😀

HTML CSS Reference

HTML Basics

```
<button>Hello</button>
<p>paragraph of text</p>
```

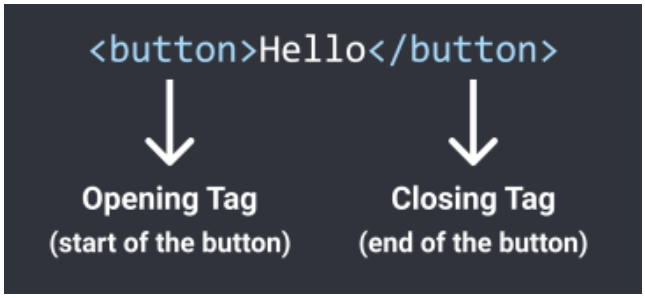
Tạo 1 nút với “Hello” bên trong

Tạo một đoạn văn bản

HTML Syntax

Cú pháp = quy tắc viết code HTML (như ngữ pháp trong tiếng Anh).

1. Các phần tử phải có thẻ mở và đóng tương ứng.



2. Trong HTML, các khoảng trắng và dòng trống được gộp thành 1 khoảng trắng

```
<p>paragraph of text</p>
<p>paragraph of text</p>
<p>
```

```
paragraph of text
</p>
```

Cả 3 ví dụ trên sẽ hiển thị kết quả giống nhau trên trang web

Thuộc tính thay đổi cách một phần tử HTML hoạt động.

```
<a href="https://youtube.com">
    Link to YouTubeLink to YouTube
</a>
```

<a> = Thẻ này tạo liên kết đến trang web khác.

href = thay đổi trang web sẽ được mở khi nhấp vào liên kết

```
<a href="https://youtube.com" target="_blank">
    Link to YouTube
</a>
```

target=”_blank” liên kết được mở trong tab mới.

CSS basics

Một cách viết code CSS là sử dụng thẻ <style> trong HTML.

```
<style>
    button {
      background-color: red;
      color: white;
    }
</style>
```

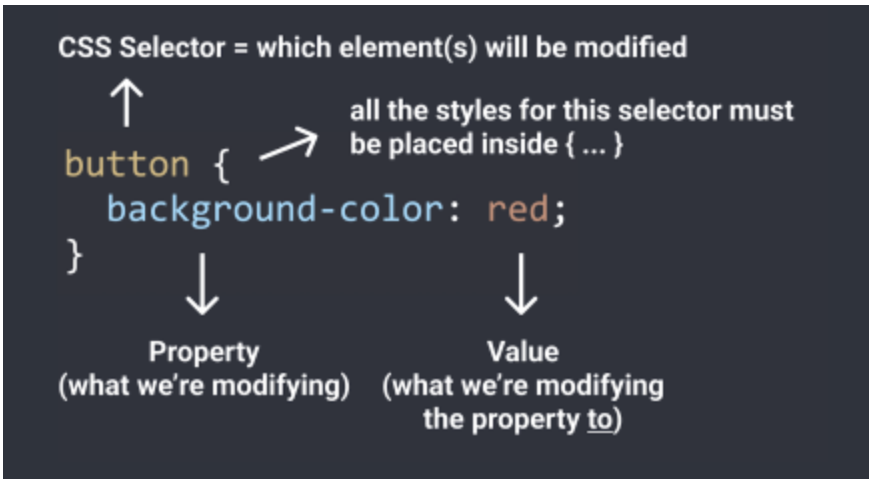
Thay đổi tất cả các nút trên trang web.

Thay đổi nền thành màu đỏ.

Thay đổi màu chữ thành trắng.

CSS Syntax

CSS selector = chọn phần tử được thay đổi



{ } tất cả styles cho selector phải đặt trong { ... }

background-color: Thuộc tính (thứ mà đang thay đổi)

red: Giá trị (giá trị mà thuộc tính sẽ được thay đổi thành)

CSS Properties

Đây là vài thuộc tính CSS phổ biến hay dùng:

```
button {
background-color: r
ed;



color: white;


height: 36px;
```

Thiết lập màu nền. Giá trị phổ biến:

Tên màu: red, white, black

Giá trị rgb: rgb(0, 150, 255) (màu xanh dương sáng)

Giá trị Hex: #0096FF (màu xanh dương)

Thiết lập màu văn bản. Chấp nhận các giá trị tương tự như màu nền (tên màu, rgb, hex).

Thiết lập chiều cao. Giá trị phổ biến:

Giá trị pixel: 36px

```
width: 105px;
border: none;

border-radius: 2px;
cursor: pointer;

border-color: red;

border-style: soli
d;



border-width: 1px;
}
```

Tỷ lệ phần trăm: 50%

Thiết lập chiều rộng. Chấp nhận các giá trị như height.

Loại bỏ viền

Tạo bo góc tròn

Thay đổi con trỏ khi di chuột qua phần tử.

Thiết lập màu của viền

Thiết lập kiểu viền. Giá trị phổ biến:

Solid: viền liền

Dotted: viền chấm chấm

Dashed: viền gạch ngang

Thiết lập độ dài của viền

How to Google CSS Properties

Đôi khi, bạn có thể sử dụng Google để tìm kiếm các thuộc tính CSS mà bạn không nhớ hoặc không biết. Khi dùng Google, tìm kiếm thứ đang cố hoàn thành. Ví dụ:

"css rounded corners" (góc bo tròn trong CSS)

"css text italic" (chữ nghiêng trong CSS)

"css adjust space between lines" (điều chỉnh khoảng cách giữa các dòng)

CSS Values

Mỗi thuộc tính CSS chấp nhận một tập hợp các giá trị cụ thể (background-color cho phép giá trị màu sắc color, cursor cho phép solid, dotted, dashed, v.v)

Dưới đây là một số giá trị hữu ích để biết:

Giá trị màu sắc:

1. Tên màu: red, white, black
2. RGB value: rgb(0, 150, 255);

RGB là cách đo lường màu sắc chính xác hơn. Mỗi màu sắc có thể được tạo ra bằng cách kết hợp các giá trị đỏ (red), xanh lá (green), và xanh dương (blue) (RGB). Trong CSS, điều này được biểu thị bằng rgb(...).

Mỗi màu có giá trị tối thiểu = 0 và giá trị tối đa = 255.

rgb(0, 0, 0); = black

rgb(255, 255, 255); = white

3. Giá trị Hex

Hex là một cách khác để viết RGB

Mỗi ký tự trong Hex có cơ số 16, có nghĩa là nó có thể có giá trị từ 0,1, 2, ... 8, 9, A, B, C, D, E, F(16 giá trị khả dụng).

Sử dụng 2 ký tự đầu tiên ta có 16 * 16 = 256 giá trị khả dụng từ 0 - 255:

00 = 0

01 = 1

...

0F = 15

10 = 16

11 = (1 * 16) + 1 = 17

...

FF = (15 * 16) + 15 = 255

Đây là cùng dải giá trị với RGB (0 đến 255), vì vậy 2 ký tự đầu tiên trong Hex dùng để biểu thị màu đỏ (red), 2 ký tự tiếp theo biểu thị màu xanh lá (green), và 2 ký tự cuối biểu thị màu xanh dương (blue). Thông thường, dễ dàng hơn khi sử dụng bộ chuyển đổi Hex sang RGB để chuyển đổi.

4. Giá trị RGBA: rgba(0, 150, 255, 0.5);

Tương tự như RGB, nhưng có thêm giá trị a(alpha). Giá trị alpha xác định độ trong suốt của màu sắc. 0 = hoàn toàn trong suốt, 1 = màu không trong suốt, 0.5 = trong suốt 50%.

Measurement Values

1. Pixels: 50px, 100px

Pixels (px) là đơn vị đo lường phổ biến trong thế giới số. Ví dụ: một màn hình 4K là 3840px x 2160px.

2. Percent: 50%, 100%

Một giá trị đo lường tương đối. Ví dụ, width: 50% có nghĩa là 50% chiều rộng của trang (hoặc nếu phần tử nằm bên trong một phần tử khác, 50% chiều rộng của phần tử chứa).

3. em / rem: 1em, 1rem

Các đơn vị đo lường tương đối hữu ích cho việc truy cập.

em = tương đối với kích thước font chữ của phần tử (2em có nghĩa là kích thước gấp đôi so với font).

rem = tương đối với kích thước font của trang, mặc định là 16px (2rem có nghĩa là kích thước font = 2 * 16px = 32px).

Class Attribute

Thuộc tính class = cho phép chúng ta chọn các phần tử cụ thể bằng CSS

```
<button class="subscribe-button">
    SUBSCRIBE
</button>
.subscribe-button {
    ...
}
```

Thêm một lớp vào phần tử. Tên lớp (phần văn bản trong dấu “. . .”) có thể là bất kỳ từ nào, nhưng không được chứa khoảng trắng.

Chọn tất cả các phần tử trên trang có lớp class=”subscribe-button”

```
<button class="youtube-button">
  SUBSCRIBE
</button>
<button class="youtube-button">
  JOIN
</button>
```

Nhiều phần tử có thể có cùng 1 lớp.

```
<button class="youtube-button subscribe-button">
  SUBSCRIBE
</button>
```

Một phần tử có thể có nhiều lớp, được phân tách bằng dấu cách.

```
button {
  ...
}

.youtube-button {
  ...
```

Phần tử có thể được chọn bởi nhiều bộ chọn CSS. Ở đây cả 3 bộ chọn CSS sẽ áp dụng cho nút trên.

```
}

.subscribe-button {
  ...
}
```

CSS Pseudo-Classes:

```
.subscribe-button:hover {
    ...
}
.subscribe-button:active {
    ...
}
```

Áp dụng khi di chuột qua phần tử có class là “subscribe-button”

Áp dụng khi nhấp chuột vào phần tử có class là “subscribe-button”

Intermediate CSS Properties

```
.subscribe-button {
    opacity: 0.5;
    opacity: 0;
    opacity: 1;

    transition: <property> <duration>;


    transition: background-color 1s;

    transition: color 0.15 s;
```

Thiết lập độ trong suốt của phần tử:
0.5 = 50% trong suốt.
0 = hoàn toàn trong suốt (vô hình).
1 = không trong suốt (đây là giá trị mặc định).

Chuyển đổi mượt mà khi thay đổi các kiểu (thường dùng khi di chuột).

Chuyển đổi màu nền trong 1 giây.

Chuyển đổi màu văn bản trong 0.15 giây.

Chuyển đổi nhiều thuộc tính bằng cách tách chúng bằng dấu phẩy.

```
    transition: <property1>
<duration1>, <property2> <d
uration2>, ...;

    transition: background-
color 0.15s, color 0.15s;

    box-shadow: <h-position
> <v-position> <blur> <colo
r>;
    box-shadow: 3px 4px 5px
black;

    box-shadow: 3px 4px 0 r
gba(0, 0, 0, 0.15);
    }
```

Chuyển đổi cả màu nền và màu văn bản trong vòng 0.15 giây.

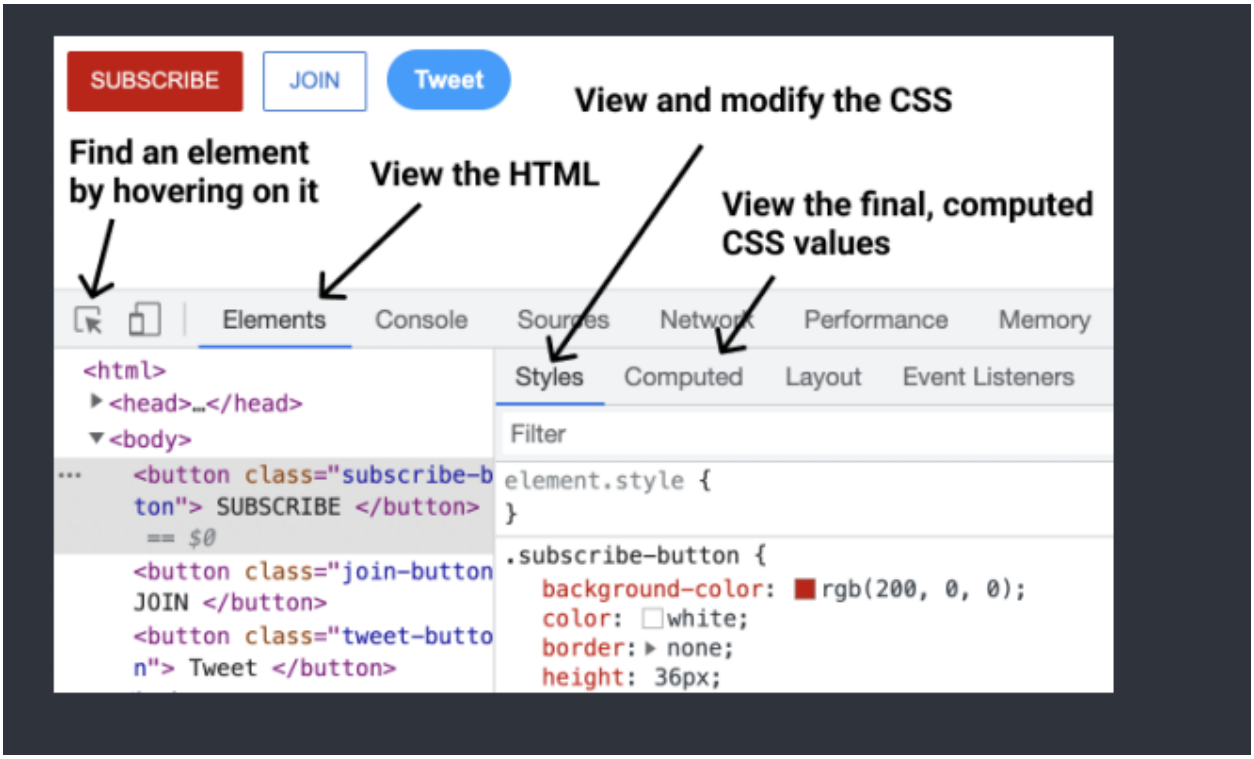
Tạo một bóng đổ cách 3px về bên phải, 4px bên dưới, với độ mờ 5px và màu đen.

Tạo một bóng đổ cách 3px về bên phải, 4px bên dưới, không có độ mờ và có một màu đen rất nhạt.

Chrome DevTools

Cho phép chúng ta xem (và sửa) HTML và CSS của một trang web trực tiếp trên trình duyệt.

Để mở DevTools: nhấp chuột phải > Inspect.



Find an element by hovering on it: Tìm một phần tử bằng cách di chuột lên nó.

View the HTML: Xem HTML.

View and modify the CSS: Xem và sửa CSS.

View the final, computed CSS values: Xem giá trị CSS cuối cùng đã được tính toán.

CSS Box Model

Xác định lượng không gian mà một phần tử chiếm.

Xác định khoảng cách giữa các phần tử với nhau.

1. Margin = không gian bên ngoài 2. Padding = không gian bên trong 3. Border = viền


```
.join-button {
    margin-right: 10px;
    margin-left: 10px;
    margin-top: 10px
    margin-bottom: 10px

    margin-right: -20px;









    margin: 10px;

    margin: 10px 20px;
    margin: <top> <left & r
ight> <bottom>;
    margin: <top> <right> <
bottom> <left>;

    padding-right: 10px;
    padding-left: 10px
    ;padding-top: 10px;
    padding-bottom: 10px;
```

Thêm 10px khoảng cách ở bên ngoài của phần tử.

Margin thông thường đẩy các thứa xa khỏi một phần tử.
Margin âm kéo các thứ lại gần một phần tử như thế này:

Viết tắt cho việc thêm 10px margin trên mọi phía
Thêm 10px margin ở trên và dưới, 20px margin ở trái và phải

Thêm 10px khoảng cách bên trong của phần tử.

Padding âm không có tác động.

```
    padding-right: -20px;

    padding: 10px;


    padding: 10px 20px;

    padding: <top> <left &
right> <bottom>;
    padding: <top> <right>
<bottom> <left>;

    border-width: 1px;
    border-style: solid;
    border-color: red;

    border: <width> <style>
<color>;
    border: 1px solid red;
```

Viết tắt cho việc thêm 10px padding trên mọi phía.

Thêm 10px padding trên và dưới, 20px padding trái và phải.

Thiết lập độ dày của viền.
Thiết lập kiểu viền (màu với kiểu solid).
Thiết lập màu viền.

Viết tắt cho 3 thuộc tính ở trên.

Text Styles

```
.title {
    font-family: Arial;
    font-family: Roboto, Ve
rdana, Arial;


    font-size: 30px;
    font-weight: bold;
    font-weight: 700;

    font-style: italic;
```

Đổi font chữ.

Sử dụng font-stack để nếu Roboto không có, sẽ chuyển sang Verdana, nếu Verdana cũng không có thì dùng Arial.

Đổi kích cỡ chữ.

Độ dày chữ (bold = 700, regular = 400, semibold = 500).

Chữ nghiêng

```
    text-align: center;

    line-height: 24px;

    text-decoration: underl
ine;
    text-decoration: none;
}
```

Căn chỉnh văn bản: center (giữa), left (trái), right (phải), justified (căn đều).

Điều chỉnh khoảng cách dòng.

Gạch chân.

Bỏ gạch chân.

Thẻ <p> mặc định có margin-top và margin-bottom. Một thực hành phổ biến là:

1. Đặt lại các margin mặc định.

```
p {
  margin-top: 0;
  margin-bottom: 0;
}
```

2. Sau đó áp dụng các margin chính xác hơn.

```
.title {
  margin-bottom: 16px;
}
```

Text Elements (cũng được gọi là Inline Elements)

- Các phần tử văn bản như , <u>, , <a> xuất hiện trong một dòng văn bản.

  Thẻ để làm nổi bật một phần của văn bản.

  Hữu ích nếu chúng ta chỉ muốn định dạng một phần của văn bản.
- Thẻ là phần tử văn bản tổng quát nhất (không có kiểu dáng mặc định).
- Ta có thể style các phần tử văn bản bằng cách sử dụng lớp:

```html
<p>
    This is a <span class="shop-link">text element</span>
</p>
```

```css
.shop-link {
    text-decoration : underline;
}
```

HTML Structure

```html
<!DOCTYPE html>
<html>
  <head>
    ...
  </head>
  <body>
    ...
  </body>
</html>
```

Khai báo với trình duyệt sử dụng phiên bản HTML hiện đại.

<head> chứa tất cả những thứ không hiển thị như tiêu đề và mô tả (còn gọi là siêu dữ liệu), cũng như các liên kết đến phông chữ và các tệp CSS stylesheets.

<body> chứa tất cả những thứ hiển thị như nút, văn bản, hình ảnh, v.v.

Những phần tử của phần Head

```html
<head>
  <title>Title in the tab</title>

  <link rel="preconnect" href="https://fonts.googleapis.com" />
  <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin />
```

Đặt tiêu đề trong các tab

Tải font từ Google vào trang.

1. Tìm kiếm "Google Fonts" trên Google.
2. Chọn font và kiểu mà bạn muốn.
3. Sao chép đoạn mã mà Google cung cấp và dán vào thẻ <head>.

```
  <link
    rel="stylesheet"
    href="https://fonts.goo
gleapis.com/css2?family=Rob
oto:wght@400;500&display=sw
ap">
    <link rel=”stylesheet”
href=”styles.css”>
</head>
```

Tải file CSS vào trang

Filepaths

```
href="styles.css"

href="fold1/styles.css"

href="fold1/fold2/style
s.css"
```

Tìm file có tên styles.css bên cạnh file HTML.

Tìm thư mục fold1 bên cạnh file HTML, sau đó vào thư mục này và tìm styles.css.

Vào thư mục fold1, vào tiếp fold2, tìm styles.css.

Images

```
<img src = "image.png">

<img src = "pics/image.
png">
<img class = "image" sr
c = "pics/image.png">
```

Tải hình ảnh image.png bên cạnh file HTML.

Tải image.png trong thư mục pics.

Thêm lớp CSS để điều chỉnh kích thước hình ảnh.

```
.image {
  width: 300px;
```

```
  height: 300px;


  object-fit: cove
r;


  object-fit: conta
in;

  object-position:
left;
  object-position:
right;
  object-position:
top;
  object-position:
bottom;
}
```

Thay đổi kích thước hình ảnh thành chiều rộng 300px. Chiều cao cũng sẽ được thay đổi để giữ tỉ lệ của hình ảnh.

Nếu cả chiều rộng và chiều cao đều được thiết lập, hình ảnh có thể bị kéo giãn.

Phóng to hình ảnh để che phủ toàn bộ khu vực width * height mà không bị kéo giãn hoặc biến dạng.

Thu nhỏ hình ảnh để nó được chứa trong khu vực width * height.

Xác định vị trí của hình ảnh trong khu vực width * height.

Inputs

```
<input type="text"

<input type="text" placeholder
="Search">

<input type="checkbox">
<input class="search-bar" type
="text" />
```

Tạo một ô nhập văn bản.

Thêm một placeholder (nhãn) cho ô nhập văn bản.

Tạo một checkbox.

```
.search-bar {
  font-size: 30px;
```

Thay đổi kích thước phông chữ khi gõ vào ô nhập văn bản.

```
}

.search-bar::placeholder {
  font-size: 30px;
}
```

Thay đổi kích thước phông chữ của placeholder.

CSS Display Property

```
.element {
  display: block;

  display: inline-b
lock;

  vertical-align: m
iddle;


  display: inline;
}
```

Phần tử sẽ chiếm toàn bộ dòng trong container của nó.

Phần tử sẽ chỉ chiếm không gian cần thiết. Xác định sự căn chỉnh theo chiều dọc của các phần tử inline-block.

Phần tử sẽ xuất hiện trong một dòng văn bản (một phần tử văn bản).

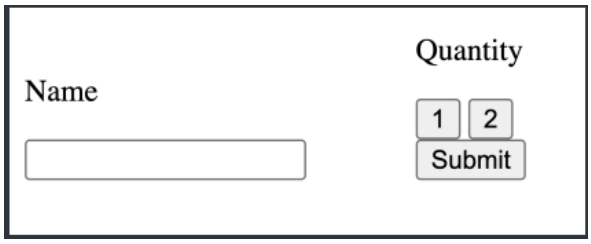
 Element

 là một thẻ chứa. Chúng ta thường đặt các phần tử khác (bao gồm cả các khác) bên trong (lồng ghép).

```
<div class="container">
  <p>Name</p>
  <input type="text">
</div>
<div class="container">
  <p>Quantity</p>
  <div>
    <button>1</button>
```

s cho phép chúng ta nhóm các phần tử lại với nhau và tạo ra các bố cục phức tạp hơn.

```
    <button>2</button>
  </div>
  <button>Submit</button>
</div>
```



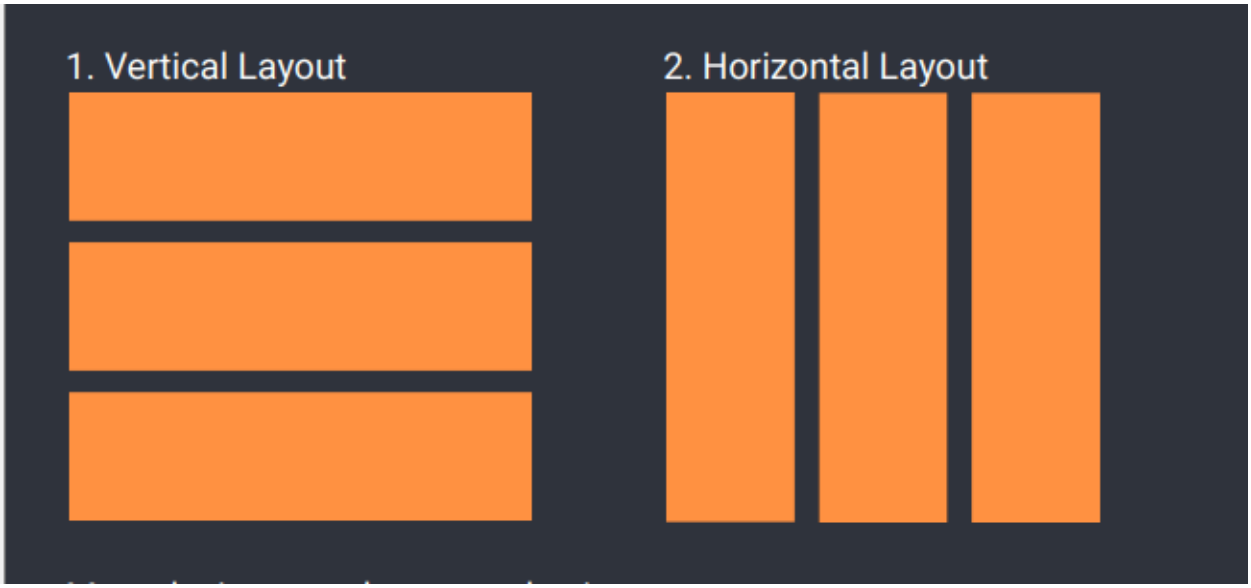

```
.container {
  display: inline-block;
  width: 200px;
}
```

Nested Layouts Technique

Có 2 loại bố cục:

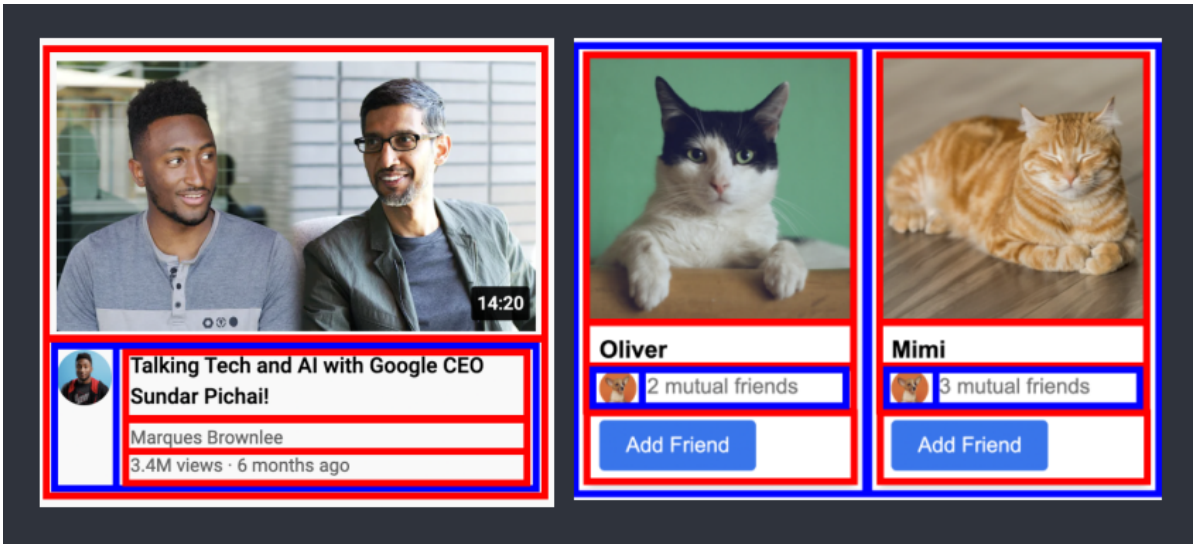
Bố cục dọc (Vertical Layout)

Bố cục ngang (Horizontal Layout)



Hầu hết các thiết kế có thể được tạo ra bằng cách sử dụng:

- **Bố cục dọc bên trong bố cục ngang bên trong bố cục dọc...**
- **Hoặc bố cục ngang bên trong bố cục dọc bên trong bố cục ngang...**



Để Tạo Bố Cục Dọc

- Sử dụng với display: block (phổ biến nhất)
- Sử dụng flexbox với flex-direction: column (được giải thích sau)
- Sử dụng CSS grid với 1 cột (được giải thích sau)

Để Tạo Bố Cục Ngang

- Sử dụng với display: inline-block (không được khuyến nghị)
- Sử dụng flexbox với flex-direction: row
- Sử dụng CSS grid với nhiều cột

Inline CSS Styles

Một cách khác để viết CSS là sử dụng thuộc tính style="...":

```
<div style="
    background-color: red;
```

- Inline style = CSS được viết bên trong một dòng HTML.

```
    color: white;
">
  ...
</div>
```

- Các kiểu inline chỉ ảnh hưởng đến phần tử có thuộc tính style="..." (không cần selectors).

CSS Grid

```
.grid {
  display: grid;
  grid-template-columns: 10
0px 100px;

  column-gap: 20px;

  row-gap: 40px;
}
```

Chuyển đổi phần tử thành grid container.

Xác định số lượng cột trong lưới và độ rộng của các cột.

Thiết lập khoảng cách giữa các cột.

Thiết lập khoảng cách giữa các hàng.

```
.grid {
  display: grid;
  grid-template-columns: 10
0px 1fr;


  grid-template-columns: 1f
r 1fr;


  grid-template-columns: 1f
r 2fr;


  justify-content: center;
```

1fr = cột sẽ chiếm số không gian còn lại trong container lưới.

Các cột sẽ chiếm một lượng không gian bằng nhau (vì chúng đều là 1fr).

Số ở trước fr = tỷ lệ không gian mà cột nhận được. Cột thứ 2 nhận gấp đôi không gian so với cột thứ nhất.

Căn chỉnh các cột theo chiều ngang ở giữa.

Phân tán các cột đều theo chiều ngang.

```
  justify-content: space-be
tween;
  align-items: center;
}
```

Căn chỉnh các cột theo chiều dọc ở giữa.

Để xem thêm các ví dụ, hãy tham khảo grid.html.

Flexbox

```
.flexbox {display: flex;

flex-direction: row;




justify-content: center;


justify-content: space-betw
een;

align-items: center;


align-items: space-between;
}
```

Chuyển đổi phần tử thành một container flexbox.

Sắp xếp các phần tử theo chiều ngang bên trong flexbox. Thông thường, chúng ta không cần chỉ định flex-direction: row; vì đây là giá trị mặc định.

Căn giữa các phần tử trong flexbox theo chiều ngang.

Trải đều các phần tử trong flexbox trên không gian ngang.

Căn giữa các phần tử trong flexbox theo chiều dọc.

Trải đều các phần tử trên không gian dọc.

```
.element-inside-fle
xbox {
  width: 100px;
  flex: 1;
```

Đặt chiều rộng của phần tử flexbox thành 100px.

Chiếm phần không gian còn lại. Giá trị 1 xác định tỷ lệ không gian mà phần tử sẽ nhận.

Không thu nhỏ phần tử khi thay đổi kích thước.

```
  flex-shrink: 0;

  width: 0;
}
```

Cho phép phần tử thu nhỏ khi thay đổi kích thước.

```
<div style="
    display: flex;
    flex-direction: row;
">

  <p style="width: 100px;">
      Flexbox element 1
  </p>
  <p style="flex: 1;">
      Flexbox element 2
  </p>

  <p style="flex: 2;">
      Flexbox element 3
  </p>
</div>
```

Tạo một flexbox nơi các phần tử được sắp xếp theo chiều ngang (đặt flex-direction: row; là mặc định nên không cần thiết phải có trong CSS).

Phần tử này có chiều rộng là 100px.

Phần tử này chiếm 1/3 không gian còn lại.

Phần tử này chiếm 2/3 không gian còn lại.

flex-direction: column;

```
.flexbox {
  display: flex;
  flex-direction: colum
n;

  justify-content: cent
er;
  justify-content: spac
e-between;
  align-items: center;
```

Sắp xếp các phần tử theo chiều dọc bên trong flexbox. Vì thế các thuộc tính justify-content và align-items sẽ được đảo ngược.

Căn giữa các phần tử theo chiều dọc trong flexbox.

Phân tán các phần tử đều trên không gian dọc.

Căn giữa các phần tử theo chiều ngang.

```
  align-items: space-be
tween;
}
```

Phân tán các phần tử đều trên không gian ngang.

Để xem thêm các ví dụ, hãy tham khảo flexbox.html.

CSS Position

●Tạo các phần tử đính kèm vào trang khi cuộn

● Tạo các phần tử xuất hiện ở phần trên của phần tử khác

```
.element {
    position: static;
}
```

Đây là giá trị mặc định mà mọi phần tử bắt đầu với. `position: static;` khiến phần tử hiển thị bình thường.

```
.fixed {
  position: fix
ed;

  top: 0;

  bottom: 10px;
  left: 50px;

  right: 100px;



  top: -5px;

  width: 100px;
  height: 100p
x;
}
```

Vị trí phần tử trong cửa sổ trình duyệt (bám chặt vào trang khi cuộn).

Đặt phần tử cách 0px từ mép trên của cửa sổ trình duyệt.

Cách 10px từ mép dưới của cửa sổ trình duyệt.

Cách 50px từ mép trái của cửa sổ trình duyệt.

Cách 100px từ mép phải của cửa sổ trình duyệt.

● Nếu bạn thiết lập các hướng đối diện (top/bottom hoặc left/right), phần tử sẽ kéo dài.

Sử dụng pixel âm sẽ đặt phần tử ra khỏi mép trên.

Thiết lập chiều rộng của phần tử là 100px.

Thiết lập chiều cao của phần tử là 100px.

● Khi sử dụng width/height, phần tử sẽ không thay đổi kích thước theo trang.

● Khi sử dụng top/bottom/left/right, phần tử sẽ thay đổi kích thước theo trang.

Position Absolute

```
.absolute {
  position: absolut
e;

  top: 0;
  bottom: 10px;

  left: 50px;

  right: 100px;

  width: 100px;
  height: 100px;
}
```

Đặt vị trí phần tử trên trang (nó sẽ cuộn cùng trang và không bám chặt khi cuộn).

Đặt phần tử cách 0px từ mép trên của trang.

Cách 10px từ mép dưới của trang.

Cách 50px từ mép trái của trang.

Cách 100px từ mép phải của trang.

Thiết lập chiều rộng của phần tử là 100px. Thiết lập chiều cao của phần tử là 100px.

Position Absolute Inside Position Fixed

Khi một phần tử với `position: absolute` nằm bên trong một phần tử với `position: fixed`, nó sẽ được định vị tương đối so với phần tử cố định đó.

● Quy tắc này cũng áp dụng cho bất kỳ giá trị `position` nào không phải là `position: static`.

● Điều này cho phép chúng ta đặt các phần tử vào các góc của các phần tử khác. Ví dụ: một nút "Close" ở góc trên bên phải.

```
<div style="
    position: fixed;
    width: 100px;
">
<button style="
    position: absolute;
```

Phần tử `position: absolute` sẽ được đặt ở góc trên bên phải của phần tử `position: fixed`.

```
    top: 0;
    right: 0;
    ">
        X
    </button>
</div>
```

Position Relative

```
.relative {
  position: relative;
  top: 10px;


  bottom: 10px;
  left: 50px;

  right: 100px;

  width: 100px;
  height: 100px;
}
```

Phần tử sẽ xuất hiện bình thường (như thể nó đang có `position: static`). Chúng ta có thể di chuyển nó bằng cách sử dụng `top/bottom/left/right`.
Đặt phần tử cách 10px từ vị trí gốc của nó (đẩy xuống 10px). Không giống như margin, nó sẽ không đẩy các phần tử khác trên trang xuống.
Đặt phần tử cách 10px từ vị trí gốc của nó (đẩy lên 10px).

Đặt phần tử cách 50px từ vị trí gốc bên trái.

Đặt phần tử cách 100px từ vị trí gốc bên phải.

Thiết lập chiều rộng của phần tử là 100px. Thiết lập chiều cao của phần tử là 100px.

Position Absolute Inside Position Relative

Khi một phần tử với `position: absolute` nằm bên trong một phần tử với `position: relative`, nó sẽ được định vị tương đối so với phần tử tương đối đó.

● Hữu ích khi chúng ta muốn hiển thị một phần tử bình thường (sử dụng `position: relative`), nhưng vẫn có thể đặt các phần tử khác vào góc (sử dụng `position: absolute`).

```
<div style="
  position: relativ
e;
  width: 100px;
">
  <button style="
    position: absol
ute;
    top: 0;
    right: 0;
  ">
    3
  </button>
</div>
```

Phần tử `position: absolute` sẽ được đặt ở góc trên bên phải của phần tử `position: relative`.

z-index

Xác định phần tử nào sẽ xuất hiện phía trước hoặc phía sau:

● Các phần tử có z-index cao hơn sẽ xuất hiện phía trước các phần tử có z-index thấp hơn. Giá trị z-index mặc định là 0.

● Các phần tử với `position: static;` luôn xuất hiện ở phía sau. z-index không ảnh hưởng đến chúng.

● Nếu z-index bằng nhau hoặc cả hai phần tử có `position: static`, phần tử được viết sau trong mã sẽ xuất hiện phía trước.

```
.fixed {
  position: fixed;
  z-index: 2;
}
.absolute {
  position: absolute;
  z-index: 1;
}
.static {
```

Phần tử này sẽ xuất hiện phía trước phần tử `position: absolute;` vì nó có z-index cao hơn.

Phần tử này sẽ xuất hiện ở phía sau vì nó có `position: static`.

```
  position: static;
}
```

Xem thêm ví dụ ở position.html.

Responsive Design

Responsive design = làm cho trang web hiển thị tốt trên mọi kích thước màn hình.

```
@media (max-width: 750px) {
.element {
        width: 350px;
    }
}
@media (min-width: 750.02px) an
d (max-width: 1000px) {
    .element {
        width: 450px;
    }
}
@media (min-width: 1000.02px) {
    .element {
        width: 600px;
    }
}
```

Chỉ áp dụng mã CSS bên dưới khi chiều rộng màn hình nằm trong khoảng 0px - 750px .

Chỉ áp dụng mã CSS này khi chiều rộng màn hình nằm trong khoảng 750px - 1000px.

Chỉ áp dụng mã CSS này khi chiều rộng màn hình trên 1000px.

Chúng ta thường sử dụng một khoảng cách .02px giữa các khoảng (như trên) vì trình duyệt có thể hỗ trợ kích thước màn hình phân số như 750.50px.

Advanced CSS Selectors

Với Dấu Phẩy

```
.class1, .class2 { ...
}
```

Nhắm mục tiêu đến nhiều class cùng lúc.
Nhắm mục tiêu đến một class và tất cả các

```
.class1, p { ... }
```

thẻ `<p>` cùng lúc.

```
.class1 img { ... }

.class1 img,
.class2 .tooltip { ...
}


.class2:hover .tooltip
{ ... }
```

Nhắm mục tiêu đến các thẻ `<img>` nằm trong các phần tử có `class="class1"`.
Nhắm mục tiêu đến các thẻ `<img>` nằm trong phần tử có `class="class1"` VÀ `.tooltip` nằm trong các phần tử có `class="class2"`.
Nhắm mục tiêu `.tooltip` chỉ khi di chuột qua các phần tử có `class="class2"`.

Inheritance

Thuộc tính văn bản được đặt trên phần tử bên ngoài sẽ được áp dụng cho các phần tử bên trong:

```
<div style="color: red;">
  <p>Paragraph</p>
</div>
```

Đoạn văn này sẽ có văn bản màu đỏ.

Đối với các kiểu văn bản toàn trang (kiểu mà chúng ta muốn áp dụng cho toàn bộ trang), chúng ta có thể thiết lập chúng trên `<body>`:

```
body {
font-family: Roboto, Arial;
color: rgb(20, 20, 20);
}
```

Tất cả các phần tử trên trang mặc định sẽ sử dụng `font-family: Roboto, Arial` và `color: rgb(20, 20, 20)`. Điều này có thể được ghi đè.

CSS Specificity

Nếu nhiều bộ chọn CSS thay đổi cùng một thuộc tính trên cùng một phần tử (xem ví dụ dưới), CSS Specificity sẽ quyết định bộ chọn nào "wins" (style nào được áp dụng).

```
body { color: black; }
p { color: red; }
.title { color: green; }
```

```
<body>
    <p class="t
itle">
        Paragra
ph
    </p>
</body>
```

Trong ví dụ này, bộ chọn `.title` có mức độ ưu tiên cao nhất (theo CSS Specificity) nên văn bản sẽ có màu xanh lá cây.

CSS Specificity Rules

Đây là toàn bộ quy tắc của CSS Specificity Rules (không cần nhớ hết).

Thông thường chỉ cần biết vài cái quy tắc hay dùng và tìm kiếm Google thêm nếu cần thiết:

1. CSS nội tuyến (Inline CSS) có mức độ ưu tiên cao hơn bộ chọn `.class`.
2. Bộ chọn `.class` có mức độ ưu tiên cao hơn bộ chọn tên phần tử (p).
3. Bộ chọn tên phần tử (p) có mức độ ưu tiên cao hơn sự thừa kế (từ body).
4. Nếu 2 bộ chọn có cùng mức độ ưu tiên, cái nào được viết sau sẽ được chọn.

General Rule of Thumb

Bộ chọn CSS càng cụ thể (nhắm mục tiêu đến tập hợp phần tử cụ thể hơn) thì có mức độ ưu tiên cao hơn.

Semantic Elements

Các phần tử hoạt động giống như `<div>`. Tuy nhiên, chúng cũng mang lại ý nghĩa cho HTML khi trình đọc màn hình, công cụ tìm kiếm hoặc thiết bị khác đọc trang web.

Các phần tử ngữ nghĩa thông dụng bao gồm:

`<header>`, `<nav>`, `<main>`, `<section>`, v.v.

Đây là danh sách Semantic Elements (các phần tử ngữ nghĩa). Chúng sẽ được tìm hiểu chi tiết hơn trong khóa học về khả năng truy cập (accessibility):

Comments

Cho phép chúng ta viết code mà trình duyệt bỏ qua. Hữu ích để ghi chú cách mã hoạt động.

```
<!-- This is a comment -->
<p class="title">
 Paragraph of text.
</p>
```

Cú pháp cho bình luận trong HTML: <!- - ... →

```
/* This is a comment */
.title {
    color: green;
}
```

Cú pháp cho bình luận trong CSS: /* ... */

Other CSS Properties

Một số thuộc tính CSS khác đã được đề cập.

```
.tooltip {
  pointer-events: none;

  white-space: nowrap;
}
```

Vô hiệu hóa tất cả tương tác với chuột (nhấp, di chuột, v.v.)

Ngăn văn bản bên trong phần tử xuống nhiều dòng.