.4.15: Các led tương ứng sẽ sáng đèn, thông tin hiển thị trên terminal

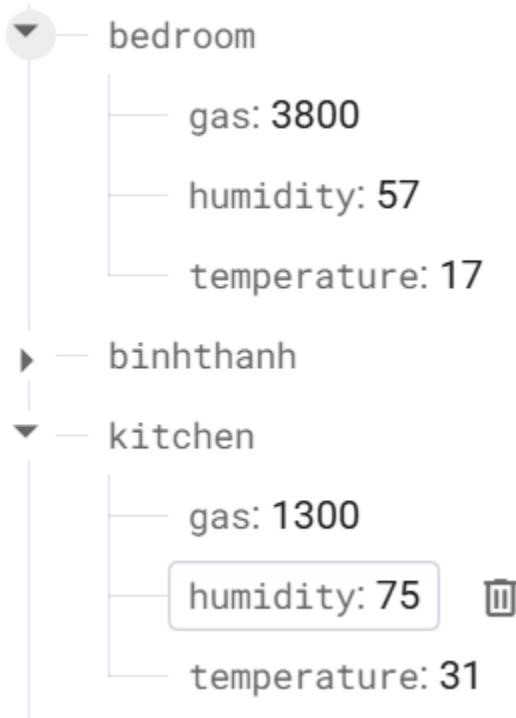
Tiếp đến ta có phòng ăn và phòng ngủ, các giá trị này chỉ được nhập trực tiếp từ Firebase, hình bên dưới sẽ là giao diện của 2 phòng khi bật cả 3 thiết bị (đối với phòng ngủ thì sẽ là trạng thái khi báo thức kêu), đồng hồ của phòng ngủ sẽ chạy theo thời gian thực.



Hình 3.4.16: Phòng ăn khi bật cả 3 thiết bị



Hình 3.4.17: Phòng ngủ khi bật cả 3 thiết bị



Hình 3.4.18: Giá trị các thông số tương ứng trên Firebase

Sau nhiều lần kiểm tra, nhóm rút ra một số thông số về thời gian đáp ứng của hệ thống qua các lần thử nghiệm như sau:

Tiêu chí	Thời gian
Cập nhật dữ liệu từ cảm biến lên Firebase	5 – 10 giây
Cập nhật dữ liệu từ Firebase về Web	Hầu như tức thời
Cập nhật dữ liệu điều khiển từ Web lên Firebase	Hầu như tức thời
Cập nhật dữ liệu điều khiển từ Web về phần cứng	5 – 15 giây
Thời gian tiên hành cảnh báo khi thông số cảm biến quá ngưỡng	1 – 2 giây

3.5 Kết luận

Hệ thống giám sát và điều khiển môi trường phòng đã hoàn thành đúng theo các yêu cầu đặt ra, bao gồm chức năng thu thập và gửi dữ liệu cảm biến (nhiệt độ, độ ẩm, khí gas) lên Firebase và ThingSpeak, cũng như điều khiển các thiết bị như quạt, TV, máy hút bụi từ xa thông qua giao diện Web. Giao diện người dùng được thiết kế trực quan, dễ sử dụng, hỗ trợ tốt cho việc theo dõi trạng thái hệ thống và nhận cảnh báo kịp thời khi các chỉ số vượt ngưỡng an toàn.

Tuy nhiên, hệ thống vẫn tồn tại một số hạn chế, đặc biệt là về thời gian đáp ứng điều khiển thiết bị. Độ trễ vẫn phụ thuộc đáng kể vào chất lượng kết nối WiFi và độ trễ từ các dịch vụ trung gian như Firebase. Điều này ảnh hưởng đến tính thời gian thực trong các tình huống yêu cầu phản hồi nhanh.

Để nâng cao hiệu quả và độ ổn định của hệ thống trong tương lai, có thể xem xét chuyển sang sử dụng các giao thức như MQTT hoặc tích hợp WebServer nội bộ, giúp giảm thiểu độ trễ, tối ưu luồng dữ liệu và cải thiện trải nghiệm người dùng.

Nhìn chung, hệ thống đã chứng minh được tính hiệu quả trong việc giám sát và điều khiển thiết bị từ xa, là nền tảng tốt cho việc phát triển các ứng dụng nhà thông minh trong thực tiễn.

LINK VIDEO DEMO: <https://youtu.be/wRjQwaAbJhs>

LINK MÔ PHỎNG WOKWI: <https://wokwi.com/projects/432459041202371585>

LINK GIAO DIỆN WEB: http://127.0.0.1:5500/dashboard_1/dashboard/landscape-responsive-card-main/index.html

TÀI LIỆU THAM KHẢO

- [1]. W3Schools, "JavaScript Tutorial", *W3Schools.com*, [Online].
- [2]. W3Schools, "ESP32 With Arduino", *W3Schools.com*, [Online].
- [3]. Firebase, "Firebase Realtime Database", *Firebase Documentation*, [Online].