

# Chương 2: CSS (Cascading Style Sheets)

Giảng Viên: ThS. Tạ Việt Phương

1

1

---

---

---

---

---

---

---

---

## CSS Cascading Style Sheets

Phát triển ứng dụng Web 2

2

---

---

---

---

---

---

---

---

### Nội dung

- Giới thiệu CSS
- Thành phần trong CSS
- Selector
- Box model
- Font và text

Phát triển ứng dụng Web 3

3

---

---

---

---

---

---

---

---

## Giới thiệu CSS



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web

4

4

---

---

---

---

---


---

---

---

## Giới thiệu CSS

- CSS = Cascading Style Sheets**
- Dùng để mô tả cách trình bày, hiển thị các thành phần trên trang Web được tạo bằng HTML
- Sử dụng tương tự như dạng TEMPLATE
- Có thể sử dụng lại cho các trang web khác
- Có thể thay đổi thuộc tính từng trang hoặc cả site nhanh chóng (cascading)



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web

5

5

---

---

---

---

---


---


---

---

## Giới thiệu CSS

- CSS là **một chuẩn để định dạng** được sử dụng để tìm và định dạng lại các phần tử được tạo ra bởi các ngôn ngữ đánh dấu (HTML, XHTML và XML...)
- HTML là cái khung, sườn của website.
- CSS là những thứ trang trí màu sắc, bố cục,... trên website đó.
- Tương tự như việc trang trí nội thất:





TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web

6

6

---

---

---

---

---

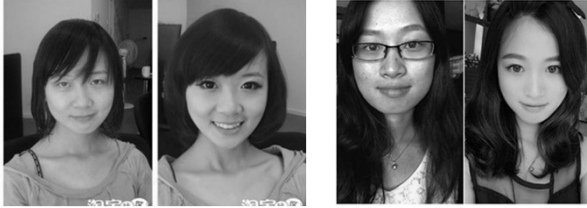
---

---

---

## Giới thiệu CSS

- Hoặc tương tự với việc trang điểm của con người



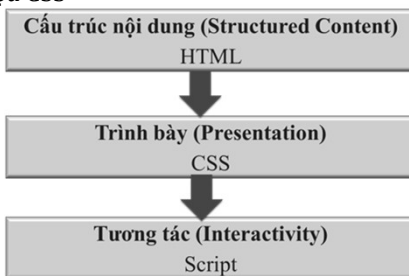
7

## Giới thiệu CSS

- Thuộc tính CSS được bổ sung vào HTML và **không phá vỡ cấu trúc** của HTML sẵn có.
- CSS làm **tăng sự nhất quán** về định dạng và hiệu năng tải trang web.
- **Thống nhất cách thể hiện và tái sử dụng** cho nhiều webpage trong website. Có thể thay đổi thuộc tính từng trang hoặc cả site nhanh chóng -> linh hoạt thay đổi cách thể hiện.
- Tập tin CSS chỉ được **tải ở lần đầu tiên** khi truy cập trang web


8

## Giới thiệu CSS



9

## Giới thiệu CSS



• HTML • CSS

Phát triển ứng dụng Web 10

10

---

---

---

---

---

---

---

---

## Thành phần trong CSS

Phát triển ứng dụng Web 11

11

---

---

---

---

---

---

---

---

## Cú pháp

Vùng chọn {

Thuộc tính 1: giá trị 1;

Thuộc tính 2: giá trị 2;

.....

Thuộc tính N: giá trị N;

}

- Vùng chọn: tên các **tag**, **id** hoặc **class** của tag
  - <h1>, <div id="div1">, <h1 class="TieuDe1">
- Thuộc tính: width, background-color, position, font
 

```
h1{
    font-weight: bold;
    font-size: 16pt;
    color: white;
    font-style: italic;
}
```

Phát triển ứng dụng Web 12

12

---

---

---

---

---

---

---

---

## Ghi chú

- Giống Ghi chú trong C++
- Sử dụng /\* Ghi chú \*/
- Ví dụ:

```
SelectorName
{
    Thuộc tính 1: giá trị 1; /*Ghichu1*/
    Thuộc tính 2: giá trị 2; /*Ghichu2*/
    .....
    Thuộc tính n: giá trị n;
}
```

13

---

---

---

---

---

---

---

---

## Phân loại CSS

- Gồm 3 loại:
  - Inline Style Sheet
  - Embedding Style Sheet (còn gọi là Internal Style Sheet)
  - External Style Sheet

14

---

---

---

---

---

---

---

---

## Inline Style Sheet

- Định nghĩa style trong thuộc tính style của từng thẻ HTML
- Cú pháp

```
<tag style="property1: value1;...property N: value N;">...
</tag>
```

- Ví dụ: `<h1 style="color: yellow">This is yellow </h1>`

```
<b style="color: navy;">Màu xanh nước biển.</b>
```

15

---

---

---

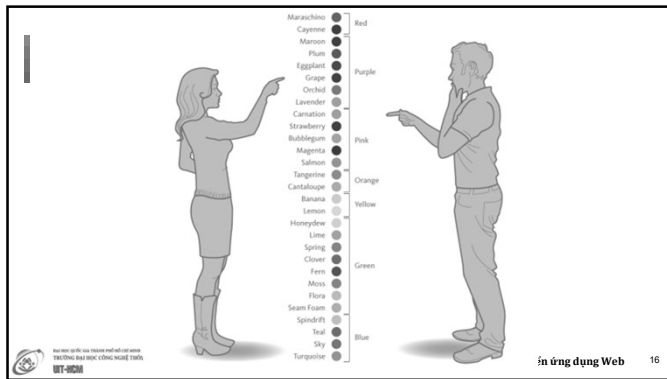
---

---

---

---

---



16

---

---

---

---

---

---

---

---

---

---

## Embedding Style Sheet

- Những thuộc tính css trong thẻ **<style>** của trang HTML
- Cú pháp
 

```
<head>
  <style type="text/css" >
    TagName{
      property 1:value1;
      property 2:value2;
      .....
      property N: valueN;
    }
  </style>
</head>
```

17

---

---

---

---

---

---

---

---

---

---

## Ví dụ

```
<HTML>
<HEAD>
<STYLE TYPE="text/css">
  p{color: green;font-size: 12pt;font-family: Arial;}
  h2{color: Red;}
</STYLE>
</HEAD>
<BODY BGCOLOR="#FFFF00">
<h1 style="color:blue;margin-left:30px;">This is a heading.</h1>
<h2>This is red</h2>
<p>this is green, 12 pt. and Garamond.</p>
</BODY>
</HTML>
```

18

---

---

---

---

---

---

---

---

---

---

## External Style Sheet

- Mọi style đều lưu trong file có phần mở rộng là \*.css (được sử dụng phổ biến)
- Định nghĩa style theo dạng **Embedding Style Sheet**
- Tạo liên kết đến file CSS
  - Liên kết bằng thẻ **<link>**.
 

```
<head>
  <link rel="stylesheet" href="URL" type="text/css" >
</head>
```
  - Liên kết bằng thẻ **<style>** với @import url.
 

```
<head>
  <style type="text/css" media="all|print|screen" >
    @import url(URL);
  </style>
</head>
```



19

---

---

---

---

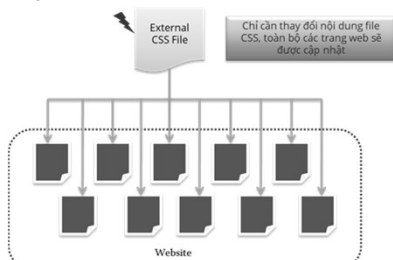
---

---

---

---

## External Style Sheet



20

---

---

---

---

---

---

---

---

## Ví dụ External Style Sheet

- Tạo file style.css

```
H2{
    FONT-WEIGHT: bold;
    FONT-SIZE: 16pt;
    COLOR: white;
    FONT-STYLE: italic;
    FONT-FAMILY: Arial;
    BACKGROUND-COLOR: red;
    font-color: white
}
```



21

---

---

---

---

---

---

---

---

## Ví dụ External Style Sheet

- Sử dụng **style.css** trong trang HTML

```
<html>
  <head>
    <title>Cascading Style Sheets</title>
    <link rel="stylesheet" href="style.css" type="text/css" >
  </head>
  <body>
    <h2>This is an H2 </h2>
  </body>
</html>
```

22

---

---

---

---

---

---

---

---

## So sánh, đánh giá

	Inline style sheet	Embedding style sheet	External style sheet
Khái báo	Kiểu 1	Kiểu 2	Kiểu 3
Cú pháp	<pre>&lt;p style="color:red; text-align:center"&gt;   BHCNTT &lt;/p&gt;</pre>	<pre>&lt;style type="text/css"&gt; .tieudel{color:red;} &lt;/style&gt; &lt;p class="tieudel"&gt;   BHCNTT &lt;/p&gt;</pre>	<pre>&lt;link rel="stylesheet" href="style.css"&gt; &lt;p class="tieudel"&gt;   BHCNTT &lt;/p&gt;</pre>
Ưu điểm	<ul style="list-style-type: none"> <li>Dễ quản lý style theo từng tag</li> </ul>	<ul style="list-style-type: none"> <li>Dễ quản lý style theo từng tài liệu web</li> <li>Không cần thêm các trang thông tin khác cho style</li> </ul>	<ul style="list-style-type: none"> <li>Thiết lập style cho nhiều tài liệu</li> <li>Thông tin các style được trình duyệt cache lại</li> </ul>
Khuyết điểm	Cần khai báo style trong từng tag của tài liệu	Cần khai báo lại style lại cho các trang khác	<ul style="list-style-type: none"> <li>Tốn thời gian download file .css -&gt; làm chậm quá trình biên dịch web ở trình duyệt trong lần đầu tiên sử dụng</li> </ul>

23

---

---

---

---

---

---

---

---

## Độ ưu tiên

- Thứ tự độ ưu tiên áp dụng định dạng style dùng trong các trang web (Độ ưu tiên giảm dần)
  - Inline style sheet
  - Embedding style sheet
  - External style sheet
  - Browser Default



24

---

---

---

---

---

---

---

---



Selector

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT- HCM

Phát triển ứng dụng Web25

25

---

---

---

---

---

---

---

---

Selector và phạm vi ảnh hưởng

- Là tên 1 style tương ứng với một thành phần được áp định dạng
- Ví dụ:

```
.TieuDel {
  color: red;
  font-family: Verdana, sans-serif;
}

<h1 class="TieuDel"> ĐHCNTT</h1>
```

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT- HCM

Phát triển ứng dụng Web26

26

---

---

---

---

---

---

---

---

Selector và phạm vi ảnh hưởng (tt)

Loại	Mô tả phạm vi ảnh hưởng	Ví dụ
Element	Định dạng áp dụng cho nội dung tất cả các tag element trong tài liệu Web	h1{color:red} /*nội dung của thẻ <h1> bị định dạng màu chữ đỏ*/
#id	Định dạng áp dụng cho Nội dung tất cả các tag có thuộc tính id trong tài liệu Web	#test {color: green;} /* ND của bất kỳ tag có thuộc tính id=test đều bị định dạng màu chữ=xanh lá*/

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT- HCM

Phát triển ứng dụng Web27

27

---

---

---

---

---

---

---

---

9

## Selector và phạm vi ảnh hưởng (tt)

<b>.class</b>	Định dạng áp dụng cho tất cả các tag có thuộc tính class trong tài liệu Web	<code>.note {color: red;}</code> /* ND của bất kỳ tag có thuộc tính <b>class=note</b> đều bị định dạng màu chữ=đỏ */
<b>element.class</b>	Định dạng áp dụng cho nội dung tag <b>Element</b> có thuộc tính class tương ứng	<code>h1.note {text-decoration: underline;}</code> /*ND của các thẻ <h1> có thuộc tính <b>class=note</b> đều bị định dạng gạch chân */

28

---

---

---

---

---

---

---

---

---

---

## Ví dụ - element

```

<html>
<head>
<title>VD CSS</title>
<style type="text/css">
  p{color:red}
  em{color:blue}
</style>
</head>
<body>
  <p>DHCNTT</p>
  <p>He thong thong tin<em>csdl</em></p>
</body>
</html>

```



29

---

---

---

---

---

---

---

---

---

---

## Ví dụ - element

```

<head>
<style type="text/css">
  p{color:red}
  em{color:blue}
</style>
</head>
<body>
  <p>Bạn à, sống đẹp lên! </p>
  <em> Chúng ta hãy sống đẹp như những con thiên nga của
  Tchaikovsky </em>
  <p> Ôi! Sống đẹp là thế nào hồi bạn...? </p>
</body>
</html>

```

Bạn à, sống đẹp lên!

*Chúng ta hãy sống đẹp như những con thiên nga của Tchaikovsky*

Ôi! Sống đẹp là thế nào hồi bạn...?

30

---

---

---

---

---

---

---

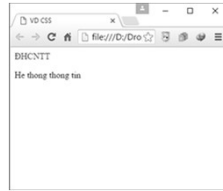
---

---

---

### Ví dụ - id

```
<html>
<head>
<title>VD CSS</title>
<style type="text/css">
    #id1{color:red}
    #id2{color:blue}
</style>
</head>
<body>
    <p id="id1">ĐHCNTT</p>
    <p id="id2">He thong thong tin</p>
</body>
</html>
```



31

---

---

---

---

---

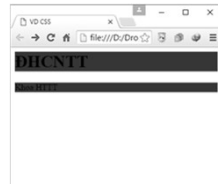
---

---

---

### Ví dụ - class

```
<html>
<head>
<title>VD CSS</title>
<style type="text/css">
    .maunen{background-color:red;}
</style>
</head>
<body>
    <h1 class="maunen">ĐHCNTT</h1>
    <p class="maunen">Khoa CNTT</p>
</body>
</html>
```



32

---

---

---

---

---

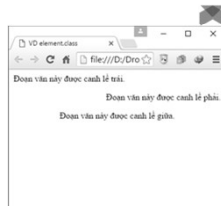
---

---

---

### Ví dụ - element.class

```
<html>
<head>
<title>VD element.class</title>
<style type="text/css">
    p.trai {text-align: left}
    p.phai {text-align: right}
    p.giua {text-align: center}
</style>
</head>
<body>
    <p class="trai">Đoạn văn này được canh lề trái.</p>
    <p class="phai">Đoạn văn này được canh lề phải.</p>
    <p class="giua">Đoạn văn này được canh lề giữa.</p>
</body>
</html>
```



33

---

---

---

---

---

---

---

---

## Pseudo-elements

- Pseudo-element: được dùng để chỉ rõ style cho một phần nào đó của phần tử được chọn.
- Ví dụ:
  - Style cho từ đầu tiên, dòng đầu tiên, hoặc là thành phần đầu tiên
  - Thêm nội dung vào trước hoặc sau một thành phần nào đó
- Cú pháp:

```
selector::pseudo-element {
  property:value;
}
```

34

---

---

---

---

---

---

---

---

## ::first-line Pseudo-elements

- Được dùng để chọn dòng đầu tiên của văn bản
- Ví dụ:

```
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

- Các thuộc tính có thể dùng:
  - font properties
  - color properties
  - background properties
  - word-spacing
  - letter-spacing
  - text-decoration
  - vertical-align
  - text-transform
  - line-height
  - clear

35

---

---

---

---

---

---

---

---

## ::first-letter Pseudo-elements

- Được dùng để chọn ký tự đầu tiên trong đoạn văn bản
- Ví dụ:

```
p::first-letter {
  color: #ffff00;
  font-size: xx-large;
}
```

- Các thuộc tính có thể dùng:
  - font properties
  - color properties
  - background properties
  - margin properties
  - padding properties
  - border properties
  - text-decoration
  - vertical-align ("float" hoặc "none")
  - text-transform
  - line-height
  - float
  - clear

36

---

---

---

---

---

---

---

---

## ::before và ::after Pseudo-elements

- ::before được dùng để chèn nội dung vào phía trước của phần tử
- Ví dụ:

```
h1::before {
  content: url(hinh.gif);
}
```

- ::after được dùng để chèn nội dung vào phía sau của phần tử

```
h1::after {
  content: url(hinh.gif);
}
```



37

[illegible]

## ::selection Pseudo-elements

- ::selection sẽ chọn đoạn văn bản được bôi đen bởi người dùng
- Ví dụ:

```
::selection {
    color: red;
    background: yellow;
}
```

- Các thuộc tính:
  - Color
  - Background
  - Cursor
  - Outline.



38

---

---

---

---

---

---

## Pseudo-elements và class

- Ví dụ:

```
p.trai::first-letter {
  color: #ffff00;
  font-size: xx-large;
}
```



39

[illegible]

## Kết hợp nhiều Pseudo-element

- Ví dụ:

```
p::first-letter {
  color: #ffff00;
  font-size: xx-large;
}
p::first-line {
  color: #ff0000;
  font-variant: small-caps;
}
```

40

---

---

---

---

---

---

---

---

## Pseudo-classes

- Pseudo-classes: được dùng để chỉ rõ trạng thái đặc biệt của một phần tử được chọn.

- Ví dụ:

- Style cho phần tử khi rê chuột vào (mouse over)
- Style cho đường link khi chưa click hoặc (unvisited hoặc visited)
- Style khi focus

- Cú pháp:

```
selector:pseudo-class {
  property:value;
}
```

41

---

---

---

---

---

---

---

---

## Pseudo-classes của thẻ <a>

```
/* unvisited link */
a:link {
  color: #FF0000;
}
/* visited link */
a:visited {
  color: #00FF00;
}
/* mouse over link */
a:hover {
  color: #FF00FF;
}
/* selected link */
a:active {
  color: #0000FF;
}
```

- Lưu ý:

- a:hover nên đứng ở dưới a:link và a:visited
- a:active nên đứng dưới a:hover

42

---

---

---

---

---

---

---

---

## :first-child Pseudo-classes

- Được dùng định dạng thành phần đầu tiên của nội dung
- Ví dụ: thẻ <p> đầu tiên trong phần nội dung sẽ được định dạng

```
p:first-child {
  color: red;
}
```

- Ví dụ:

- Tất cả các thẻ <i> đầu tiên của thẻ <p> đều được định dạng

```
p i:first-child {
  color: blue;
}
```

- Tất cả các thẻ <i> trong thẻ <p> đầu tiên đều được định dạng

```
p:first-child i {
  color: blue;
}
```

43

---

---

---

---

---

---

---

---

## Pseudo-class và class

```
a.highlight:hover {
  color: #ff0000;
}
```

```
<p><a class="highlight" href="">Pseudo-classes và classes</a></p>
```

44

---

---

---

---

---

---

---

---

## Box model

45

---

---

---

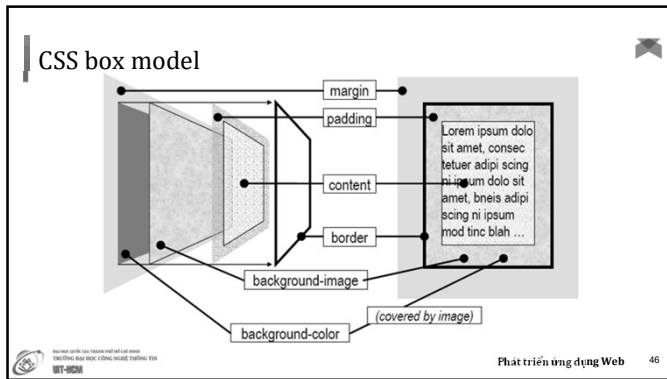
---

---

---

---

---



46

---

---

---

---

---

---

---

---

### Background

- Là thành phần nằm sau phần nội dung như là đoạn văn, hình ảnh
- Mở rộng đến phần border, bao gồm cả padding nhưng không vượt quá border.
- Các thuộc tính:
  - **background-color**: [colour-rgb], [colour-hex], [colour-name], transparent
  - **background-image**: [url()], none
    - VD: `body { background-image: url(tiles.gif); }`
  - **background-position**: top, bottom, left, right, center, [x-% y-%], [x-pos y-pos]
    - VD: `background-position: 50% 30px;`
  - **background-repeat**: repeat, repeat-x, repeat-y, no-repeat.
    - Dùng để lặp lại hình ảnh theo chiều ngang hoặc chiều dọc

Phát triển ứng dụng Web 47

47

---

---

---

---

---

---

---

---

### Background

- **background-attachment**: scroll, fixed.
  - VD:
 

```
div {
    background-image: url(flowers.gif);
    background-attachment: fixed;
  }
```
- **background**: background-color background-image backgroundrepeat\* background-attachment\* backgroundposition\*
  - VD:
 

```
body {
    background: black url(tile.gif) no-repeat top left;
  }
```

Phát triển ứng dụng Web 48

48

---

---

---

---

---

---

---

---



## Border

- Được phân chia với thành phần khác bởi margin
- Các thuộc tính:
  - **border:** border-width border-style border-color
    - VD: p { border: 1px dashed #000; }
  - **border-style:** none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset
  - **border-color:** [colour-rgb()], [colour-hex], [colour-name]
  - **border-[top, right, bottom, left]:** border-width border-style border-color
    - VD: h1 {border-bottom: 1px double green; }

49

---

---

---

---

---

---

---

---

## Border

- **border-[top, right, bottom, left]-width:** thin, medium, thick, [length]
- **border-[top, right, bottom, left]-style:** none, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset
- **border-[top, right, bottom, left]-color:** [colour-rgb()], [colour-hex], [colour-name]

50

---

---

---

---

---

---

---

---

## Margin

- Cho phép chia các thành phần riêng biệt
- Các thuộc tính:
  - **margin:** margin-top margin-right margin-bottom margin-left.  
Thiết lập margin cho mỗi thành phần, theo chiều kim đồng hồ bắt đầu từ vị trí top
    - VD: p {margin: 4px 10px 4px 10px; }
  - **margin-[top, right, bottom, left]:** auto, [length], [%]
    - VD: li { margin-top: 4em; }
  - **Margin với 1 giá trị:** p {margin: 4px } Tất cả các cạnh của margin sẽ có cùng giá trị (4px)
  - **Margin với 2 giá trị:** p { margin: 10em auto; }
    - Giá trị đầu tiên (10em): cho cạnh trên (top) và cạnh dưới (bottom)
    - Giá trị thứ 2 (auto): cho cạnh trái (left) và cạnh phải (right)

51

---

---

---

---

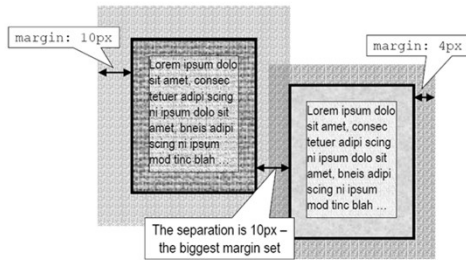
---

---

---

---

## Margin (tt)



52

---

---

---

---

---

---

---

---

## Padding

- Là khoảng cách giữa nội dung (content) và border
- Các thuộc tính:
  - **padding**: padding-top padding-right padding-bottom padding-left
    - VD: p {padding: 4px 10px 4px 10px; }
  - **Padding với 1 giá trị**: p {padding: 4px } Tất cả các cạnh của padding sẽ có cùng giá trị (4px)
  - **Padding với 2 giá trị**: p {padding: 6px 4px; }
    - Giá trị đầu tiên (6px): cho cạnh trên (top) và cạnh dưới (bottom)
    - Giá trị thứ 2 (4px): cho cạnh trái (left) và cạnh phải (right)
  - **padding-[top, right, bottom, left]: [length], [%]**
    - VD: li {padding : 4em; }

53

---

---

---

---

---

---

---

---

## Positioning

- Dùng để xác định vị trí của các phần tử
- Vị trí các phần tử được xác định bởi các thuộc tính top, right, bottom, left. Tuy nhiên các thuộc tính này sẽ không hoạt động trừ khi thuộc tính position được thiết lập
- Cú pháp:
  - **position**: static, relative, absolute, fixed
  - **static**: các phần tử được thiết lập mặc định là static và luôn hiển thị bình thường theo thứ tự của trang web. Khi thiết lập static các phần tử không chịu tác động của các thuộc tính top, right, bottom và left
  - **relative**: có vị trí tương đối so với vị trí thông thường hiện tại, các phần tử khác sẽ không được đặt bên trái của phần tử này

54

---

---

---

---

---

---

---

---

## Positioning

- **fixed**: các phần tử được thiết lập cố định tại vị trí xác định, các thuộc tính `top`, `right`, `bottom` và `left` được sử dụng cho thuộc tính này.
- **absolute**: vị trí phần tử con được xác định dựa trên phần tử cha của nó.
- **top or bottom**: `auto`, `[%]`, `[length]`
- **left or right**: `auto`, `[%]`, `[length]`
- **overflow**: `visible`, `hidden`, `scroll`, `auto`: nếu nội dung không vừa với phần tử đang chứa thì có thể thêm thuộc tính `overflow` để hiện thành cuộn hoặc không
- **clip**: `[shape]`, `auto`: dùng để cắt các phần tử theo kích cỡ

55

---

---

---

---

---

---

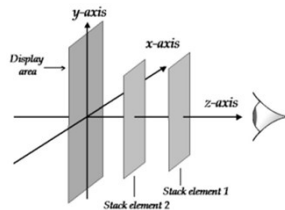
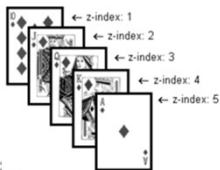
---

---

## Positioning

- **z-index**: `auto`, `[number]`: thiết lập thứ tự trước sau của các phần tử

```
#menu {
  position: absolute;
  z-index: 10;
}
```



56

---

---

---

---

---

---

---

---

## Font và Text

57

---

---

---

---

---

---

---

---

## Font và Text

- **generic** và **specific** font

- Thuộc tính

- font-family
- font-size, font-style, font-height, font-variant, font-stretch, line-height
- text-align, text-decoration, text-indent, text-shadow, text-transform
- white-space, word-spacing, letter-spacing, direction, unicode-bidi

58

---

---

---

---

---

---

---

---

## Font

- **specific** font là các dạng font như "Times New Roman", "Arial"

- **generic font** là các dạng font như là "serif", "san-serif", "monospace", "cursive" hoặc "fantasy".

	Font Samples		
serif	defg	defg	defg
sans-serif	defg	defg	defg
monospace	defg	defg	defg
cursive	defg	defg	defg
fantasy	defg	defg	DEFG

59

---

---

---

---

---

---

---

---

## Font family

- Thuộc tính được dùng để xác định font cho nội dung

```
p {
    font-family: Verdana;
}
```

- Có thể dùng nhiều loại font trong thuộc tính font-family để thay thế khi loại font đấy không có trong máy

```
p {
    font-family: Verdana, Arial, san-serif;
}
```

60

---

---

---

---

---

---

---

---

## Thuộc tính của Font

- o **font-size**: xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, [length], [%]
- o **font-style**: normal, italic, oblique
- o **font-weight**: normal, bold, bolder, lighter, [100, 200, ..., 900]
- o **font-variant**: normal, small-caps
- o **font-stretch**: normal, wider, narrower, ultra-condensed, extra-condensed, condensed, semi-condensed, semi-expanded, expanded, extra-expanded, ultra-expanded
- o **line-height**: normal, [number], [length], [%]



61

---

---

---

---

---

---

---

---

## Thuộc tính của Font

- Có thể viết cùng lúc nhiều thuộc tính của font trong một dòng
 

```
font: [style] [variant] [weight] size [/line-height] family
```

  - Những thuộc tính trong [] có thể có hoặc không
  - Thuộc tính **size** và **family** bắt buộc phải có
  - Ba thuộc tính đầu có thể đảo thứ tự cho nhau
  - Nếu sử dụng thuộc tính **line-height** thì phải dùng ngay sau thuộc tính **size**
- Ví dụ:
 

```
p {
    font: italic normal bold 10pt/2em Helvetica, sans-serif;
}
```



62

---

---

---

---

---

---

---

---

## Thuộc tính của Text

- o **text-align**: left, right, center, justify
- o **text-decoration**: none, underline, overline, line-through, blink
- o **text-indent**: [length], [%]
- o **text-shadow**: none, [color], [length]
- o **text-transform**: none, capitalize, uppercase, lowercase
- o **vertical-align**: baseline, sub, super, top, text-top, middle, bottom, text-bottom. [length], [%]
- o **white-space**: normal, pre, nowrap
- o **word-spacing**: normal, [length]
- o **letter-spacing**: normal, [length]
- o **direction**: ltr, rtl. left to right, right to left
- o **unicode-bidi**: normal, embed, bidi-override
- o **word-spacing**: value. Khoảng cách giữa các từ
- o **letter-spacing**: value. Khoảng cách giữa các ký tự
- o **line-height**: length. Khoảng cách giữa các dòng của văn bản



63

---

---

---

---

---

---

---

---

## Responsive Design

Học viện quốc gia thành phố Hồ Chí Minh  
 TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 64

64

---

---

---

---

---

---

---

---

## Nguyên tắc cơ bản

- Responsive Web Design (RWD)**
- Các trang web có thể được xem bằng nhiều thiết bị khác nhau: máy tính để bàn, máy tính bảng và điện thoại. Trang web phải được hiển thị đẹp và dễ sử dụng trên bất kể thiết bị nào. Nội dung không bị tràn ra ngoài trên thiết bị có kích thước nhỏ, mà phải **thích ứng với nội dung của nó**.

Desktop

Tablet

Phone

Học viện quốc gia thành phố Hồ Chí Minh  
 TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 65

65

---

---

---

---

---

---

---

---

## Nguyên tắc cơ bản

- Responsive Web Design (RWD)**
- Khi sử dụng CSS và HTML để thay đổi kích thước, ấn, co lại, phóng to hoặc di chuyển 1 hoặc 1 số thành phần nội dung để làm cho bố cục trang web trở nên tương thích ở bất kỳ màn hình nào.

Mobile

Tablet

Desktop

Học viện quốc gia thành phố Hồ Chí Minh  
 TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

Phát triển ứng dụng Web 66

66

---

---

---

---

---

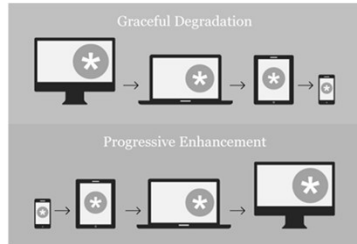
---

---

---

## Nguyên tắc cơ bản

- Nhắc lại 2 chiến lược thiết kế web thường gặp.
- “Graceful Degradation” (sự xuống cấp từ từ) tập trung vào vẻ bề ngoài hơn là nội dung.
- “Progressive Enhancement” chủ yếu tập trung vào nội dung.



Phát triển ứng dụng Web 67

67

## Nguyên tắc cơ bản

- **Một số nguyên tắc cơ bản**
- Đặt mình vào vị trí của end-user
- Luôn luôn là Mobile First: thực hiện layout cho điện thoại di động trước bất kỳ thiết bị nào khác. Đây cũng là nguyên tắc thực hiện responsive của các CSS Library phổ biến như: Bootstrap, Foundation... phải ưu tiên tối ưu hiệu suất và hiển thị cho các thiết bị này trước.
- Hiển thị nội dung kiểu dòng chảy (flow). Nguyên tắc này có nghĩa là nội dung chỉ nên hiển thị trên 1 dòng từ trên xuống dưới, tránh việc để người dùng phải vuốt ngang để có thể xem được nội dung.



Phát triển ứng dụng Web 68

68

## Nguyên tắc cơ bản

- **Một số nguyên tắc cơ bản**
- Sử dụng các breakpoint hợp lý. Chia ra thành nhóm – các thiết bị có kích thước giống nhau để giảm thiểu thời gian và số lượng code CSS. Ví dụ: các thiết bị tablets có độ phân giải chiều rộng tối đa thường là 992px, thì chỉ lấy breakpoints này làm mốc và viết CSS trong đó

```
@media only screen and (min-width: 992px) {
  CSS code here
}
```

Không nên viết min-width : 995px



Phát triển ứng dụng Web 69

69

## Nguyên tắc cơ bản

### • Một số nguyên tắc cơ bản

- Sử dụng các giá trị tương đối thay vì giá trị tuyệt đối. Nên sử dụng các giá trị tương đối khi đặt giá trị width hoặc height cho các phần tử hiển thị trên mobile. Cụ thể là dùng %, hạn chế việc sử dụng các giá trị tuyệt đối như px. Vì chúng không thể tự resize theo chiều rộng/ngang của devices được

```
@media only screen and (<min-width: 992px>) {
  .image {
    width: 100%;
  }
}
```

- Hạn chế khoảng trống, giảm độ lớn font chữ và lược bỏ quảng cáo

70

---

---

---

---

---

---

---

---

---

---

## Nguyên tắc cơ bản

### • Cài đặt khung hình (Viewport)

- Khung hình (Viewport) là khu vực hiển thị của người dùng trên trang web.
- Khung nhìn thay đổi tùy theo thiết bị và sẽ nhỏ hơn trên điện thoại di động. Các trình duyệt trên các thiết bị di động đã tự thu nhỏ toàn bộ trang web cho vừa với màn hình. Nhưng cách này chưa tối ưu..
- HTML5 đã giới thiệu một phương pháp cho phép các nhà thiết kế web kiểm soát khung nhìn thông qua thẻ <meta>.

<meta name="viewport" content="width=device-width, initial-scale=1.0">

71

---

---

---

---

---

---

---

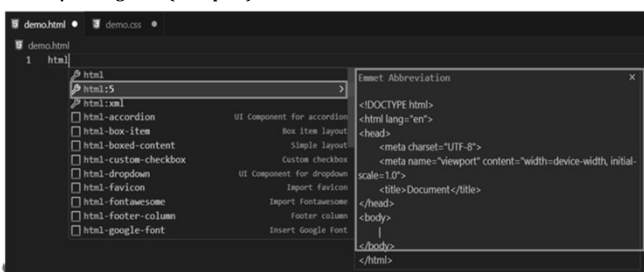
---

---

---

## Nguyên tắc cơ bản

### • Cài đặt khung hình (Viewport)



72

---

---

---

---

---

---

---

---

---

---



## Nguyên tắc cơ bản

### • Cài đặt khung hình (Viewport)

- Định dạng này cung cấp cho trình duyệt cách kiểm soát kích thước và tỷ lệ của trang.
- ❑ Phần width = device-width đặt chiều rộng của trang theo chiều rộng màn hình của thiết bị (sẽ thay đổi tùy theo thiết bị).
- ❑ Phần này initial-scale = 1.0 đặt mức thu phóng ban đầu khi trang được trình duyệt tải lần đầu tiên.

73

---

---

---

---

---

---

---

---

## Layout

- Bố cục trang web (Website layout hay Layout website) là sự sắp xếp của tất cả các phần tử trực quan trên trang web và các mối quan hệ giữa chúng.
- Ví dụ: bố cục chia đôi màn hình



74

---

---

---

---

---

---

---

---

## Layout

Ví dụ: bố cục bất đối xứng



75

---

---

---

---

---



---

---

---

## Layout

Ví dụ: bố cục 1 cột

Phát triển ứng dụng Web 76

76

---

---

---

---

---



---

---

---

## Layout

Ví dụ: bố cục dải ngang

Phát triển ứng dụng Web 77

77

---

---

---

---

---

---

---

---

## Xây dựng layout

- **Xây dựng với Table.**
- Cách đơn giản nhất để tạo ra các layout là sử dụng thẻ <table> trong HTML. Nội dung được sắp xếp vào các cột và hàng. Vì thế ta có thể lợi dụng những hàng và cột này mà không cần sử dụng quá nhiều CSS.
- Trang web có thể được thiết kế thành nhiều cột với các phần nội dung khác nhau. Có thể giữ nội dung chính trong cột giữa và cột trái làm cột chứa menu, và cột phải dùng để đặt các quảng cáo. Loại layout này được mô tả như hình sau:

Phát triển ứng dụng Web 78

78

---

---

---

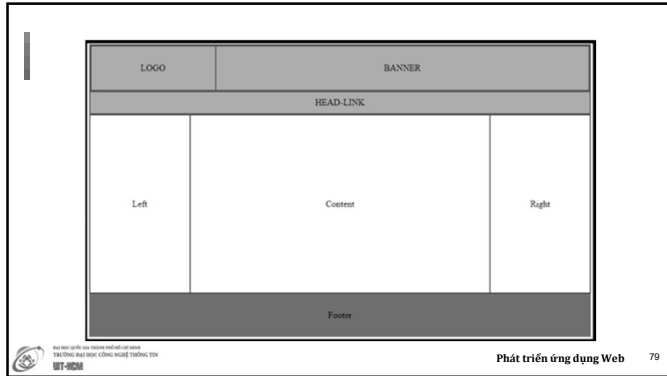
---

---

---

---

---



79

---

---

---

---

---

---

---

---

### Xây dựng layout

- **Xây dựng với Table. Ví dụ 1 phần của table**

```
<table border="0" cellpadding="0" cellspacing="0" width="100%">
<tr>
<td>
<table align="center" border="0" cellpadding="0" cellspacing="0" width="900" style="border-collapse: collapse;">
<tr>
<td bgcolor="#70bbd9">
<table border="1" cellpadding="0" cellspacing="0" width="100%">
<tr>
<td width="25%" valign="top" align="center" style="padding: 30px;">
LOGO
</td>
<td width="75%" valign="top" align="center" style="padding: 30px;">
BANNER
</td>
</tr>
</table>
</td>
</tr>
</table>
```

Phát triển ứng dụng Web 80

80

---

---

---

---

---

---

---

---

### Xây dựng layout

- **Xây dựng với Table.**
- Tuy nhiên <table> lại bộc lộ khá nhiều nhược điểm khi sử dụng làm layout cho một trang web có cấu trúc như trên vì load chậm, khó tùy chỉnh và khó kết hợp với CSS - Javascript để tạo lên sự linh hoạt.

Phát triển ứng dụng Web 81

81

---

---

---

---

---

---

---

---

## Xây dựng layout

### • Sử dụng Flex - box

- Flexbox là một kiểu dàn trang (layout mode) tự cân đối kích thước của các phần tử bên trong để hiển thị trên mọi thiết bị. Ta không cần thiết lập kích thước của phần tử, không cần sử dụng float, chỉ cần thiết lập nó hiển thị chiều ngang hay chiều dọc, lúc đó các phần tử bên trong có thể tự hiển thị theo ý muốn.
- Thành phần quan trọng nhất của Flexbox là:
  - container: là thành phần lớn bao quanh các phần tử bên trong, các item bên trong sẽ hiển thị dựa trên thiết lập của container này.
  - item: là phần tử con của container, ta có thể thiết lập nó sẽ sử dụng bao nhiêu cột trong một container, hoặc thiết lập thứ tự hiển thị của nó.

82

---

---

---

---

---

---

---

---

---

---

## Xây dựng layout

### • Sử dụng Flex - box

- Một số thuộc tính cơ bản:
  - Dùng display: flex; để tạo ra một flex container.
  - Dùng justify-content để căn ngang các items.
  - Dùng align-items để căn dọc các items.
  - Dùng flex-direction nếu muốn các items theo hướng chiều dọc chứ không phải ngang.
- Dùng row-reverse hoặc column-reverse để đảo ngược thứ tự mặc định.
- Dùng order để tùy chỉnh thứ tự một item cụ thể.
- Dùng align-self để căn dọc một item cụ thể.
- Dùng flex để tạo ra một flexible boxes có thể stretch và shrink

83

---

---

---

---

---

---

---

---

---

---

## Xây dựng layout



84

---

---

---

---

---

---

---


---

---

---

Xây dựng layout

```
<!DOCTYPE html>
<html>
<head>
<link rel="stylesheet" href="demo.css">
</head>
<body>
<div class="wrapper">
  <header class="header-flex-2">Header</header>
  <article class="main">
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT-HCM

Phát triển ứng dụng Web

85

85

---

---

---

---

---

---


---

---

Xây dựng layout

<p>Mô-đun Flexbox Layout (Flexible Box)  
(Theo khuyến nghị của W3C kể từ tháng 10 năm 2017)  
nhằm mục đích cung cấp một cách bố trí, sắp xếp  
và phân phối không gian hiệu quả hơn các item trong một container,  
ngay cả khi kích thước của chúng  
không xác định hoặc động (Do đó có từ 'flex').</p>

```
</article>
<aside class="aside aside-1">Aside 1</aside>
<aside class="aside aside-2">Aside 2</aside>
<footer class="footer-flex-2">Footer</footer>
</div>
</body>
</html>
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT-HCM

Phát triển ứng dụng Web

86

86

---

---

---

---

---

---

---


---

Xây dựng layout

```
.wrapper {
  display: flex;
  /* kích hoạt flex box */
  flex-flow: row wrap;
  font-weight: bold;
  text-align: center;
}
.wrapper> {
  padding: 10px;
  flex: 1 100%;
  /* cho tất cả phần tử bên trong có độ dài
  100% và tỉ lệ chiếm không gian trống là như
  nhau*/
}

.header-flex-2 {
  background: tomato;
}

.footer-flex-2 {
  background: lightgreen;
  order: 4;
}
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT-HCM

Phát triển ứng dụng Web

87

87

---

---

---

---

---

---

---

---

### Xây dựng layout


Hiện nay, theo lời khuyên từ Mozilla thì chúng ta sử dụng Flexbox để thiết lập bố cục trong phạm vi nhỏ (ví dụ như những khung trong website) và khi thiết lập bố cục ở phạm vi lớn hơn (như chia cột website) thì vẫn nên sử dụng kiểu thông thường là dàn trang theo dạng lưới (grid layout).

```
.main {
  text-align: left;
  background: deepskyblue;
  height: 400px;
  flex: 3 0px;
  /* cho phần nội dung main ở giữa chiếm 3
  phần không gian trống so với 2 phần aside
  bên cạnh */
  order: 2;
}

.aside {
  flex: 1 0 0;
}
```

```
/* 2 phần aside sẽ chỉ chiếm 1 phần
không gian */
.aside-1 {
  background: gold;
  height: 400px;
  order: 1;
}

.aside-2 {
  background: hotpink;
  height: 400px;
  order: 3;
}
```



TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT-NGHIÊN

Phát triển ứng dụng Web 88

88

---

---

---

---

---


---


---

---

### Xây dựng layout

Hiện nay, theo lời khuyên từ Mozilla thì chúng ta sử dụng Flexbox để thiết lập bố cục trong phạm vi nhỏ (ví dụ như những khung trong website) và khi thiết lập bố cục ở phạm vi lớn hơn (như chia cột website) thì vẫn nên sử dụng kiểu thông thường là dàn trang theo dạng lưới (grid layout).





TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT-NGHIÊN

Phát triển ứng dụng Web 89

89

---

---

---

---

---


---


---

---

### Xây dựng layout

- **Sử dụng Grid:**
- Grid là một module tạo bố cục website trong CSS bằng cách hỗ trợ hệ thống bố cục theo dạng lưới 2 chiều, gồm hàng và cột.
- Grid ra đời nhằm đơn giản hóa việc xây dựng giao diện website và hoạt động rất tốt với Flexbox. Flexbox cũng là 1 module hỗ trợ xây dựng bố cục nhưng áp dụng với các bố cục một chiều đơn giản. Khi Grid và Flexbox kết hợp với nhau, ta có thể tạo ra nhiều bố cục website phức tạp và đa dạng hơn.
- Grid cho phép ta tạo một ma trận bố cục 2 chiều gồm các dòng và các cột. Ở mỗi dòng, cột và mỗi phần tử trong Grid có thể chỉnh sửa style. Vì vậy Grid cũng rất thích hợp để tạo bố cục trang Web.





TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UT-NGHIÊN

Phát triển ứng dụng Web 90

90

---

---

---

---

---

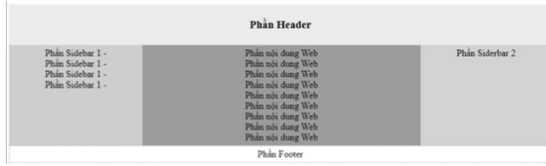
---

---

---

## Xây dựng layout

### • Sử dụng Grid:



91

---

---

---

---

---

---

---

---

---

---

## Xây dựng layout

### • Sử dụng Grid: Bốn bước cơ bản để tạo layout bằng grid:

- ❑ Tạo một thành phần container, và định nghĩa nó là display: grid;.
- ❑ Sử dụng container đó để định nghĩa các grid track sử dụng các thuộc tính grid-template-columns và grid-template-rows.
- ❑ Đặt các thành phần con bên trong container.
- ❑ Thiết lập nơi mà mỗi phần tử con thuộc về trong grid bằng cách định nghĩa grid-column và grid-row của nó.

92

---

---

---

---

---

---

---

---

---

---

## Xây dựng layout

### • Sử dụng Grid:

```
<div class="grid-container">
  <div class="grid-item header">
    <h3>Phần Header</h3>
  </div>
  <div class="grid-item sidebar-1">
    Phần Sidebar 1<br>
    Phần Sidebar 1<br>
    Phần Sidebar 1<br>
    Phần Sidebar 1<br>
  </div>
```

```
<div class="grid-item content">
  Phần nội dung Web<br>
  Phần nội dung Web<br>
  Phần nội dung Web<br>
  Phần nội dung Web<br>
  Phần nội dung Web<br>
  Phần nội dung Web<br>
  Phần nội dung Web<br>
  Phần nội dung Web<br>
  Phần nội dung Web<br>
</div>
<div class="grid-item sidebar-2">Phần Sidebar
2</div>
<div class="grid-item footer">Phần Footer</div>
</div>
```

93

---

---

---

---

---

---

---

---

---

---

## Xây dựng layout

- Sử dụng Grid:**

```
'header header header header',
'sidebar-1 content content content sidebar-2',
'footer footer footer footer';
text-align: center;
}
```

```
.header {
padding: 5px;
grid-area: header;
width: 100%;
height: 70px;
background-color: yellow;
box-sizing: border-box;
}

.sidebar-1 {
padding: 5px;
grid-area: sidebar-1;
background-color: pink;
}

.sidebar-2 {
padding: 5px;
grid-area: sidebar-2;
background-color: lightgreen;
}

.content {
padding: 5px;
grid-area: content;
background-color: #BB8FCE;
}

.footer {
padding: 5px;
grid-area: footer;
border: 1px dashed #AAA;
}
```

Phát triển ứng dụng Web 94

94

---

---

---

---

---

---

---

---

## Xây dựng layout

- Sử dụng Framework (Bootstrap):**
- Bootstrap là một framework HTML, CSS, và JavaScript cho phép thiết kế phát triển responsive web mobile.
- Nó cho phép thiết kế website responsive nhanh hơn và dễ dàng hơn. Bootstrap bao gồm các HTML templates, CSS templates và Javascript tạo ra những cái cơ bản có sẵn như: typography, forms, buttons, tables, navigation, modals, image carousels và nhiều thứ khác.
- Trong bootstrap có thêm các plugin Javascript giúp cho việc thiết kế responsive dễ dàng và nhanh chóng hơn.

Phát triển ứng dụng Web 95

95

---

---

---

---

---


---

---

---

## Các vị trí xây dựng Responsive thông dụng.

- Responsive menu:**
  - Vị trí menu điều hướng các hoạt động của website. Thông thường với vị trí này chúng ta phải tạo responsive cho nó. Ở giao diện màn hình lớn thì menu hiển thị đầy đủ các mục chính (cấp 1) dàn ngang nhưng qua giao diện nhỏ thì chúng được ẩn đi chỉ hiển thị một nút nhỏ (menu hamburger). Khi người dùng click vào nút đó thì hiển thị menu ra theo chiều dọc



Phát triển ứng dụng Web 96

96

---

---

---

---

---

---

---

---



## Các vị trí xây dựng Responsive thông dụng.

### • Responsive Column:

- Mỗi giao diện thông thường chúng ta có các vị trí sidebar left, sidebar right và content. Như vậy với ba vị trí này thì chúng ta tạm chia làm ba column. Nếu ở giao diện lớn thì chúng ta sẽ hiển thị nó ở dạng 3 column nhưng ở giao diện nhỏ thì chúng ta chỉ hiển thị nó ở dạng 1 column.

### • Responsive font size

- Với font size thì chúng ta hay thay đổi kích thước cho nó, với giao diện lớn thì chúng ta hiển thị kích thước lớn nhưng qua giao diện nhỏ thì đôi lúc chúng ta cần cho kích thước nhỏ lại để nội dung hiển thị trên một hàng hoặc hiển thị nhỏ lại để dễ nhìn hơn.



Phát triển ứng dụng Web 97

97

---

---

---

---

---

---

---

---

## Các vị trí xây dựng Responsive thông dụng.

### • Responsive image:

- Với hình ảnh thì nếu ta thiết lập chiều rộng và chiều cao cho nó thì khi qua giao diện nhỏ sẽ bị vỡ ngay vì kích thước của hình ảnh lớn hơn kích thước của thiết bị. Lúc này ta phải thay đổi lại kích thước sao cho hiển thị đúng với chiều rộng của thiết bị.

- Ví dụ:** người lập trình muốn đoạn code đã chia màn hình thành 2 loại kích thước khác nhau:

- Loại nhỏ: Kích thước bé hơn hoặc bằng 768px

- Loại lớn: Kích thước lớn hơn 768px

- Vậy khi thay đổi kích thước của trình duyệt nếu đang nằm trong khoảng nào thì CSS ở khoảng đó sẽ có tác dụng.



Phát triển ứng dụng Web 98

98

---

---

---

---

---

---

---

---

## Các vị trí xây dựng Responsive thông dụng.

```
/* // Trình duyệt nhỏ có width là bé hơn hoặc bằng 768px */
@media only screen and (max-width: 768px) {
  #sidebar {
    width: 100%
  }
}
```

```
/* // Trình duyệt nhỏ có width là lớn hơn 768px */
@media only screen and (min-width: 769px) {
  #sidebar {
    width: 300px
  }
}
```



Phát triển ứng dụng Web 99

99

---

---

---

---

---


---

---

---

Bài tập

- Sử dụng lại phần HTML ở BT1
- Xem BT2 trên courses



Đại học Quốc gia Thành phố Hồ Chí Minh

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UTM-HCM

Phát triển ứng dụng Web100

100

---

---

---

---

---

---

---

Q & A



Cảm ơn đã theo dõi

Hy vọng cùng nhau đi đến thành công.



Đại học Quốc gia Thành phố Hồ Chí Minh

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN

UTM-HCM

Phát triển ứng dụng Web101

101

---

---

---

---

---

---

---