TẬP ĐOÀN CÔNG NGHIỆM VIỄN THÔNG QUÂN ĐỘI VIETTEL

**CÔNG TY ĐẦU TƯ CÔNG NGHỆ VIETTEL**



**BÁO CÁO THỰC TẬP**

**FRONT-END**

**Lý thuyết tuần 2**

**NGƯỜI HƯỚNG DẪN: MAI DUY HIẾU**

**THỰC TẬP SINH: ĐÀM THI LAN ANH**

**PHÒNG BAN: KỸ THUẬT CÔNG NGHỆ**

**Hà Nội- 2025**

**MỤC LỤC**

[PHẦN 1. SASS/SCSS là gì? 1](#_Toc200930668)

[1.1. Chọn cách viết theo SCSS 3](#_Toc200930669)

[1.2. Tóm tắt SCSS 3](#_Toc200930670)

[1.2.1. Biến 3](#_Toc200930671)

[1.2.2. Toán tử 4](#_Toc200930672)

[1.3. SCSS nesting 6](#_Toc200930673)

[1.4. SCSS extend 9](#_Toc200930674)

[1.5. SCSS @mixin & @include 10](#_Toc200930675)

[1.6. Placeholders 12](#_Toc200930676)

[1.7. @function 13](#_Toc200930677)

[1.8. Câu lệnh @if @else 15](#_Toc200930678)

[1.9. Vòng lặp @for 16](#_Toc200930679)

[1.10. Vòng lặp @while 17](#_Toc200930680)

[1.11. Vòng lặp @each 18](#_Toc200930681)

[1.12. @import 18](#_Toc200930682)

[PHẦN 2. Responsive là gì? 19](#_Toc200930683)

[2.1. Responsive viewport 20](#_Toc200930684)

[2.1.1. Cấu trúc 20](#_Toc200930685)

[2.1.2. Cách sử dụng 21](#_Toc200930686)

[2.2. Media queries là gì? 21](#_Toc200930687)

[2.2.1. Cấu trúc 21](#_Toc200930688)

[2.2.2. Ví dụ 22](#_Toc200930689)

[2.3. Ứng dụng Flexbox và CSS Grid trong Responsive Web Design 23](#_Toc200930690)

[2.3.1. Flexbox (Flexible Box Layout) 23](#_Toc200930691)

[2.3.2. CSS Grid Layout 23](#_Toc200930692)

[2.3.3. Bố cục responsive với Flexbox và CSS Grid 23](#_Toc200930693)

[2.3.4. Ví dụ minh họa 24](#_Toc200930694)

[PHẦN 3. Đơn vị đo lường trong thiết kế website 27](#_Toc200930695)

[3.1. Đơn vị tuyệt đối 27](#_Toc200930696)

[3.2. Đơn vị tương đối 27](#_Toc200930697)

[3.3. Nên dùng đơn vị nào? 28](#_Toc200930698)

# SASS/SCSS là gì?

#### **SASS** (Systematically Awesome Style Sheets) là một một ngôn ngữ kịch bản tiền xử lý, từ nó được chuyển đổi ra CSS bằng các công cụ biên dịch khác nhau. **SASS** có hai loại cú pháp. Cú pháp gốc ban đầu là loại cú pháp viết theo kiểu thụt đầu dòng giống như Haml, nó phân chia các khối code bằng ký tự xuống dòng và thụt đầu dòng, các file viết theo cú pháp này thường để phần mở rộng .sass

#### Cú pháp viết code **SASS** thứ hai là **SCSS** (Sassy CSS), nó sử dụng các định dạng các khối code giống như chính **CSS**. Nó sử dụng {} để bọc khối code và ký hiệu ; để chia dòng trong khối lệnh đó. Các code viết theo **SCSS** lưu trong file có phần mở rộng .css

#### **SASS**và **SCSS**về bản chất vấn đề là giống nhau, chỉ khác nhau ở cách viết**.**

###### Ví dụ file SASS

A screenshot of a computer program

AI-generated content may be incorrect.

###### Ví dụ file SCSS



###### Ví dụ file CSS

A screenshot of a computer

AI-generated content may be incorrect.

## Chọn cách viết theo SCSS

#### Khi sử dụng **SASS** bạn có thể tùy thích chọn viết theo cú pháp SASS hay SCSS, nếu chọn **SASS** thì có vẻ ngắn gọn, dễ đọc nhưng nó lại không kế thừa cú pháp CSS. Nên chọn cú pháp **SCSS** vì nó kế thừa chính **CSS**, nghĩa là một đoạn mã **CSS**, đưa vào **SCSS** thì nó chấp nhận luôn đoạn mã đó, nên bạn sẽ thấy rất quen thuộc. Tóm lại khi bạn viết **SASS** theo cú pháp **SCSS** thì bạn viết giống hệt **CSS** nhưng có thêm một vài mở rộng như biến, hàm, toán tử ...

## Tóm tắt SCSS

### Biến

#### Các khai báo và sử dụng biến trong SCSS khá giống với các các ngôn ngữ khác, cũng phân chia biến toàn cục (global) và biến cục bộ, cách sử dụng cũng tương tự.

#### Cách khai báo biến SCSS:



##### $name tên biến, đặt bất tên bất kỳ không khoảng trắng.

##### value giá trị của biến, là các giá trị CSS hoặc công thức.

##### Cẩn thận cú pháp sử dụng dấu 2 chấm : không phải dấu bằng =.

1. Biến toàn cục

#### Là biến có phạm vi tác động tới tất cả các vị trí trong file SCSS.

|  |  |
| --- | --- |
| SCSS | CSS |
| $a: 20px;  $b: $a + 30px;  #main {  margin-bottom: $a;  }  #footer {padding: $a; } | #main {  margin-bottom: 20px;  }  #footer {  padding: 20px;  } |

#### Ta thấy biến $a được sử dụng bên ngoài và cả bên trong các thành phần khác.

1. Biến cục bộ

#### Là biến có phạm vi tác động chỉ bên trong phạm vi được khai báo.

|  |  |
| --- | --- |
| SCSS | CSS |
| #main {  $a: 20px;  margin-bottom: $a;  }  #footer {  padding: $a;  } | Màn hình cmd báo lỗi. |

#### Ta thấy biến $a chỉ được sử dụng bên trong thành phần được khai báo là #main.

### Toán tử

###### Biểu thức quan hệ

|  |  |  |
| --- | --- | --- |
| Biểu thức | Giải thích | Ví dụ |
| == | Bằng | $a == $b |
| > | Lớn hơn | $a > $b |
| >= | Lớn hơn hoặc bằng | $a >= $b |
| <= | Nhỏ hơn hoặc bằng | $a <= $b |

###### Biểu thức số học

|  |  |  |
| --- | --- | --- |
| Biểu thức | Giải thích | Ví dụ |
| + | Cộng | $a + $b |
| - | Trừ | $a - $b |
| \* | Nhân | $a \* $b |
| / | Chia | $a / $b |
| % | Chia có số dư | $a % $b |

#### Note: Cẩn thận khi sử dụng biểu thức với giá trị khác đơn vị:

###### Biểu thức chuỗi

|  |  |
| --- | --- |
| Ví dụ | Kết quả |
| "Helvetica" + " Neue" | "Helvetica Neue" |
| sans- + serif | sans-serif |
| sans - serif | sans-serif |

###### Biểu thức đúng sai

|  |  |  |
| --- | --- | --- |
| Biểu thức | Ví dụ | Kết quả |
| true | $a: true | $a nhận giá trị đúng |
| not true | $a: not true | $a nhận giá trị sai |
| false | $a: false | $a nhận giá trị sai |
| not false | $a: not false | $a nhận giá trị đúng |
| true and true | $a: true; $b: true; $a and $b | Cho giá trị đúng |
| true and false | $a: true; $b: false; $a and $b | Cho giá trị sai |
| true or false | $a: true; $b: false; $a or $b | Cho giá trị đúng |
| false or false | $a: false; $b: false; $a or $b | Cho giá trị sai |

## SCSS nesting

#### SCSS nesting là cấu trúc viết gọn theo dạng phân cấp, giúp dễ dàng kiểm soát thành phần con thuộc thành phần cha nào rất tiện lợi.

#### Ngoài ra còn giúp kiểm soát code của thành phần theo các dạng màn hình khá tốt.

###### SCSS nesting – theo tag

#### Phân cấp cha con, thành phần bên trong sẽ là thành phần con.

|  |  |
| --- | --- |
| SCSS | CSS |
| section {  width: 300px;  h2 {  font-size: 12px;  }  div {  padding: 20px;  }  }  #main {  width: 300px;  h2 {  font-size: 12px;  }  .list {  padding: 20px;  li {background-color: #f1f1f1;}  }  } | section {  width: 300px;  }  section h2 {  font-size: 12px;  }  section div {  padding: 20px;  }  #main {  width: 300px;  }  #main h2 {  font-size: 12px;  }  #main .list {padding: 20px;}  #main .list li {  background-color: #f1f1f1;  } |

###### SCSS nesting – đại diện

#### Ký tự & sẽ đại diện cho thành phần cha bên ngoài gần nhất.

|  |  |
| --- | --- |
| SCSS | CSS |
| .btn-submit {  width: 100px;  background-color: red;  &:hover {  background-color: blue;  }  }  .list {  margin-bottom: 20px;  &.list-last {  margin-bottom: 0;  }  }  .list {  li {  float: left;  margin-right: 10px;  &:last-child {  margin-right: 0;  }  }  } | .btn-submit {  width: 100px;  background-color: red;  }  .btn-submit:hover {  background-color: blue;  }  .list {  margin-bottom: 20px;  }  .list.list-last {  margin-bottom: 0;  }  .list li {  float: left;  margin-right: 10px;  }  .list li:last-child {  margin-right: 0;  } |

###### SCSS nesting – thuộc tính

#### Các thuộc tính có tiền tố giống nhau, có thể được gom chung với nhau.

#### Một số thuộc tính có tiền tố giống nhau thường dùng:

##### font-

##### margin-

##### padding-

##### text-

##### background-

##### border-

##### ...

|  |  |
| --- | --- |
| SCSS | CSS |
| section {  width: 300px;  font: {  family: Arial, sans-serif;  size: 16px;  }  h2 {  font-size: 20px;  }  div {  padding: 20px;  }  } | section {  width: 300px;  font-family: Arial, sans-serif;  font-size: 16px;  }  section h2 {  font-size: 20px;  }  section div {  padding: 20px;  } |

###### SCSS nesting – @media

#### @media được sử dụng bên trong thành phần nào thì sẽ sử dụng cho thành phần đó.

|  |  |
| --- | --- |
| SCSS | CSS |
| #main {  width: 300px;  @media only screen and (min-width: 641px) { width: 200px; }  } | #main {  width: 300px;  }  @media only screen and (min-width: 641px) {  #main {width: 200px;}  } |

## SCSS extend

#### SCSS @extend, kế thừa những thuộc tính của thành phần khác đã được khai báo trước đó, giúp giảm thời gian code và trùng lặp code.

#### Ví dụ về SCSS extend:

|  |  |
| --- | --- |
| SCSS | CSS |
| .box {  background-color: #f1f1f1;  border: 1px solid #ccc;  }  .section {  @extend .box;  padding: 20px;  } | .box, .section {  background-color: #f1f1f1;  border: 1px solid #ccc;  }  .section {  padding: 20px;  } |

Note: Chú ý là thành phần kế thừa và được kế thừa phải tương thích nhau.

## SCSS @mixin & @include

#### Khác với @exclude là kế thừa thuộc tính của thành phần có sẵn, tuy nhiên @mixin & @include cung cấp một giải pháp tốt hơn nhiều, đó là tạo một bộ các thuộc tính, và chèn nó vào bên trong các thành phần muốn sử dụng, quan trọng là @mixin & @include cung cấp đối số có thể tùy chỉnh trong từng thành phần được sử dụng.

###### SCSS @mixin & @include không có đối số

|  |  |
| --- | --- |
| SCSS | CSS |
| @mixin reset-list {  margin: 0;  padding: 0;  list-style: none;  }  ul {  @include reset-list;  background-color: #f1f1f1;  li {  padding: 10px;  border: 1px solid #ccc;  }  } | ul {  margin: 0;  padding: 0;  list-style: none;  background-color: #f1f1f1;  }  ul li {  padding: 10px;  border: 1px solid #ccc;  } |

#### Ta thấy thành phần ul đã sử dụng các thuộc tính được khai báo tại @mixin reset-list.

###### SCSS @mixin & @include - đối số là giá trị

|  |  |
| --- | --- |
| SCSS | CSS |
| @mixin center($w) {  margin: 0 auto;  width: $w;  }  .main {  @include center(300px);  padding: 20px;  }  .box {  @include center(100px);  background-color: #f1f1f1;  } | .main {  margin: 0 auto;  width: 300px;  padding: 20px;  }  .box {  margin: 0 auto;  width: 100px;  background-color: #f1f1f1;  } |

#### Ta thấy cả 2 thành phần .main và .box đều đã sử dụng các thuộc tính được khai báo tại @mixin center, tuy nhiên với 2 đối số $w được truyền khác nhau.

###### SCSS @mixin & @include - đối số là thuộc tính

|  |  |
| --- | --- |
| SCSS | CSS |
| @mixin center($prop, $val) {  margin: 0 auto;  width: 300px;  #{$prop}: $val;  }  .main {  @include center(color,red);  padding: 20px;  }  .box {  @include center(font-size,24px);  background-color: #f1f1f1;  } | .main {  margin: 0 auto;  width: 300px;  color: red;  padding: 20px;  }  .box {  margin: 0 auto;  width: 300px;  font-size: 24px;  background-color: #f1f1f1;  } |

## Placeholders

#### Cách sử dụng SCSS placeholder khá giống với @mixin, tuy nhiên các thành phần sử dụng placeholders sẽ được gom chung vào cùng một bộ các thuộc tính.

###### Ví dụ về CSS placeholder

|  |  |
| --- | --- |
| SCSS | CSS |
| %reset-list {  margin: 0;  padding: 0;  background: none;  }  ul {  @extend %reset-list;  background-color: #f1f1f1;  li {  padding: 10px;  border: 1px solid #ccc;  }  }  div {  @extend %reset-list;  } | ul, div {  margin: 0;  padding: 0;  background: none;  }  ul {  background-color: #f1f1f1;  }  ul li {  padding: 10px;  border: 1px solid #ccc;  } |

#### Ta thấy thành phần ul và div đã được gom chung và sử dụng các thuộc tính khai báo tại %reset-list.

###### Placeholder không sử dụng đối số

#### Do tính chất gom sử dụng chung các thuộc tính, nên placeholder không sử dụng được cho đối số.

|  |  |
| --- | --- |
| SCSS | CSS |
| $center($w) {  margin: 0 auto;  width: $w;  }  .main {  @extend $center(300px);  padding: 20px;  }  .box {  @extend $center(100px);  background-color: #f1f1f1;  } | //cmd báo lỗi sai cấu trúc. |

## @function

#### SCSS @function, khá giống nhau với @mixin, tuy nhiên thay vì biên dịch CSS style như @mixin, thì @function sẽ trả về giá trị, thuộc tính, hoặc function hoặc thậm chí là một @mixin.

###### Ví dụ về SCSS funtion

#### Function tính chiều rộng các phần tử con.

|  |  |
| --- | --- |
| SCSS | CSS |
| @function getWidth($widthWarp, $item) {  @return $widthWarp / $item;  }  .list {  width: 600px;  li {  float: left;  width: getWidth(600px,4);  } | .list {  width: 600px;  }  .list li {  float: left;  width: 150px;  } |

#### Function tính chiều rộng các phần tử con theo phần trăm.

|  |  |
| --- | --- |
| SCSS | CSS |
| @function getWidth($item) {  @return 100% / $item;  }  .two-cols {  width: columns(2);  }  .three-cols {  width: columns(3);  }  .four-cols {  width: columns(4);  }  .five-cols {  width: columns(5);  } | .two-cols {  width: 50%; }  .three-cols {  width: 33.33333%; }  .four-cols {  width: 25%; }  .five-cols {  width: 20%; } |

## Câu lệnh @if @else

#### SCSS câu lệnh @if @else rất tiện lợi để thiết lập sẵn bộ template dùng chung cho nhiều vị trí.

###### Ví dụ về SCSS @if @else

**Mẫu dành cho position**

Ta sẽ thử xây dựng một @mixin của thuộc tính position, sẽ xác định theo tên vị trí $pos, và giá trị trục $y và trục $x tương ứng.

|  |  |
| --- | --- |
| SCSS | CSS |
| @mixin position($pos,$y,$x) {  position: absolute;  @if $pos == 'tl' {  left: $x;  top: $y;  }@else if $pos == 'tr' {  right: $x;  top: $y;  }@else if $pos == 'bl' {  left: $x;  bottom: $y;  }@else if $pos == 'br' {  right: $x;  bottom: $y;  }  }  #main {  position: relative;  .box01 {  @include position('tl',20px,40px)  }  .box02 {  @include position('bl',50px,30px)  }  } | #main {  position: relative;  }  #main .box01 {  position: absolute;  left: 40px;  top: 20px;  }  #main .box02 {  position: absolute;  left: 30px;  bottom: 50px;  } |

* 'tl': top left
* 'tr': top right
* 'bl': bottom left
* 'br': bottom right

Ta thấy các với các giá trị khai báo vị trí như trên ta có thể sử dụng rất gọn thuộc tính position.

## Vòng lặp @for

#### SCSS @for thường được dùng để tác động tới danh sách, với tính chất tăng dần.

###### Ví dụ về SCSS vòng lặp @for

Tạo độ giảm dần opacity

|  |  |
| --- | --- |
| SCSS | CSS |
| $lenght: 5;  @for $i from 1 through $lenght {  .item-#{$i} {  opacity: 1/$i;  }  } | .item-1 {  opacity: 1;  }  .item-2 {  opacity: 0.5;  }  .item-3 {  opacity: 0.33333;  }  .item-4 {  opacity: 0.25;  }  .item-5 {  opacity: 0.2;  } |

Nội dung trên có nghĩa là: lặp cho nội dung bên trong @for sao cho $i chạy từ 1 cho tới $length.

## Vòng lặp @while

#### SCSS @while sẽ lặp nội dung bên trong while trong khi vẫn còn thỏa điều kiện ban đầu.

###### Ví dụ về vòng lặp @while

Thực hiện lặp theo số lần từ 1 tới giá trị cho trước

|  |  |
| --- | --- |
| SCSS | CSS |
| $i: 4;  @while $i > 0 {  .item-#{$i} {  width: 800px/$i;  $i: $i - 1;  }  } | .item-4 {  width: 200px;  }  .item-3 {  width: 266.66667px;  }  .item-2 {  width: 400px;  }  .item-1 {  width: 800px;  } |

Thực hiện lặp theo số lần từ giá trị bất kỳ

|  |  |
| --- | --- |
| SCSS | CSS |
| $i: 3;  @while $i < 5 {  .item-#{$i} {  color: red;  $i : $i + 1;  }  } | .item-3 {  color: red;  }  .item-4 {  color: red;  } |

## Vòng lặp @each

SCSS @each tiện lợi cho việc thiết lập một loạt các giá trị cùng tính chất nhau, đặc biệt là trong danh sách với các thuộc tính gần như là như nhau.

## @import

#### SCSS cung cấp một quy tắc @import thông qua quá trình biên dịch, không như @import của CSS yêu cầu request HTTP từ trình duyệt.

#### Việc import SCSS sẽ đồng thời kết hợp các công thức nhận được để biên dịch thành file CSS.

#### Việc import SCSS tuân thủ một số quy tắc:

###### File import sẽ có dạng dấu gạch dưới ở trước tên file \_name.scss.

###### Nội dung import chỉ chứa name mà không có dấu gạch \_ và định dạng file .scss.

# Responsive là gì?

#### Responsive là thuật ngữ dùng để chỉ một trang web có thể hiển thị tốt trên các thiết bị khác nhau, bao gồm: PC (Personal Computer - máy tính cá nhân bao gồm máy tính bàn và laptop), Tablet (máy tính bảng), SP (Smartphone - điện thoại thông minh).

#### Hiển thị tốt ở đây là trang web không bị vỡ giao diện, mà giao diện sẽ linh hoạt điều chỉnh sao cho phù hợp với từng thiết bị khác nhau.

#### Linh hoạt giao diện có thể là:

##### Chiều rộng của các thành phần thay đổi, co giản theo từng màn hình khác nhau.

##### Hình ảnh thay đổi cho phù hợp hơn, ví dụ khi xem trên PC thì chữ trong hình nhỏ vừa đủ coi, trên SP thì sẽ sử dụng hình với nội dung chữ lớn hơn cho phù hợp với các thiết bị điện thoại.

##### Vị trí các thành phần thay đổi, ví dụ nội dung bên phần sidebar (aside) không quan trọng như các phần khác, thì sẽ di chuyển xuống phía dưới.

##### Mục (item) của danh sách (list) trên một hàng cũng cần thay đổi cho phù hợp, ví dụ trên PC thì là 4 mục, nhưng trên SP thì hiển thị 2 hoặc 1 mục để cho dễ nhìn.

##### Nhiều thay đổi khác ...

A screenshot of a computer

AI-generated content may be incorrect.

## Responsive viewport

#### Nội dung HTML sẽ đối ứng với màn hình giao diện PC, nên khi xem trên các thiết bị như Tablet hay SP, nội dung sẽ hiển thị không như ý muốn, cụ thể là nội dung bị phóng to hoặc thu nhỏ với tỷ lệ khác đi, để giải quyết vấn đề này ta dùng khai báo viewport cho file HTML.

#### Chúng ta hãy thử so sánh hiển thị file HTML trên các thiết bị khi chưa sử dụng và có sử dụng viewport.



### Cấu trúc

#### Nội dung viewport được khai báo bằng thẻ <meta> với các giá trị tương ứng sau:

|  |
| --- |
| <meta name="viewport" content="width=device-width, maximum-scale=1.0, initial-scale=1.0, user-scalable=no"> |

#### Ngoài ra tùy vào yêu cầu, ta còn có thể sử dụng các giá trị khác:

|  |  |
| --- | --- |
| Giá trị | Mô tả |
| width=device-width | Chiều rộng bằng chiều rộng của thiết bị. |
| height=device-height | Chiều cao bằng chiều cao của thiết bị. |
| maximum-scale=1.0 | Tỷ lệ lớn nhất (1.0 ứng với 100%). |
| minimum-scale=1.0 | Mức phóng to tối thiểu của thiết bị (1.0 ứng với 100%). |
| initial-scale=1.0 | Độ phóng to ban đầu (1.0 ứng với 100%). |
| user-scalable=no | Cho phép người dùng phóng to màn hình hay không (giá trị yes hoặc no) |

### Cách sử dụng

#### Nội dung viewport được viết bên trong tag <head> của file HTML.

A computer screen shot of a computer code

AI-generated content may be incorrect.

#### Khai báo thẻ meta trong phần head

## Media queries là gì?

#### Để xử lý thay đổi giao diện giữa các dạng màn hình khác nhau (chiều rộng màn hình khác nhau), chúng ta sẽ sử dụng truy vấn @media.

### Cấu trúc

Truy vấn @media được viết trong file CSS, với cấu trúc như sau:

|  |
| --- |
| @media only|not Media-type and (Media-future and|or|not Media-future) {  tag {  property: value;  }  } |

###### Media-type

|  |  |
| --- | --- |
| Giá trị | Mô tả |
| screen | Dành cho trang hiển thị (không dành cho bản in). |
| print | Dành cho bản in (không dành cho trang hiển thị). |
| all | Mặc định, dành cho trang hiển thị và cả bảng in. |
| speech | Dành cho phiên bản hỗ trợ đọc thành tiếng. |

###### Media-future

|  |  |
| --- | --- |
| Giá trị | Mô tả |
| max-width | Chiều rộng lớn nhất màn hình của thiết bị, tức là những thiết bị có màn hình nhỏ hơn max-width sẽ bị ảnh hưởng source code. |
| min-width | Chiều rộng nhỏ nhất màn hình của thiết bị, tức là những thiết bị có màn hình lớn hơn min-width sẽ bị ảnh hưởng source code. |

### Ví dụ

|  |
| --- |
| div {  background-color: blue;  }  @media only screen and (max-width: 1024px) {  div {  background-color: red;  }  } |

#### Nội dung code trên như sau:

##### Tất cả các thiết bị sẽ có background-color: blue, riêng các thiết bị có màn hình nhỏ hơn 1024px sẽ có background-color: red.

##### only screen: chỉ dành cho trang hiển thị (không dành cho bản in (print)).

##### max-width: chỉ tác dụng cho chiều rộng màn hình lớn nhất là 1024px, tức là những màn hình nào nhỏ hơn 1024px đều sẽ bị ảnh hưởng code bên trong, cụ thể là div sẽ có background-color: red.

## Ứng dụng Flexbox và CSS Grid trong Responsive Web Design

#### Responsive Web Design (thiết kế web đáp ứng) là phương pháp xây dựng giao diện có khả năng **tự điều chỉnh và thích nghi với nhiều kích thước màn hình khác nhau**, từ điện thoại, máy tính bảng đến máy tính để bàn. Trong quá trình phát triển giao diện, **Flexbox** và **CSS Grid** là hai công cụ mạnh mẽ được sử dụng phổ biến để hỗ trợ việc tạo bố cục linh hoạt, hiện đại và dễ kiểm soát.

### Flexbox (Flexible Box Layout)

#### Flexbox là mô hình bố cục một chiều (theo hàng hoặc cột), cho phép phân phối không gian giữa các phần tử trong container một cách linh hoạt. Flexbox rất hữu ích để căn chỉnh các phần tử con, sắp xếp lại chúng khi thay đổi kích thước màn hình mà không cần float hay sử dụng position phức tạp.

#### Tính năng nổi bật:

###### Hỗ trợ căn giữa theo cả chiều ngang và dọc.

###### Cho phép phần tử tự động co giãn.

###### Dễ dàng sắp xếp lại thứ tự phần tử khi responsive.

### CSS Grid Layout

#### CSS Grid là hệ thống bố cục hai chiều (hàng và cột), cho phép chia trang thành các vùng rõ ràng. Grid rất mạnh mẽ khi tạo các giao diện phức tạp như dashboard, layout tạp chí, blog hoặc e-commerce.

#### Tính năng nổi bật:

###### Tạo bố cục với hàng và cột rõ ràng.

###### Dễ kiểm soát kích thước và vị trí của từng thành phần.

###### Kết hợp tốt với media queries để thay đổi cấu trúc layout theo độ rộng màn hình.

### Bố cục responsive với Flexbox và CSS Grid

|  |  |  |
| --- | --- | --- |
| Công cụ | Dùng cho | Ưu điểm nổi bật |
| Flexbox | Bố cục đơn giản một chiều (menu, thẻ,...) | Dễ sử dụng, căn chỉnh linh hoạt |
| CSS Grid | Bố cục hai chiều (sidebar, content, ...) | Quản lý layout phức tạp hiệu quả hơn |

### Ví dụ minh họa

###### Responsive với Flebox và CSS Grid

#### Cấu trúc:

A screenshot of a computer

AI-generated content may be incorrect.

#### File index.html:

A screen shot of a computer program

AI-generated content may be incorrect.

###### File main.scss

\* {

    box-sizing: border-box;

    margin: 0;

    padding: 0;

}

html,

body {

    height: 100%;

}

body {

    font-family: Arial, sans-serif;

    line-height: 1.6;

    display: flex;

    flex-direction: column;

}

// Header và Footer

header,

footer {

    background-color: #2c3e50;

    color: white;

    text-align: center;

    padding: 20px;

}

// Navbar với Flexbox

*.navbar* {

    display: flex;

    justify-content: space-around;

    background-color: #34495e;

    flex-wrap: wrap;

    a {

        color: white;

        padding: 15px;

        text-decoration: none;

        &*:hover* {

            background-color: #1abc9c;

        }

    }

}

// Layout chính với CSS Grid

main {

    flex: 1;

}

*.layout* {

    display: grid;

    grid-template-columns: 1fr 3fr;

    gap: 20px;

    padding: 20px;

}

*.sidebar* {

    background-color: #ecf0f1;

    padding: 20px;

}

*.content* {

    background-color: #f8f9fa;

    padding: 20px;

}

// Responsive

@media (max-width: 768px) {

*.layout* {

        grid-template-columns: 1fr;

    }

}

@media (max-width: 600px) {

*.navbar* {

        flex-direction: column;

        align-items: center;

    }

}

#### Cách chạy:

###### Cài SCSS nếu chưa có: npm install -g sass

###### Chạy lệnh để biên dịch: sass styles/main.scss styles/main.css

#### Giao diện code:

A screenshot of a computer

AI-generated content may be incorrect.

# Đơn vị đo lường trong thiết kế website

## Đơn vị tuyệt đối

#### Các đơn vị này không thay đổi theo môi trường trình duyệt hay thiết bị:

|  |  |  |
| --- | --- | --- |
| Đơn vị | Ý nghĩa | Ví dụ |
| px | Pixel (điểm ảnh) | width: 200px |
| cm | Centimet (xăng-ti-mét) | margin: 2cm |
| mm | Millimet (mi-li-mét) | padding: 10mm |
| in | Inch (1in = 2.54cm) | height: 1in |
| pt | Point (1pt = 1/72 inch) | font-size: 12pt |
| pc | Pica (1pc = 12pt) | font-size: 1pc |

Thực tế, px là phổ biến nhất trong các đơn vị tuyệt đối.

## Đơn vị tương đối

#### Phụ thuộc vào các yếu tố như kích thước phần tử cha, cửa sổ trình duyệt, font chữ mặc định,...

|  |  |  |
| --- | --- | --- |
| Đơn vị | Ý nghĩa | Ví dụ |
| % | Phần trăm của phần tử cha | width: 50% |
| em | So với font-size của phần tử cha | font-size: 2em |
| rem | So với font-size của phần tử gốc (html) | font-size: 1.5rem |
| vw | Viewport Width (1vw = 1% chiều rộng trình duyệt) | width: 80vw |
| vh | Viewport Height (1vh = 1% chiều cao trình duyệt) | height: 50vh |
| vmin | Giá trị nhỏ hơn giữa vw và vh | font-size: 5vmin |
| vmax | Giá trị lớn hơn giữa vw và vh | font-size: 5vmax |
| ch | Chiều rộng của ký tự "0" trong font hiện tại | width: 40ch |

## Nên dùng đơn vị nào?

|  |  |
| --- | --- |
| Mục đích | Gợi ý đơn vị phổ biến |
| Kích thước cố định | px |
| Kích thước linh hoạt | %, em, rem, vw, vh |
| Font chữ linh hoạt | em, rem |
| Responsive (đa thiết bị) | rem, vw, vh, % |

#### Khi làm giao diện reponsive , ưu tiên:

###### rem hoặc em cho kích thước chữ

###### %, vw, vh cho layout

###### Tránh dùng quá nhiều px vì sẽ làm mất khả năng tự co giãn.