CSS (Cascading Style Sheets)

1. CSS Units

Absolute units?

Thường được sử dụng để đề cập đến các đơn vị đo lường có giá trị tuyệt đối, không phụ thuộc vào các điều kiện khác nhau như kích thước màn hình hoặc thiết bị.

1. **px** (Pixels): Là đơn vị đo lường phổ biến nhất, đại diện cho điểm ảnh trên màn hình. Một pixel tương đương với một điểm ảnh trên màn hình.
2. **in** (Inches): Đo lường bằng inch (1 inch = 2.54 cm).
3. **cm** (Centimeters): Đo lường bằng centimeter.
4. **mm** (Millimeters): Đo lường bằng millimeter.
5. **pt** (Points): Phổ biến trong in ấn, tương đương với 1/72 inch.
6. **pc** (Picas): Cũng phổ biến trong in ấn, tương đương với 12 points.

Relative units?

Relative units trong CSS, hoặc các đơn vị tương đối, được sử dụng để xác định các kích thước như chiều ngang, chiều dọc, hình padding, margin, và kích thước font, mà kích thước này phụ thuộc vào một điểm gốc nào đó.

1. **%**: dựa trên kích thước của phần tử cha, như width của phần tử cha.
2. **rem**: dựa trên kích thức font-size của phần tử root (thường là `<html>`).
3. **em**: dựa trên kích thước font-size của phần tử cha.
4. **vw** (viewport width): 1vw bằng 1% của chiều ngang viewport.
5. **vh** (viewport height): 1vh bằng 1% của chiều dọc viewport.
6. **vmin** và **vmax**: phụ thuộc vào kích thước nhỏ nhất và lớn nhất giữa vw và vh.
7. **ex**: được đo lường dựa trên chiều cao của ký tự "x" trong phông chữ hiện tại.
8. **ch**: được đo lường dựa trên chiều rộng của ký tự "0" (chữ số 0) trong phông chữ hiện tại.

2. Padding:

- **padding** là một thuộc tính được sử dụng để thêm khoảng trắng xung quanh nội dung của một phần tử.
- lưu ý: làm tăng kích thước tổng của phần tử, không sử dụng được số âm, .
- ví dụ:

```
padding-top: 10px; /* trên 10px */
padding-right: 10px; /* phải 10px */
padding-bottom: 10px; /* dưới 10px */
padding-left: 10px; /* trái 10px */

padding: 10px; /* trên dưới trái phải 10px */

padding: 10px 20px; /* trên dưới 10px, trái phải 10px */
```

```
padding: 10px 20px 30px; /* trên 10px trái, phải 20px dưới 30px */

padding: 10px 20px 30px 40px; /* trên 10px trái 20px phải 30px dưới 40px */
```

3. Border:

- **border** là một thuộc tính được sử dụng để định rõ đường biên (border) của một phần tử HTML. border có thể được sử dụng để đặt độ dày, kiểu, và màu sắc của đường biên
- **border-width**: độ dày đường biên.
- **border-style**: kiểu đường biên.
- **border-color**: màu sắc đường biên.
- **border**: **border-width border-style border-color**.
- lưu ý: làm tăng kích thước tổng của phần tử giống padding.
- ví dụ:

```
.selector{
  border: 1px solid red;
}
```

4. Margin:

- **margin** là một thuộc tính được sử dụng để xác định khoảng trắng xung quanh một phần tử, tạo ra khoảng cách giữa phần tử và các phần tử khác trong trang web.
- khi margin quá khung chứa thì nó sẽ rớt xuống dòng;
- ví dụ:

```
margin-top: 10px; /* trên 10px */
margin-right: 10px; /* phải 10px */
margin-bottom: 10px; /* dưới 10px */
margin-left: 10px; /* trái 10px */

margin: 0 auto; /* căn giữa, hoạt động ở thẻ block */

margin: left auto; /* dồn về phía bên phải */

margin: right auto; /* dồn về phía bên trái */

margin: 10px; /* trên dưới trái phải 10px */

margin: 10px 20px; /* trên dưới 10px, trái phải 10px */

margin: 10px 20px 30px; /* trên 10px trái, phải 20px dưới 30px */

margin: 10px 20px 30px 40px; /* trên 10px trái 20px phải 30px dưới 40px */

margin-inline: 20px; /* trái 20px phải 20px */
```

```
 margin-block: 20px; /* trên 20px dưới 20px */
```

- lưu ý: khác với **padding** và **border** thì **margin** không làm tăng kích thước của phần tử.
- khi 2 thẻ dùng margin cùng kích thước thì sẽ bị margin collab.

5. Box-sizing:

- Là một thuộc tính trong CSS được sử dụng để xác định cách tính toán kích thước của một phần tử.
- Mặc định: **box-sizing: content-box;**
- Nếu sử dụng **box-sizing: border-box** thì k nên đặt **padding** hay **border** hoặc cả 2 cộng lại không được quá **width height** của phần tử.
- 1 element sẽ bao gồm padding border và nội dung content nếu padding hay border quá to so với content thì sẽ bị lỗi.

6. CurrentColor:

- **currentColor** là một giá trị chuyển đổi màu sắc hiện tại của phần tử đó. Nó giúp bạn áp dụng màu sắc hiện tại của văn bản vào các thuộc tính khác như border-color, background-color, hoặc box-shadow, mà không cần phải chỉ định màu sắc cụ thể.
- Ví dụ:

```
div {
  color: blue;
  border: 2px solid currentColor;
  padding: 10px;
  background-color: currentColor;
}
```

Điều này giúp giữ cho màu sắc đồng bộ và dễ bảo trì trong trường hợp bạn thay đổi màu sắc chủ đạo của văn bản.

7. Opacity:

- **opacity** trong CSS được sử dụng để xác định độ mờ của một phần tử và nó có giá trị từ 0 đến 1. Giá trị 0 đại diện cho phần tử hoàn toàn trong suốt, trong khi giá trị 1 là không có độ mờ.
- Ví dụ:

```
.transparent-box {
  background-color: blue;
  opacity: 0.5; /* nền màu xanh và độ mờ là 0.5 (50% độ mờ) */
}
.transparent-box {
  background-color: rgba(0, 0, 255, 0.5); /* Màu xanh với 50% độ mờ */
}
```

8. Visibility:

- `visibility` trong CSS được sử dụng để kiểm soát sự hiển thị của một phần tử trên trang web. Nó có thể có giá trị là visible, hidden, hoặc collapse (đối với các bảng).

1. **Visible**: Mặc định, phần tử sẽ hiển thị.

```
.visible-element {
  visibility: visible;
}
```

2. **Hidden**: Phần tử không được hiển thị, nhưng vẫn chiếm không gian trên trang.

```
.hidden-element {
  visibility: hidden;
}
```

3. **Collapse** (cho bảng): Chỉ áp dụng cho các phần tử bảng. Nếu sử dụng collapse, phần tử sẽ được ẩn và không chiếm không gian, tương tự như hidden, nhưng có hiệu ứng đặc biệt cho bảng.

```
table {
  visibility: collapse;
}
```

9. Display:

- `display` trong CSS được sử dụng để xác định cách một phần tử được hiển thị trong trang web.

1. **block**: Phần tử sẽ chiếm toàn bộ chiều rộng của phần cha và xuống dòng mới, không chia sẻ hàng ngang với các phần tử khác.

```
.block-element {
  display: block;
}
```

2. **inline**: Phần tử sẽ chỉ chiếm chiều rộng và chiều cao tối đa cần thiết để hiển thị nội dung, không tạo ra xuống dòng mới.

```
.inline-element {
  display: inline;
}
```

3. **inline-block**: Phần tử sẽ chiếm chiều rộng và chiều cao tối đa cần thiết để hiển thị nội dung, nhưng vẫn tạo ra xuống dòng mới giống như block.

```
.inline-block-element {
  display: inline-block;
}
```

4. **none**: Phần tử sẽ bị ẩn và không chiếm không gian trên trang.

```
.hidden-element {
  display: none;
}
```

5. **flex**: Phần tử sẽ sử dụng mô hình linh hoạt (flexbox) để tự động điều chỉnh và sắp xếp các phần tử con.

```
.flex-container {
  display: flex;
}
```

6. **grid**: Phần tử sẽ sử dụng mô hình lưới (grid) để tự động sắp xếp và kiểm soát vị trí các phần tử con.

```
.grid-container {
  display: grid;
}
```

10. Overflow:

- `overflow` trong CSS được sử dụng để xác định cách xử lý nội dung khi nó vượt ra khỏi kích thước của phần tử chứa.

1. **visible**: Mặc định. Nội dung sẽ vượt ra khỏi ranh giới của phần tử và hiển thị bên ngoài.

```
.visible-overflow {
  overflow: visible;
}
```

2. **hidden**: Bất kỳ phần nào của nội dung vượt ra khỏi ranh giới của phần tử sẽ bị ẩn.

```
.hidden-overflow {
  overflow: hidden;
}
```

3. **scroll**: Nếu nội dung vượt ra khỏi ranh giới của phần tử, thanh cuộn sẽ xuất hiện để cho phép người dùng cuộn để xem nội dung đầy đủ.

```
.scroll-overflow {
  overflow: scroll;
}
```

4. **auto**: Thanh cuộn chỉ xuất hiện khi nội dung vượt quá ranh giới của phần tử.

```
.auto-overflow {
  overflow: auto;
}
```

11. 0bject-fit

- **object-fit** trong CSS được sử dụng để xác định cách hiển thị nội dung của một phần tử `<img>`, `<video>`, hoặc `<iframe>` trong kích thước của phần tử đó. Điều này làm cho nó có thể được điều chỉnh và cắt giữ sao cho vừa với kích thước của phần tử mà không làm biến đổi tỷ lệ khung hình.

1. **fill**: Mặc định. Nội dung sẽ được kéo rộng hoặc co lại để lấp đầy toàn bộ phần tử mà không giữ tỷ lệ khung hình.

```
.object-fit-fill {
  object-fit: fill;
}
```

2. **contain**: Nội dung sẽ được co lại hoặc kéo rộng để fit chính xác vào phần tử mà không làm biến đổi tỷ lệ khung hình. Phần nào của phần tử có thể trống lấp.

```
.object-fit-contain {
  object-fit: contain;
}
```

3. **cover**: Nội dung sẽ được kéo rộng hoặc co lại để lấp đầy toàn bộ phần tử mà không làm biến đổi tỷ lệ khung hình. Một phần của nội dung có thể bị cắt đi.

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

4. **none**: Nội dung sẽ không bị điều chỉnh, và sẽ hiển thị theo tỷ lệ khung hình gốc, có thể biến đổi hình dạng nếu không giữ kích thước chính xác.

```
.object-fit-none {
  object-fit: none;
}
```

5. **scale-down**: Nếu nội dung nhỏ hơn hoặc bằng kích thước phần tử, thì scale-down sẽ giống như contain. Ngược lại, nó sẽ hoạt động như none.

```
.object-fit-scale-down {
  object-fit: scale-down;
}
```

12. Background:

1. Thuộc tính `background` là một thuộc tính tổng hợp cho tất cả các thuộc tính liên quan đến nền của một phần tử.

```
div {
  background: #f00 url('hinh-anh.jpg') no-repeat center center;
  width: 200px;
  height: 200px;
}
```

2. `background-color`: Đặt màu nền cho phần tử.

```
div {
  background-color: yellow;
}
```

3. `background-image`: Sử dụng hình ảnh làm nền cho phần tử.

```
div {
  background-image: url('duong-dan-den-hinh-anh.jpg');
}
```

4. `background-repeat`: Xác định cách hình ảnh nền lặp lại.

```
div {
  background-repeat: no-repeat;
```

```
}
```

5. `background-position`: Đặt vị trí ban đầu của hình ảnh nền.

```
div {
  background-position: center top;
}
```

6. `background-size`: Đặt kích thước của hình ảnh nền.

```
div {
  background-size: cover;
}
```

7. `background-attachment`: Xác định liệu hình ảnh nền có di chuyển khi cuộn trang hay không.

```
div {
  background-attachment: fixed;
}
```

8. `background-origin`: Xác định điểm bắt đầu của hình ảnh nền (phần tử con, phần tử đệm, hoặc phần tử biên).

```
div {
  background-origin: content-box;
}
```

9. `background-clip`: Xác định phần nào của phần tử sẽ hiển thị màu nền (phần nội dung, phần đệm, hoặc phần biên).

```
div {
  background-clip: padding-box;
}
```

10. `background-blend-mode`: Đặt chế độ hỗn hợp (blend mode) cho hình ảnh nền và màu nền.

```
div {
  background-blend-mode: multiply;
}
```

13. Gradient:

- `gradient` là một cách để tạo hiệu ứng chuyển đổi màu từ một màu sang một màu khác hoặc từ một màu đến trong suốt. Có hai loại chính của gradient trong CSS: linear gradient (gradient tuyến tính) và radial gradient (gradient hình tròn).

1. Linear Gradient (Gradient Tuyến Tính):

Để tạo một linear gradient, bạn sử dụng thuộc tính background-image với giá trị là linear-gradient().

```
.element {
  background-image: linear-gradient(to bottom, #ffcc00, #ff6600);
}
```

2. Radial Gradient (Gradient Hình Tròn):

Radial gradient tạo hiệu ứng chuyển đổi màu từ tâm của hình tròn ra ngoài.

```
.element {
  background-image: radial-gradient(circle, #ffcc00, #ff6600);
}
```

- Ví dụ: https://uigradients.com/

14. Aspect ratio:

- Có thể đặt tỷ lệ khía cạnh (aspect ratio) cho một phần tử bằng cách sử dụng thuộc tính padding trong kết hợp với width hoặc height.

1. Sử dụng padding:

```
.container-1-1 {
  width: 100%;
  position: relative;
  padding-top: 100%; /* 1:1 (100% = 1/1) */
}
.container-3-4 {
  width: 100%;
  position: relative;
  padding-top: 75%; /* 4:3 (75% = 3/4) */
}
.container-16-9 {
  width: 100%;
  position: relative;
  padding-top: 56.25%; /* 16:9 (56.25% = 9/16) */
}
.child {
```

```
  position: absolute;
  top: 0;
  left: 0;
  width: 100%;
  height: 100%;
}
```

2. Sử dụng aspect-ratio (CSS4):

```
.aspect-ratio-16-9 {
  aspect-ratio: 16/9; /* 16:9 */
}

.aspect-ratio-4-3 {
  aspect-ratio: 4/3; /* 4:3 */
}

.aspect-ratio-1-1 {
  aspect-ratio: 1/1; /* 1:1 */
}
```

- Lưu ý: với sự tiện lợi của aspect-ratio, khi nó được hỗ trợ rộng rãi, đó sẽ là cách tốt nhất để thiết lập aspect ratio trong CSS.

https://caniuse.com/

14. min-width, min-height, max-width, max-height:

1. min-width:

`min-width`: Đặt giá trị tối thiểu cho chiều rộng của một phần tử. Phần tử không thể nhỏ hơn giá trị này.

```
.min-width {
    min-width: 200px;
}
```

2. min-height:

`min-height`: Đặt giá trị tối thiểu cho chiều cao của một phần tử. Phần tử không thể nhỏ hơn giá trị này.

```
.example {
    min-height: 100px;
}
```

3. max-width:

`max-width`: Đặt giá trị tối đa cho chiều rộng của một phần tử. Phần tử không thể lớn hơn giá trị này.

```
.example {
    max-width: 500px;
}
```

4. max-height:

`max-height`: Đặt giá trị tối đa cho chiều cao của một phần tử. Phần tử không thể lớn hơn giá trị này.

```
.example {
    max-height: 300px;
}
```