

# THIẾT KẾ WEB



# CSS

ThS. Dương Hữu Thành  
Khoa CNTT, Đại học Mở Tp.HCM  
[thanh.dh@ou.edu.vn](mailto:thanh.dh@ou.edu.vn)

HTML



JS



CSS

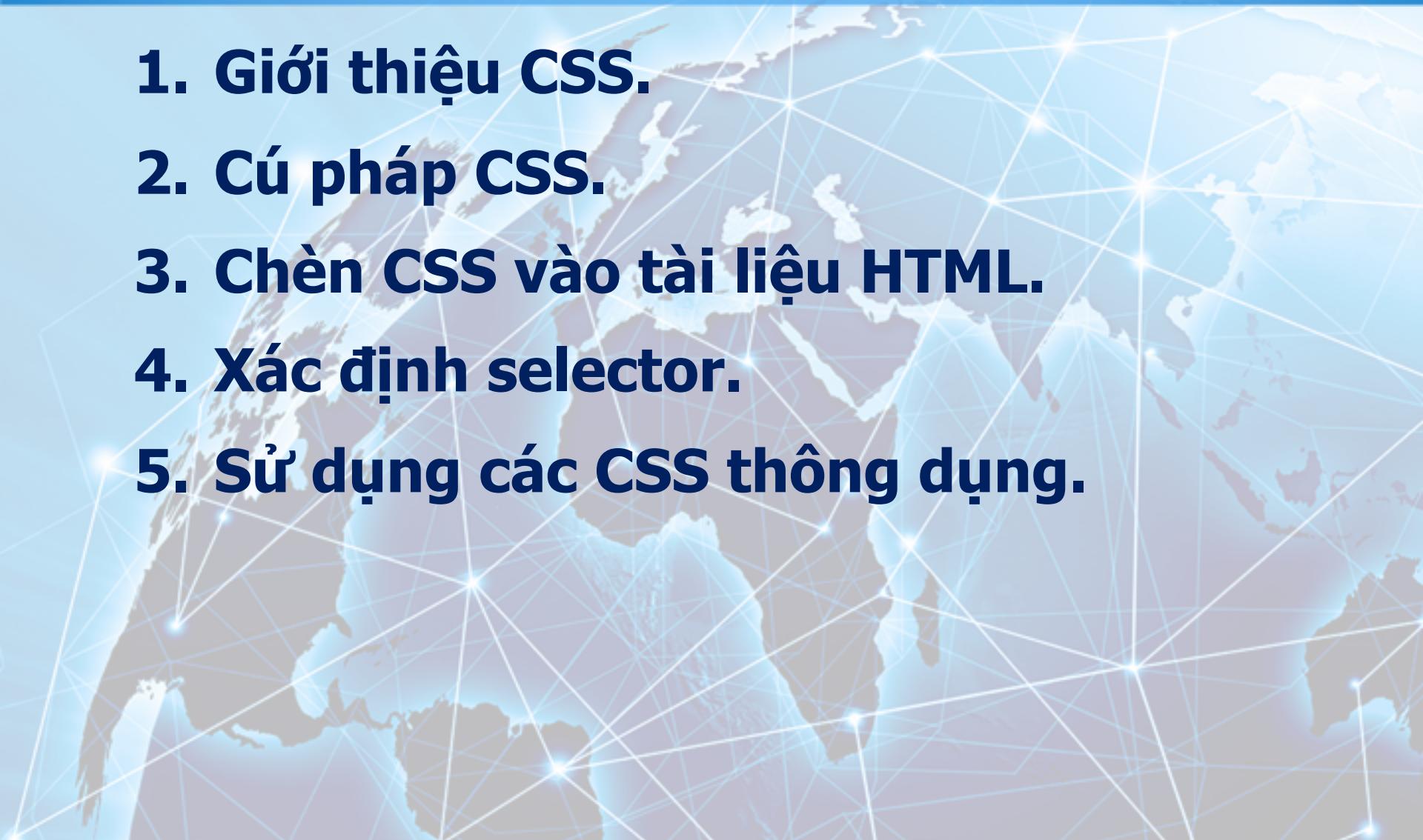




# Nội dung chính



- 1. Giới thiệu CSS.**
- 2. Cú pháp CSS.**
- 3. Chèn CSS vào tài liệu HTML.**
- 4. Xác định selector.**
- 5. Sử dụng các CSS thông dụng.**





# Giới thiệu CSS



Nguồn: internet

**HTML + CSS**

**Chỉ có HTML**



# Giới thiệu CSS



- CSS viết tắt của **Cascading Style Sheet**.
- CSS là ngôn ngữ quy định **cách trình bày** (style) của các tài liệu viết bằng HTML, XHTML, XML, SVG, UML, v.v.
- CSS mô tả cách thức (**HOW**) một trang web hiển thị đến người dùng.
- Mục đích của CSS là thiết kế, trình bày trang web **dễ dàng** và đảm bảo tính **nhất quán** trong trình bày website.



# Giới thiệu CSS



- CSS1 được giới thiệu năm 1996 – mô hình định dạng đơn giản, trực quan cho các thẻ HTML.
- CSS2 được giới thiệu 05/1998.
- CSS3 được giới thiệu 1999 , nó phân chia tài liệu thành các module và mỗi module có những đặc trưng mở rộng mới trong CSS2.



# Cú pháp CSS

- Cú pháp để thiết lập CSS cho một thành phần HTML gồm 2 phần sau: **Selector** và **Declaration**

Có thể có nhiều declaration thiết lập cho các selector, cách nhau bằng dấu chấm phẩy (;

Declaration

Tập tin CSS  
sẽ có phần  
mở rộng là  
**\*.css**

```
Selector { Property: Value; }
```

Chỉ định thẻ HTML được áp dụng CSS.  
Có thể khai báo nhiều selector cùng lúc  
cách nhau bằng dấu phẩy (,)

Một thuộc tính  
của thẻ HTML.

Các giá trị gán  
cho thuộc tính



# Cú pháp CSS - Ví dụ

- Thiết lập thuộc tính biên (border) cho thành phần table

```
table {  
    border: 1px solid #008B8B;  
}
```

- Thiết lập font và màu văn bản cho các thành phần h1 và h2

```
h1, h2 {  
    font-family: Tahoma;  
    color: blue;  
}
```



# Ghi chú trong CSS



- Ghi chú sẽ được trình duyệt bỏ qua.
- Để ghi chú trong CSS sử dụng cặp ký hiệu **/\*** và **\*/**.

```
/* Thiết lập CSS cho các thành phần h1, h2 */
h1, h2 {
    font-family: Tahoma; /* Font là Tahoma */
    color: blue; /* Màu văn bản là xanh */
}
```



# Chèn CSS vào tài liệu HTML



- Có 3 cách để chèn CSS vào các thành phần trong tài liệu HTML:
  - Dùng thuộc tính **style** trên từng thành phần (**inline**).
  - Dùng thành phần **<style>** bên trong **<head>** của tài liệu HTML (**internal**).
  - Nạp tập tin **\*.css** từ bên ngoài vào tài liệu HTML dùng thẻ **<link>** bên trong **<head>** (**external**).



# Ví dụ

```
<b style="color:blue;font-family:Tahoma;">OU</b>
```

```
<head>
  <style type="text/css">
    table {
      border: 1px solid #008B8B;
      color: #FFFFFF;
    }
  </style>
</head>
```

```
<head>
  <link rel="stylesheet" type="text/css"
        href="css/main.css" />
</head>
```

inline < -----

internal ----->

external < -----



# Thứ tự ưu tiên của CSS



- Thứ tự ưu tiên áp dụng CSS sẽ **tăng dần** theo trật tự **external < internal < inline**, và **thấp nhất** là các CSS mặc định của trình duyệt (default browser).
- Để thay đổi độ ưu tiên của CSS sử dụng thuộc tính **!important** ngay sau phần giá trị của một thuộc tính.



# Ví dụ

```
<!DOCTYPE html>
<html>
<head>
    <style>
        p {
            font-weight: bold;
            font-size: 30px;
            color: blue !important;
        }
    </style>
</head>
<body>
    <p style="color: red;">CSS</p>
</body>
</html>
```

CSS



# Cascading Style Sheet

**Xác định Selector**



# Xác định selector



- Để xác định một selector có thể sử dụng một trong các yếu tố sau:
  - Sử dụng **tên** thành phần: div, p, table, tr, td, ...
  - Sử dụng **thuộc tính id** của các thành phần.
  - Sử dụng **thuộc tính class** của các thành phần.
  - Sử dụng **bất kỳ thuộc tính** nào khác của các thành phần.



# Sử dụng tên thành phần



```
p {  
    font-size: 30px;  
    font-weight: bold;  
    background-color: #D9EDF7;  
}
```

Thiết lập CSS tất cả các thành phần **p** trong một trang.

```
<p>CSS1</p>  
<div>CSS2</div>  
<p>CSS3</p>
```

**CSS1**

CSS2

**CSS3**

# Sử dụng tên thành phần

```
div span {  
    font-weight: bold;  
    font-size: 30px;  
    color: RED;  
}
```

Thiết lập CSS tất cả các thành phần **span** là **con** (descendant) thành phần **div**.

```
<div>  
    <p>CSS1</p>  
    <p>  
        <span>CSS2</span>  
    </p>  
    <span>CSS3</span>  
</div>
```

CSS1

**CSS2**

**CSS3**

# Sử dụng tên thành phần



```
div > span {  
    font-weight: bold;  
    font-size: 30px;  
    color: RED;  
}
```

Thiết lập CSS tất cả các thành phần **p** là con **trực tiếp** (child) thành phần **div**.

```
<div>  
    <p>CSS1</p>  
    <p><span>CSS2</span></p>  
    <span>CSS3</span>  
</div>
```

CSS1

CSS2

**CSS3**

# Sử dụng tên thành phần



```
div + p {  
    font-weight: bold;  
    font-size: 30px;  
    color: red;  
}
```

Thiết lập CSS tất cả các thành phần **p** là thành phần anh em **ngay sau** (adjacent sibling) thành phần **div**.

```
<p>CSS</p>  
<div>CSS1</div>  
<p>CSS2</p>  
<p>CSS3</p>
```

CSS  
CSS1  
**CSS2**  
CSS3

# Sử dụng tên thành phần

```
div ~ p {  
    font-weight: bold;  
    font-size: 30px;  
    color: RED;  
}
```

Thiết lập CSS tất cả các thành phần **p** là thành phần anh em ( sibling) thành phần **div**.

```
<p>CSS</p>  
<div>CSS1</div>  
<p>CSS2</p>  
<p>CSS3</p>
```

CSS  
CSS1  
**CSS2**  
**CSS3**



# Sử dụng thuộc tính id

```
#main {  
    text-align: center;  
    font-weight: bold;  
    font-size: 25px;  
    color: blue;  
}
```

Thiết lập CSS cho thành phần có **id** là **main**.

```
<p id="first">CSS1</p>  
<div>CSS2</div>  
<p id="main">CSS3</p>
```

CSS1

CSS2

**CSS3**

# Sử dụng thuộc tính id



```
p#main {  
    text-align: center;  
    font-weight: bold;  
    font-size: 25px;  
    color: blue;  
}
```

Thiết lập CSS cho thành phần **p** có **id** là **main**.

```
#main p {  
    text-align: center;  
    font-weight: bold;  
    font-size: 25px;  
    color: blue;  
}
```

Thiết lập CSS cho thành phần **p** nằm bên trong thành phần có **id** là **main**.

# Sử dụng thuộc tính class

```
.message {  
    font-size: 20px;  
    font-weight: bold;  
    background-color: #F5F5F5;  
}
```

Thiết lập CSS tất cả các thành phần có thuộc tính **class** là **message**.

```
<p class="message">GOOD</p>  
<div class="message info">  
    <p>INFO - LOOK GREAT!</p>  
</div>  
<div class="message">  
    <p class="success">SUCCESS</p>  
</div>
```

**GOOD**

**INFO - LOOK GREAT!**

**SUCCESS**

# Sử dụng thuộc tính class

```
p.message {  
    font-size: 36px;  
}
```

Thiết lập CSS tất cả các thành phần **p** có thuộc tính **class** là **message**.

```
<p class="message">GOOD</p>  
<div class="message info">  
    <p>INFO - LOOK GREAT!</p>  
</div>  
<div class="message">  
    <p class="success">SUCCESS</p>  
</div>
```

GOOD

INFO - LOOK GREAT!

SUCCESS

# Sử dụng thuộc tính class



```
.message.info {  
    background-color: #D9EDF7;  
}
```

Thiết lập CSS tất cả các thành phần có thuộc tính **class** chứa hai giá trị là **message** và **info**.

```
<p class="message">GOOD</p>  
<div class="message info">  
    <p>INFO - LOOK GREAT!</p>  
</div>  
<div class="message">  
    <p class="success">SUCCESS</p>  
</div>
```

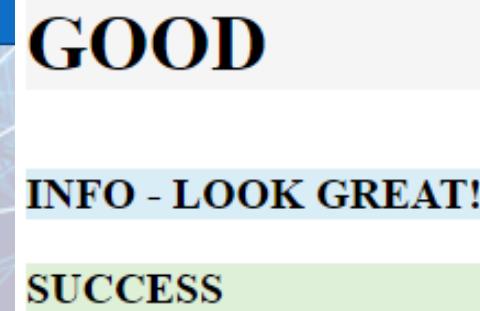
GOOD  
INFO - LOOK GREAT!  
SUCCESS

# Sử dụng thuộc tính class

```
.message .success {  
    background-color: #DFF0D8;  
}
```

Thiết lập CSS tất cả các thành phần có thuộc tính **class** là **success** nằm bên trong thành phần có thuộc tính **class** là **message**.

```
<p class="message">GOOD</p>  
<div class="message info">  
    <p>INFO - LOOK GREAT!</p>  
</div>  
<div class="message">  
    <p class="success">SUCCESS</p>  
</div>
```



GOOD

INFO - LOOK GREAT!

SUCCESS

# Sử dụng các thuộc tính khác

```
[title] {  
    font-size: 30px;  
    font-weight: bold;  
    color: brown;  
}
```

Thiết lập CSS tất cả các thành phần có thuộc tính **title**.

```
p[title] {  
    color: blue;  
}
```

Thiết lập CSS tất cả thành phần **p** có thuộc tính **title**.

```
<p title="first name">Thanh</p>  
<p title="last name">Duong</p>  
<p title="full name">Thanh Duong</p>  
<p>HCMC OU</p>  
<span title="year">1990</span>
```

Thanh

Duong

Thanh Duong

HCMC OU

1990

# Sử dụng các thuộc tính khác

```
p[title="full name"] {  
    font-size: 30px;  
    font-weight: bold;  
    color: brown;  
}
```

Thiết lập CSS tất cả các thành phần p với thuộc tính **title** có giá trị là “full name”.

```
<p title="first name">Thanh</p>  
<p title="last name">Duong</p>  
<p title="full name">Thanh Duong</p>  
<p>HCMC OU</p>
```

Thanh

Duong

**Thanh Duong**

HCMC OU

# Sử dụng các thuộc tính khác



`p[title~="name"] {`  
}

Thiết lập CSS tất cả các thành phần p có thuộc tính **title** với giá trị **chứa từ** "name".

`p[title|= "first"] {`  
}

Thiết lập CSS tất cả các thành phần p có thuộc tính **title** với giá trị có **từ bắt đầu** là "first".

`p[class^="f"] {`  
}

Thiết lập CSS tất cả các thành phần p có thuộc tính **class** với giá trị có **chuỗi bắt đầu** là "f".

`p[class$="me"] {`  
}

Thiết lập CSS tất cả các thành phần p có thuộc tính **class** với giá trị có **chuỗi kết thúc** là "me".

`p[class*="a"] {`  
}

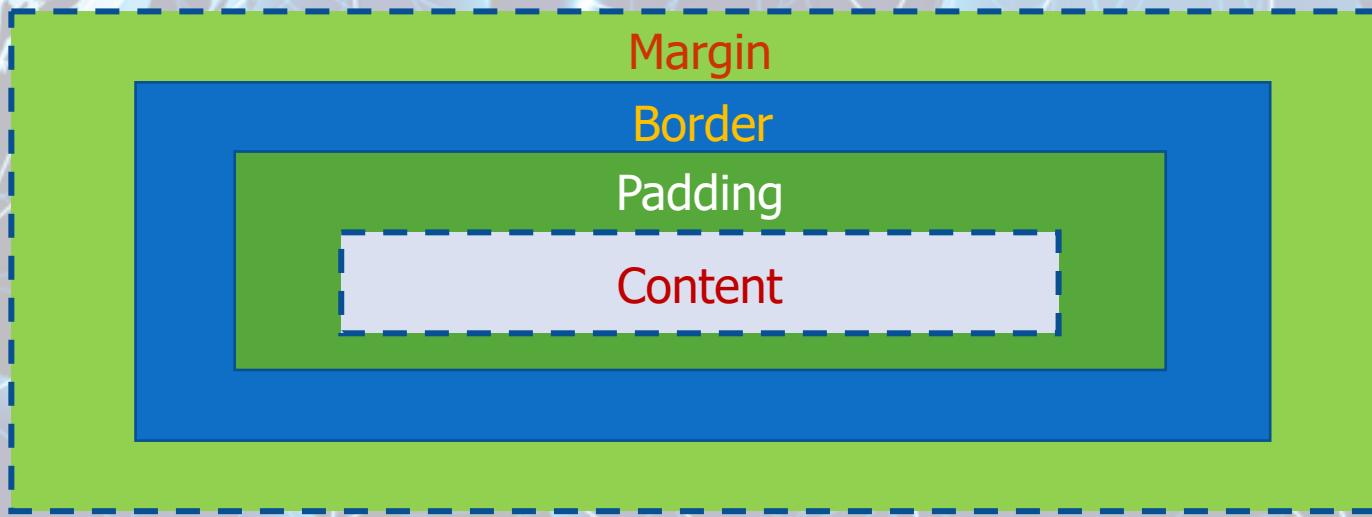
Thiết lập CSS tất cả các thành phần p có thuộc tính **class** với giá trị **chứa chuỗi** "a".



# Box Model



- Các thành phần HTML thường được xem như là các box.
- Trong CSS, thuật ngữ “box model” nói về việc thiết kế và sắp xếp bố cục tài liệu HTML, bao gồm: **margin**, **border**, **padding**, và **nội dung** của thành phần.





# Box Model



- **Content:** nội dung của box, có thể là văn bản hoặc hình ảnh.
- **Padding:** vùng trong suốt (transparent) quanh nội dung, hay nói cách khác là các khoảng cách từ nội dung đến border.
- **Border:** biên xung quanh padding và content.
- **Margin:** vùng trong suốt ngoài biên.

# Cascading Style Sheet

**CSS Background**



# CSS Background



- **background-color**: thiết lập màu nền.
- **background-image**: thiết lập ảnh nền.
- **background-repeat**: thiết lập việc lặp ảnh nền.
- **background-position**: vị trí bắt đầu thiết lập ảnh nền.
- **background-attachment**: điều khiển việc trượt (scroll) ảnh nền.
- **background**: thuộc tính ngắn gọn (shorthand) để thiết lập cho các thuộc tính trên.



# background-color



- background-color nhận 1 trong 3 giá trị sau:
  - **Tên màu:** red, green, blue, gold, brown, ...
  - **Mã hex:** #0000FF (blue), #A52A2A (brown), ...
  - **Giá trị rgb():** rgb(255, 0, 0) (red), ...
- Ví dụ:

```
body {  
    background-color: #008B8B;  
}
```

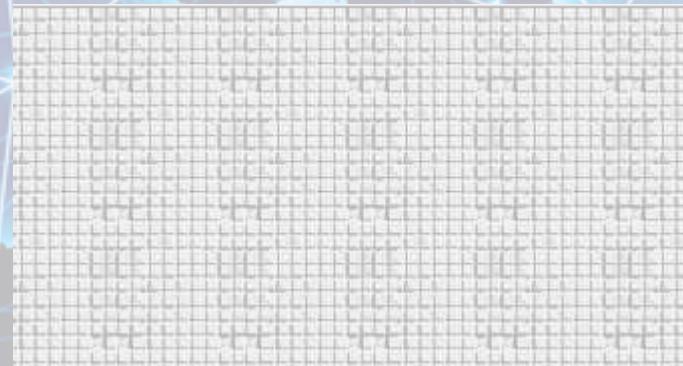


# background-image

- background-image: nhận giá trị dạng  
url("<đường-dẫn-file-ảnh>");

- Ví dụ

```
body {  
    background-image: url ('images/bg.gif');  
}
```



# background-repeat



- Mặc định background-image sẽ hiển thị **lặp** ảnh theo chiều dọc và chiều ngang, muốn chỉnh sửa cách lặp này dùng **background-repeat**.
- background-repeat nhận 1 trong các giá trị sau:
  - **repeat-x**: lặp theo chiều ngang.
  - **repeat-y**: lặp theo chiều dọc.
  - **repeat**: lặp theo chiều ngang và chiều dọc (mặc định).
  - **no-repeat**: không lặp.



# background-repeat

```
body {  
    background-image: url('images/bg.gif');  
    background-repeat: repeat-x;  
}
```



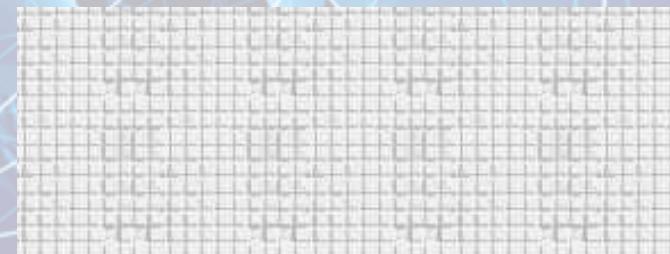
repeat-x



repeat-y



no-repeat



repeat



# background-position



- Xác định **vị trí hiển thị** của background, có thể nhận một trong các giá trị **left, right, center, top, hoặc một tọa độ** màn hình.
- Ví dụ: hiển thị background từ tọa độ (100, 200), tức là vị trí cách bên trái (left) 100px và cách đỉnh (top) 200px.

```
body {  
    background-image: url('images/css3.png');  
    background-repeat: no-repeat;  
    background-position: 100px 200px;  
}
```



# background-attachment

- background-attachment chỉ định ảnh nền sẽ **cố định hoặc trượt** (scroll) theo khi kéo trượt trang web, nhận giá trị:
  - **fixed**: ảnh nền luôn cố định khi trượt trang web.
  - **scroll**: ảnh nền sẽ trượt theo khi trượt trang web.

# background



- background là cách viết ngắn gọn cho tất cả thuộc tính trên với thứ tự thiết lập các giá trị như sau:
  - background-color
  - background-image
  - background-repeat
  - background-attachment
  - background-position
- Chú ý: không có vấn đề nếu một trong các giá trị trên bị thiếu, miễn là những giá trị còn lại nằm theo thứ tự này.



# background

- Ví dụ: 2 CSS sau là tương đương

```
body {  
    background-image: url('images/css3.png');  
    background-repeat: no-repeat;  
    background-position: 100px 200px;  
    background-attachment: fixed;  
}
```

```
body {  
    background: url('images/css3.png') no-repeat  
    fixed 100px 200px;  
}
```

# Cascading Style Sheet

**CSS Font**



# CSS Font



- **font-family**: thiết lập danh sách tên font (face).
- **font-style**: thiết lập kiểu font, nhận giá trị normal, italic, hoặc oblique.
- **font-variant**: thiết lập kiểu chữ nhỏ hơn, nhận giá trị **normal** hoặc **small-caps**.
  - **small-caps**: tất cả các ký tự thường sẽ được chuyển thành ký tự hoa, nhưng kích thước những ký tự hoa này sẽ nhỏ hơn những ký tự hoa gốc trong văn bản.



# CSS Font

- **font-weight**: thiết lập độ đậm (bold) hoặc sáng (light) của font.
- **font-size**: thiết lập kích thước font.
- **font**: thuộc tính ngắn gọn (shorthand) để thiết lập cho các thuộc tính trên.



# CSS Font

```
span {  
    font-family: Tahoma, Arial, Serif;  
    font-size: 30px;  
}  
  
span.word {  
    font-style: italic;  
    font-variant: small-caps;  
}  
  
<div>  
    <span>CSS:</span>  
    <span class="word">Cascading Style Sheet</span>  
</div>
```

CSS: *CASCADING STYLE SHEET*

# font-family

- Trong CSS, có 2 loại giá trị cho font-family:
  - generic family**: một nhóm các font có cách hiển thị giống nhau.
  - font family**: font cụ thể.



Generic Family	Font Family	Mô tả
Serif	Times New Roman Georgia	Kết thúc mỗi ký tự là đường gạch nhỏ.
Sans-serif	Arial Verdana	Không có đường gạch kết thúc mỗi ký tự
Monospace	Courier New Lucida Console	Tất cả các ký tự có chiều rộng bằng nhau.

Nguồn: [https://www.w3schools.com/css/css\\_font.asp](https://www.w3schools.com/css/css_font.asp)



# font-family



- Giá trị **font-family** là các tên font hoặc nhóm font cách nhau bằng dấu phẩy (,).
- Trình duyệt sẽ duyệt theo danh sách font từ **trái sang phải** cho đến khi có một font được hỗ trợ sẽ áp dụng nó vào trang.
- Nếu tên font có nhiều hơn một từ thì đặt trong cặp dấu nháy kép ("").
- Ví dụ "Times New Roman"



# font-size

- Thuộc tính này thiết lập **kích thước font** chữ, có thể nhận các giá trị: xx-small, x-small, small, medium, large, x-large, xx-large, smaller, larger, giá trị pixels (hoặc em), hoặc tỷ lệ phần trăm.
- Nếu không chỉ định font-size, thì kích thước font cho văn bản bình thường là 16px (16px = 1em).

```
<p style="font-size:20px;">CSS</p>
<p style="font-size:xx-large;">CSS</p>
<p style="font-size:x-large;">CSS</p>
```

CSS

CSS

CSS



# font-weight

- Thuộc tính này dùng thiết lập độ **đậm** của font, có thể nhận các giá trị: normal, bold, bolder, lighter, 100, 200, 300, 400, 500, 600, 700, 800, 900.

```
p {  
    font-family: arial;  
    font-size: 30px;  
}
```

**CSS Font Bold**

**CSS Font Bolder**

**CSS Font 800**

```
<p style="font-weight:bold;">CSS Font Bold</p>  
<p style="font-weight:bolder;">CSS Font Bolder</p>  
<p style="font-weight:800;">CSS Font 800</p>
```



# font

- font là cách viết **ngắn gọn** để thiết lập cho tất cả các thuộc tính trên.
- Thứ tự thiết lập các giá trị:
  - font-style
  - font-variant
  - font-weight
  - font-size/line-height
  - font-family



# font

- Ví dụ: 2 CSS sau là tương đương

```
body {  
    font-style: italic;  
    font-weight: bold;  
    font-size: 30px;  
    font-family: arial;  
}
```

```
body {  
    font: italic bold 30px arial;  
}
```

# Cascading Style Sheet

**CSS Text**



# CSS Text



- **color**: thiết lập màu văn bản.
- **direction**: thiết lập hướng văn bản, có giá trị ltr hoặc rtl.
- **text-align**: thiết lập canh lề văn bản trong tài liệu, có giá trị left, right, center hoặc justify.
- **text-transform**: chuyển đổi văn bản thường sang hoa hoặc ngược lại.
- **text-decoration**: có giá trị none, underline, overline, line-through, hoặc blink.



# CSS Text



- **letter-spacing**: thiết lập khoảng cách giữa các ký tự trong một từ.
- **word-spacing**: thiết lập khoảng cách giữa các từ trong văn bản.
- **line-height**: thiết lập khoảng cách giữa các dòng.



# CSS Text



- **text-indent**: thiết lập thụt lề (indent) cho văn bản của đoạn văn.
- **white-space**: thiết lập cách xử lý khoảng trắng trong các thành phần.
- **text-shadow**: thêm bóng mờ (shadow) xung quanh văn bản.



# CSS Text

```
p.content {  
    color: blue;  
    letter-spacing: 3px;  
    word-spacing: 10px;  
    text-indent: 1cm;  
}  
  
p.source {  
    direction: rtl;  
}
```

```
<p class="content">CSS  
describes how HTML elements  
are to be displayed on  
screen, paper, or in other  
media</p>  
<p class="source">(source:  
https://www.w3schools.com/css/css\_intro.asp)</p>
```

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

(source: [https://www.w3schools.com/css/css\\_intro.asp](https://www.w3schools.com/css/css_intro.asp))



# text-transform



- Giá trị có thể của thuộc tính này
  - **none**: không chuyển đổi.
  - **capitalize**: chuyển ký tự đầu mỗi từ thành hoa.
  - **uppercase**: chuyển tất cả các ký tự thành hoa.
  - **lowercase**: chuyển tất cả các ký tự thành thường.



# text-transform



```
<p style="text-transform: capitalize;">Cascading style  
sheet</p>  
<p style="text-transform: uppercase;">CasCading Style  
SHeet</p>  
<p style="text-transform: lowercase;">CasCading Style  
SHeet</p>
```

Cascading Style Sheet

CASCADING STYLE SHEET

cascading style sheet



# text-decoration



```
<p style="text-decoration: underline;">underline</p>
<p style="text-decoration: overline;">overline</p>
<p style="text-decoration: line-through;">line-through</p>
```

underline

overline

~~line-through~~



# white-space



- Thuộc tính này chỉ định cách xử lý các khoảng trắng trong các thành phần.
- Thuộc tính này có thể nhận các giá trị sau:
  - **normal**: văn bản sẽ xuống dòng khi dài hơn chiều rộng cửa sổ chứa nó.
  - **nowrap**: văn bản không xuống dòng khi dài hơn chiều rộng cửa sổ chứa nó.
  - **pre**: văn bản sẽ hiển thị đúng như nó được viết trong HTML.



# text-shadow

- Ví dụ tạo bóng ở vị trí chiều ngang 3px, chiều dọc 2px, và màu bóng mở là gray.

```
p {  
    text-indent: 1cm;  
    font-size: 25px;  
    font-weight: bold;  
    text-shadow: 3px 2px gray;  
}
```

**CSS describes how HTML elements are to be displayed on screen, paper, or in other media**

# Cascading Style Sheet

**CSS Padding**



# CSS Padding



- Thuộc tính padding dùng thiết lập **khoảng cách xung quanh nội dung** của các thành phần.
- Sử dụng các thuộc tính cho padding:
  - padding-top
  - padding-right
  - padding-bottom
  - padding-left
  - padding



# CSS Padding



- Các thuộc tính padding có thể nhận các giá trị:
  - **Số**: đơn vị px, pt, cm,..., mặc định giá trị là 0
  - Số **tỷ lệ phần trăm** chiều rộng so với thành phần chứa nó.
  - **inherit**: padding sẽ kế thừa từ thành phần cha.
- Các thuộc tính padding **không thể nhận giá trị âm**.



# CSS Padding

```
div {  
    border-style: solid;  
    height: 50px;  
    padding-left: 25px;  
    padding-top: 30px;  
}
```

```
<div>CSS</div>
```

A diagram of a rectangular div element with a black border. Inside the div, the word "CSS" is written in blue. Red dashed arrows point from the text "CSS" to the left and top edges of the div, indicating the padding values. The arrow pointing left is labeled "25px" and the arrow pointing top is labeled "30px".

# Thuộc tính padding



- padding là thuộc tính ngắn gọn để thiết lập giá trị padding cho các hướng (top, right, bottom, left).
- Thuộc tính này có thể nhận từ 1 đến 4 giá trị.

## **padding: 1px 2px 3px 4px**

- top-padding: 1px
- right-padding: 2px
- bottom-padding: 3px
- left-padding: 4px

## **padding: 1px 2px**

- top-padding: 1px
- right-padding: 2px
- bottom-padding: 1px
- left-padding: 2px

## **padding: 1px 2px 3px**

- top-padding: 1px
- right-padding: 2px
- bottom-padding: 3px
- left-padding: 2px

## **padding: 1px**

- top-padding: 1px
- right-padding: 1px
- bottom-padding: 1px
- left-padding: 1px

# Cascading Style Sheet

**CSS Border**



# CSS Border



- **border-style**: thiết lập kiểu hiển thị của biên.
- **border-width**: thiết lập chiều rộng biên.
- **border-color**: thiết lập màu sắc cho biên.
- **border**: thuộc tính ngắn gọn (shorthand) cho các thuộc tính trên.
- **border-radius**: giá trị bo tròn ở các góc biên.



# border-style

- **dotted**: biên là các dấu chấm nối liên tiếp.
- **dashed**: biên là các đường nét đứt.
- **solid**: biên là đường liền nhau.
- **double**: biên là hai đường liền nhau song song.
- **none**: không có biên.
- **hidden**: tạo biên ẩn.
- Ngoài ra, dùng các giá trị sau tạo biên 3D:  
groove, ridge, inset, outset.





# border-style

```
<p style="border-style: dotted;">dotted</p>
<p style="border-style: dashed;">dashed</p>
<p style="border-style: solid;">solid</p>
<p style="border-style: double;">double</p>
<p style="border-style: groove;">groove</p>
<p style="border-style: ridge;">ridge</p>
<p style="border-style: inset;">inset</p>
<p style="border-style: outset;">outset</p>
```

dotted

dashed

solid

double

groove

ridge

inset

outset



# border-style



- Tương tự border-style cũng có 4 thuộc tính tương ứng cho 4 phía của biên
  - border-**top**-style
  - border-**right**-style
  - border-**bottom**-style
  - border-**left**-style



# border-style

- Thuộc tính border-style có thể nhận 1 đến 4 giá trị.

## **border-style: dotted solid double dashed**

- border-top-style: dotted
- border-right-style: solid
- border-bottom-style: double
- border-left-style: dashed

## **border-style: dotted solid**

- border-top-style: dotted
- border-right-style: solid
- border-bottom-style: dotted
- border-left-style: solid

## **border-style: dotted solid double**

- border-top-style: dotted
- border-right-style: solid
- border-bottom-style: dashed
- border-left-style: solid

## **border-style: dotted**

- border-top-style: dotted
- border-right-style: dotted
- border-bottom-style: dotted
- border-left-style: dotted



# border-color



- **border-color** là thuộc tính chung chỉ định **màu** cho 4 biên ở 4 hướng: top, right, bottom và left.
- Để chỉ định màu biên cho từng phía khác nhau dùng các thuộc tính tương ứng sau:
  - border-**top**-color
  - border-**right**-color
  - border-**bottom**-color
  - border-**left**-color



# border-color

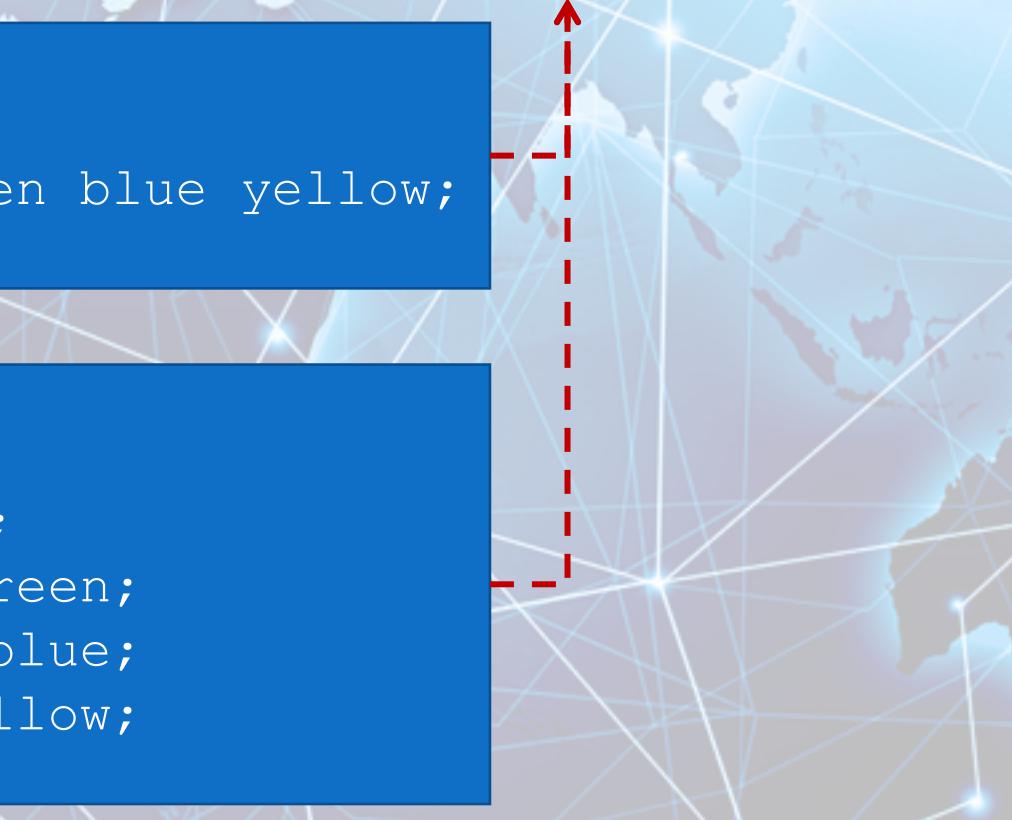
```
p {  
    border-style: solid;  
    border-color: red;  
}
```



```
p {  
    border-style: solid;  
    border-color: red green blue yellow;  
}
```



```
p {  
    border-style: solid;  
    border-top-color: red;  
    border-right-color: green;  
    border-bottom-color: blue;  
    border-left-color: yellow;  
}
```



# border-width



- Thuộc tính này thiết lập chiều rộng cho biên, nó có thể nhận giá trị là một số của pixels, cm, pt hoặc thin, medium, thick.
- Tương tự, nó cũng có 4 thuộc tính cho 4 phía:
  - border-**top**-width
  - border-**right**-width
  - border-**bottom**-width
  - border-**left**-width



# border

- Thuộc tính ngắn gọn cho tất cả thuộc tính trên với thứ tự **border-width|border-style|border-color**.
- Ví dụ: 2 CSS sau là tương đương

```
p {  
    border-style: dashed;  
    border-color: blue;  
    border-width: 5px;  
    height: 50px;  
}
```

```
p {  
border: 5px dashed blue;  
    height: 50px;  
}
```





# border

- Tương tự, thuộc tính này cũng có các thuộc tính tương ứng cho 4 phía:
  - border-top
  - border-right
  - border-bottom
  - border-left



# border-radius



- Để bo tròn các góc của biên dùng thuộc tính border-radius.

```
p {  
    border: 5px solid blue;  
border-radius: 10px;  
    height: 50px;  
}
```



- Thuộc tính này không hỗ trợ IE8 và các phiên bản trước đó.



# CSS outline



- Outline dùng thiết lập css cho đường xung quanh thành phần HTML.
- Thuộc tính outline **khác** với thuộc tính border, nó **không được tính** trong kích thước (chiều rộng, chiều cao) của thành phần HTML.
- Các thuộc tính
  - outline-style (giá trị giống border-style)
  - outline-color
  - outline-width
  - outline: outline-width|outline-style|outline-color

# Cascading Style Sheet

**CSS Margin**



# CSS Margin

- Thuộc tính margin dùng thiết lập **khoảng cách xung quanh các thành phần**.
- Sử dụng các thuộc tính sau cho margin:
  - margin-top
  - margin-right
  - margin-bottom
  - margin-left
  - margin



# CSS Margin



- Các thuộc tính margin có thể nhận các giá trị:
  - auto: trình duyệt tự tính margin.
  - Số: đơn vị px, pt, cm,...
  - Số tỷ lệ phần trăm chiều rộng so với thành phần chứa nó.
  - inherit: chỉ định margin sẽ kế thừa từ thành phần cha.
- Các thuộc tính này **có thể nhận giá trị âm.**

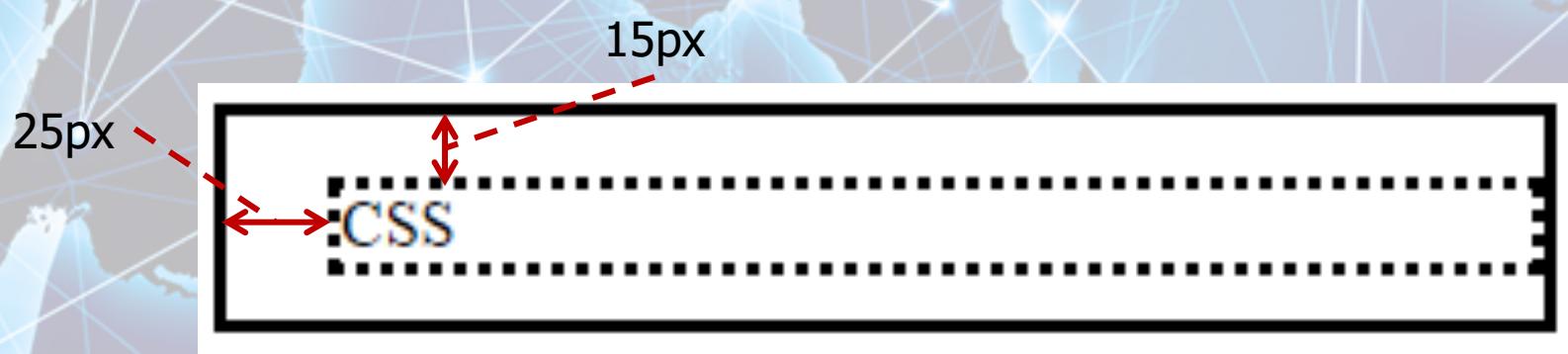


# Ví dụ

```
div {  
    border-style: solid;  
    height: 50px;  
}
```

```
div > p {  
    border-style: dotted;  
    margin-left: 25px;  
    margin-top: 15px;  
}
```

```
<div>  
    <p>CSS</p>  
</div>
```



# Thuộc tính margin

- margin là thuộc tính ngắn gọn thiết lập giá trị margin cho các hướng (top, right, bottom, left).
- Thuộc tính này có thể nhận từ 1 đến 4 giá trị.

## **margin: 1px 2px 3px 4px**

- top-margin: 1px
- right-margin: 2px
- bottom-margin: 3px
- left-margin: 4px

## **margin: 1px 2px**

- top-margin: 1px
- right-margin: 2px
- bottom-margin: 1px
- left-margin: 2px

## **margin: 1px 2px 3px**

- top-margin: 1px
- right-margin: 2px
- bottom-margin: 3px
- left-margin: 2px

## **margin: 1px**

- top-margin: 1px
- right-margin: 1px
- bottom-margin: 1px
- left-margin: 1px

# Cascading Style Sheet

**CSS List**

- **list-style-type**: thiết lập kiểu các dấu marker trước mỗi item, một số giá trị:
  - Danh sách không có thứ tự: disc, circle, square
  - Danh sách có thứ tự: lower-alpha, upper-alpha, lower-roman, upper-roman, ...
- **list-style-position**: chỉ định các marker mỗi item hiển thị bên trong hay ngoài nội dung, nhận giá trị **none**, **outside** hoặc **inside**.
- **list-style-image**: thiết lập ảnh cho các marker trước mỗi item.



# CSS List



- **list-style**: thuộc tính ngắn gọn thiết lập cho các thuộc tính trên, thứ tự thiết lập cho thuộc tính này:

list-style-type|list-style-position|list-style-image

- **marker-offset**: thiết lập khoảng cách giữa marker và văn bản trong danh sách.



# CSS List

```
ul.out {  
    list-style-type: circle;  
    list-style-position: outside;  
}  
  
ul.in {  
    list-style-type: square;  
    list-style-position: inside;  
}  
  
li {  
    background-color: gray;  
}
```

- Apple
- Orange
- Banana

- Apple
- Orange
- Banana

```
<ul class="out">  
    <li>Apple</li>  
    <li>Orange</li>  
    <li>Banana</li>  
</ul>  
<ul class="in">  
    <li>Apple</li>  
    <li>Orange</li>  
    <li>Banana</li>  
</ul>
```

# Cascading Style Sheet

**CSS Table**



# CSS Table

- Có thể sử dụng các thuộc tính CSS khác vào bảng như border, background, color, v.v.
- **border-collapse**: gộp các biên của bảng thành một đường duy nhất.
- **width/height**: thiết lập chiều rộng, chiều cao bảng.



# CSS Table



- **text-align:** canh lề nội dung văn bản trong ô (td, th) theo chiều ngang (left, right, center). Giá trị mặc định của td là left, của th là center.
- **vertical-align:** canh lề nội dung văn bản trong ô (td, th) theo chiều dọc (top, bottom, middle). Giá trị mặc định là middle.

# Ví dụ

```
<table>
  <tr>
    <th>Họ tên</th>
    <th>Quê quán</th>
    <th>Giới tính</th>
  </tr>
  <tr>
    <td>Dương Lễ</td>
    <td>Tp.HCM</td>
    <td>Nam</td>
  </tr>
  <tr>
    <td>Dương Võ</td>
    <td>Cần Thơ</td>
    <td>Nam</td>
  </tr>
</table>
```

```
table, th, td {
  border: 2px solid blue;
  padding: 5px;
}
```

Họ tên	Quê quán	Giới tính
Dương Lễ	Tp.HCM	Nam
Dương Võ	Cần Thơ	Nam

```
table, th, td {
  ...
  border-collapse: collapse;
}
```

Họ tên	Quê quán	Giới tính
Dương Lễ	Tp.HCM	Nam
Dương Võ	Cần Thơ	Nam

# Ví dụ

```
table {  
    width: 50%;  
    text-align: center;  
}  
  
/* Khi hover trên dòng thì đổi màu nền của dòng  
thành #4CAF50 */  
tr:hover {  
    background-color: #4CAF50;  
}
```

Họ tên	Quê quán	Giới tính
Dương Lễ	Tp.HCM	Nam
Dương Võ	Cần Thơ	Nam

# Ví dụ

Thiết lập các dòng chẵn có màu nền #2196F3.  
→ Để chọn dòng lẻ sửa thành nth-child(odd)

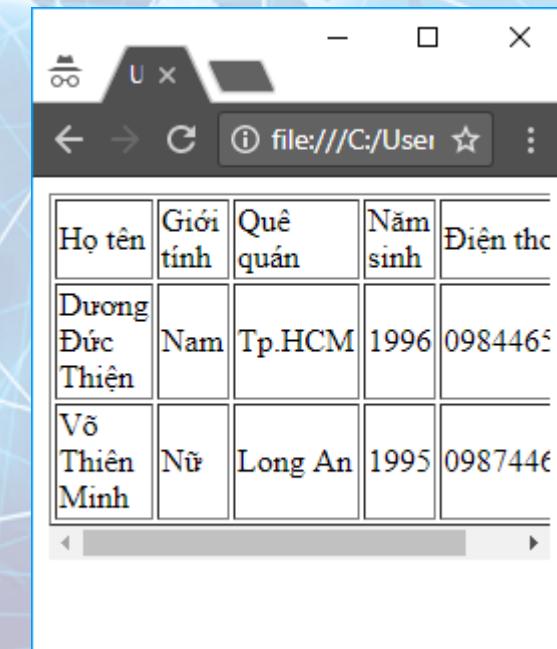
```
tr:nth-child(even) {  
background-color: #2196F3;  
}
```

Họ tên	Quê quán	Giới tính
Dương Lẽ	Tp.HCM	Nam
Dương Võ	Cần Thơ	Nam
Dương Văn	Long An	Nữ

# Ví dụ

- Hiển thị thanh trượt khi màn hình quá nhỏ để hiển thị hết nội dung bảng, bằng cách đặt toàn bộ table trong một thành phần container, chẳng hạn <div> và thuộc tính style của <div> như sau:

```
<div style="overflow-x: auto;">  
    <table>...</table>  
</div>
```



Họ tên	Giới tính	Quê quán	Năm sinh	Điện thoại
Dương Đức Thiện	Nam	Tp.HCM	1996	0984465
Võ Thiên Minh	Nữ	Long An	1995	0987446

# Cascading Style Sheet

**css Link**



- Liên kết có thể được style với nhiều thuộc tính CSS khác như font, border, background, ...
- Ngoài ra, liên kết sẽ được style khác nhau phụ thuộc trạng thái liên kết.
- Một liên kết có 4 trạng thái
  - **a:link**: liên kết bình thường, chưa click vào.
  - **a:visited**: liên kết đã từng được click vào.
  - **a:hover**: rê chuột trên liên kết.
  - **a:active**: liên kết khi đang click trên link.



# CSS Link

```
a {  
    background-color: #F1F1F1;  
    /* Bỏ gạch chân của link */  
    text-decoration: none;  
}  
  
a:link {  
    color: #1D559F;  
}  
  
a:visited {  
    color: #8F0902;  
}  
  
a:hover {  
    color: #FF9800;  
}  
  
a:active {  
    color: #00A652;  
}
```

Chú ý trật tự viết CSS sau

- a:hover phải **nằm sau** a:link và a:visited.
- a:active phải **nằm sau** a:hover.

# Ví dụ tạo link button

```
<a href="#" class="link">Button Link 1</a>
<a href="#" class="info">Button Link 2</a>
<a href="#" class="warning">Button Link 3</a>
```

```
a {
    text-decoration: none;
    text-align: center;
    display: inline-block;
    padding: 12px 14px;
    font-size: 18px;
}

a.info {
    border: 1px solid #00BAE9;
}

a.warning {
    background-color: #FB6402;
}
```

Button Link 1    Button Link 2    Button Link 3



# Ví dụ tạo link button

```
a:hover.link {  
    text-decoration: underline;  
}  
  
a:hover.info {  
    background-color: #00BAE9  
}  
  
a:hover.warning {  
    background-color: #FF0000;  
}
```



Hover trên link 1



Hover trên link 2



Hover trên link 3

# Cascading Style Sheet

**CSS Dimension**



- Để thay đổi kích thước của các thành phần dùng các thuộc tính sau:
  - **height**: thiết lập chiều cao cho thành phần.
  - **width**: thiết lập chiều rộng cho thành phần.
  - **line-height**: thiết lập chiều cao cho một dòng văn bản.
  - **max-height/min-height**: thiết lập chiều cao tối đa/tối thiểu cho thành phần.
  - **max-width/min-width**: thiết lập chiều rộng tối đa/tối thiểu cho thành phần.



# Ví dụ

```
div {  
    font-size: 16px;  
    border: 2px solid blue;  
    width: 30%;  
    line-height: 30px;  
    max-height: 90px;  
}
```

CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once.



CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once.

CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once.

# Cascading Style Sheet

**CSS Overflow và Float**



- Thuộc tính overflow chỉ định **xén** (clip) nội dung hoặc **thêm thanh trượt** (scrollbar) khi nội dung của thành phần vượt quá diện tích hiển thị của thành phần đó.
- Để thiết lập overflow theo chiều ngang hoặc chiều dọc dùng các thuộc tính tương ứng sau:
  - **overflow-x**
  - **overflow-y**
- Chú ý: các thuộc tính này chỉ làm việc với các thành phần dạng **block có chiều cao xác định**.



- Các giá trị có thể cho thuộc tính overflow
  - **visible**: không xén nội dung và hiển thị tràn ra phạm vi thành phần (mặc định).
  - **hidden**: xén nội dung và phần nội dung còn lại sẽ không được hiển thị.
  - **scroll**: xén nội dung và thêm thanh trượt để thấy luôn phần nội dung còn lại.
  - **auto**: xén nội dung và chỉ thêm thanh trượt khi cần thiết.

# Ví dụ

```
div {  
    background-color: gray;  
    height: 80px;  
    width: 180px;  
    overflow: scroll;  
}
```

<div>CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once.</div>

CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once.

CSS describes how HTML elements are to be displayed on screen, paper, or in other media. CSS saves a lot of work. It can control the layout of multiple web pages all at once.

visible

hidden

CSS describes how HTML elements are to be displayed on screen, paper, or in other media.

scroll

CSS describes how HTML elements are to be displayed on screen, paper, or in other media.

auto



# CSS Float

- Thuộc tính float chỉ định một thành phần “**nổi lên**” để tạo không gian trống cho các thành phần phía sau di chuyển lên và chiếm chỗ trống đó.
- Ví dụ: chỉ định một hình ảnh nổi bên phải phần văn bản. Xét HTML sau:

```
<div>
    
    <p>HTML describes the structure of Web pages using
markup</p>
    <p>CSS describes how HTML elements are to be
displayed on screen, paper, or in other media</p>
    <p>JavaScript is the programming language of HTML and
the Web.</p>
</div>
```



# CSS Float

```
div {  
    width: 400px;  
    background-color: grey;  
}
```

```
img {  
    float: right;  
}
```



HTML describes the structure of Web pages using markup

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

JavaScript is the programming language of HTML and the Web.

HTML describes the structure of Web pages using markup

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

JavaScript is the programming language of HTML and the Web.



Có thiết lập float cho img

Không thiết lập float cho img



# CSS Clear

- Thuộc tính **clear** chỉ định hướng của thành phần mà các thành phần khác **không được** float.
- Ví dụ: trở lại ví dụ trên và thêm CSS như sau để chỉ định không được float bên phải thành phần p.

```
p {  
    clear: right;  
}
```



HTML describes the structure of Web pages using markup

CSS describes how HTML elements are to be displayed on screen, paper, or in other media

JavaScript is the programming language of HTML and the Web.



# clearfix

```
<div class="clearfix">
    
    <p>HTML, CSS, Javascript</p>
</div>
```

```
div {
    width: 400px;
    background-color: grey;
}
img {
    float: right;
}
```

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

HTML, CSS, Javascript



HTML, CSS, Javascript



# Cascading Style Sheet

**CSS Position**

# CSS Position



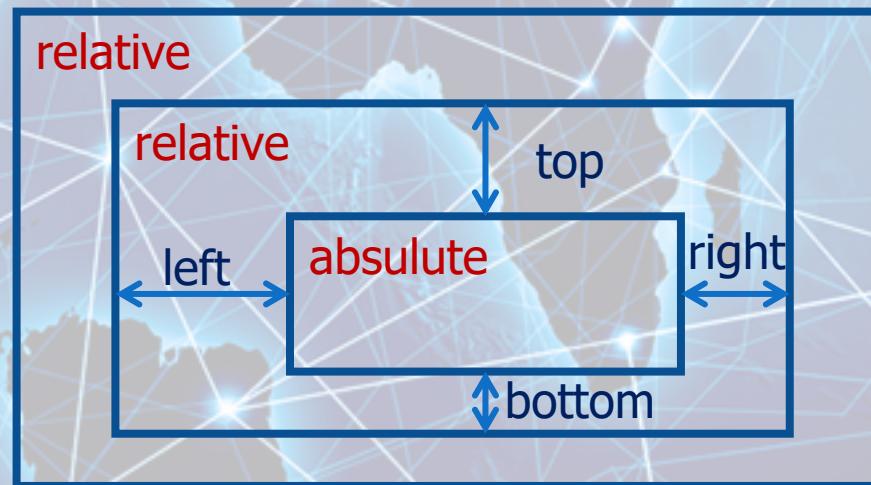
- Thuộc tính **position** chỉ định kiểu vị trí thiết lập cho một thành phần.
- Thuộc tính này có thể nhận các giá trị:
  - static (**mặc định**)
  - relative
  - fixed
  - absolute



- **static**: thành phần luôn nằm một vị trí, không bị ảnh hưởng bởi các thuộc tính top, left, right, bottom.
- **fixed**: thành phần luôn nằm cố định ở một vị trí, ngay cả khi kéo trượt trang. Các thuộc tính top, left, right, bottom dùng chỉ định vị trí thành phần.

# CSS Position

- **relative**: thành phần có thể đặt ở bất kỳ vị trí nào phụ thuộc giá trị các thuộc tính top, left, right, bottom.
- **absolute**: thành phần có thể đặt vị trí tương đối trong thành phần cha gần nó nhất và phải là thành phần relative, nếu không sẽ là body.



# Thuộc tính z-index



- Khi thiết lập vị trí cho các thành phần thì sẽ có trường hợp thành phần này **nằm chồng** (overlap) lên thành phần khác.
- Để chỉ định được trật tự các thành phần khi hiển thị dùng thuộc tính **z-index**, giá trị z-index của thành phần **lớn hơn sẽ nằm trên** thành phần có giá trị z-index nhỏ hơn.
- Chú ý: z-index chỉ có **hiệu lực** khi position có giá trị là **absolute, relative, hoặc fixed**.

# Thuộc tính z-index



- Cú pháp:

```
z-index: auto | number | initial | inherit;
```

- Ví dụ:

```
<h1 style="color:yellow;">CSS3</h1>

```

```
img {
    position: absolute;
    top: 0;
    left: 0;
    z-index: -1;
}
```

CSS3

# Cascading Style Sheet

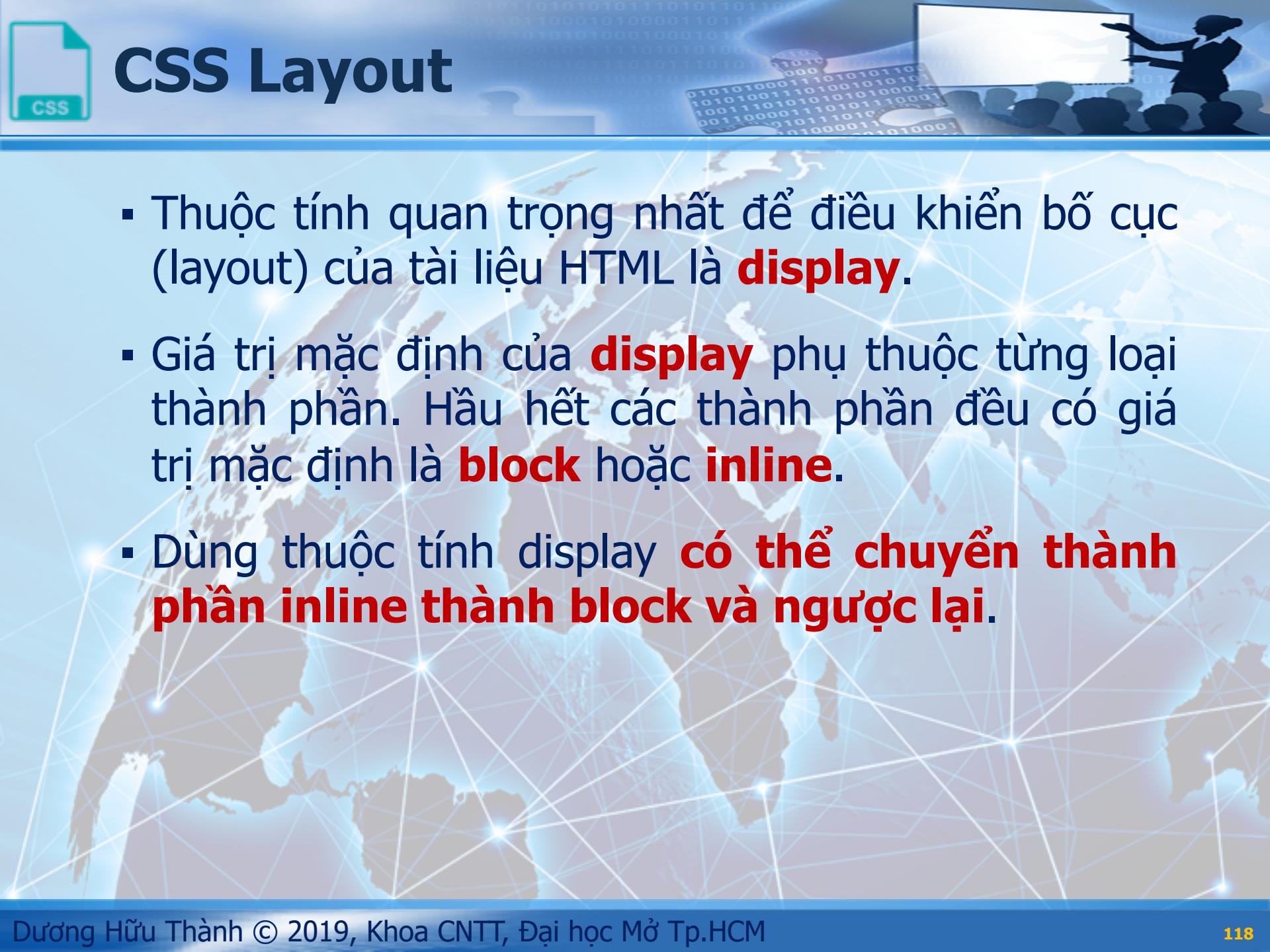
**CSS Layout**



# CSS Layout



- Thuộc tính quan trọng nhất để điều khiển bố cục (layout) của tài liệu HTML là **display**.
- Giá trị mặc định của **display** phụ thuộc từng loại thành phần. Hầu hết các thành phần đều có giá trị mặc định là **block** hoặc **inline**.
- Dùng thuộc tính display **có thể chuyển thành phần inline thành block và ngược lại**.



# Ẩn một thành phần



- Để ẩn một thành phần có thể dùng các CSS sau:
  - **display: none** → ẩn thành phần và có thể bị các thành phần khác chiếm vùng hiển thị.
  - **visibility: hidden** → ẩn thành phần, nhưng không gian nó chiếm vẫn giữ nguyên đó.



# Giá trị inline-block



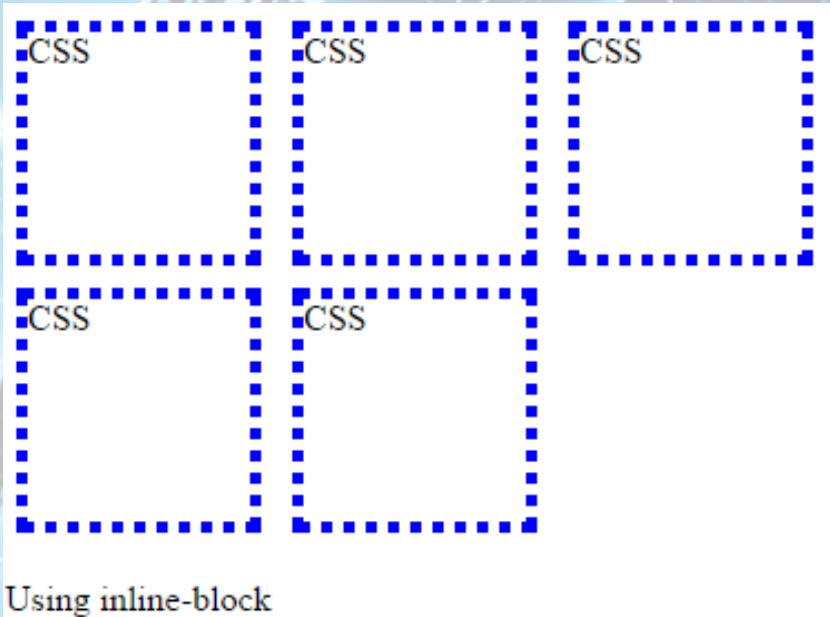
- Nếu thuộc tính display nhận giá trị là **inline-block** thì các thành phần đó hiển thị cũng giống thành phần inline, nhưng **được phép** thiết lập chiều rộng và chiều cao cho các thành phần.
- Ví dụ: có HTML như sau

```
<div>CSS</div>
<div>CSS</div>
<div>CSS</div>
<div>CSS</div>
<div>CSS</div>
<p>Using inline-block</p>
```



# Giá trị inline-block

```
div {  
    border: 5px dotted blue;  
    width: 100px;  
    height: 100px;  
    margin: 5px;  
    display: inline-block;  
}
```



# Giá trị flex

- Giá trị **display:flex** giúp thiết kế layout responsive dễ dàng hơn, mà không cần sử dụng float hay position.
- Ví dụ: cho HTML như sau

```
<div class="flex">
  <div class="f">
    <div class="f">
      <div class="f">
        </div>
    </div>
```

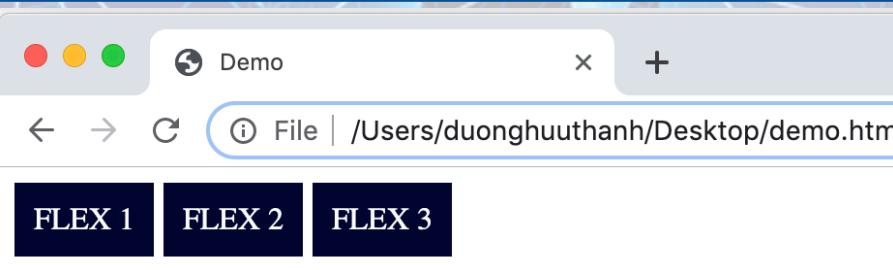
```
.flex-container {
  display:flex;
}

.flex-item {
  color:#FAFAFA;
  background-color:#020732;
  padding: 10px;
  margin-right:5px;
}
```

# Giá trị flex

- Sử dụng `display:flex` như sau

```
.flex-container {  
    display:flex;  
    flex-direction:row;  
    flex-wrap: wrap;  
}  
  
.flex-item {  
    color:#FAFAFA;  
    background-color:#020732;  
    padding: 10px;  
    margin-right:5px;  
}
```



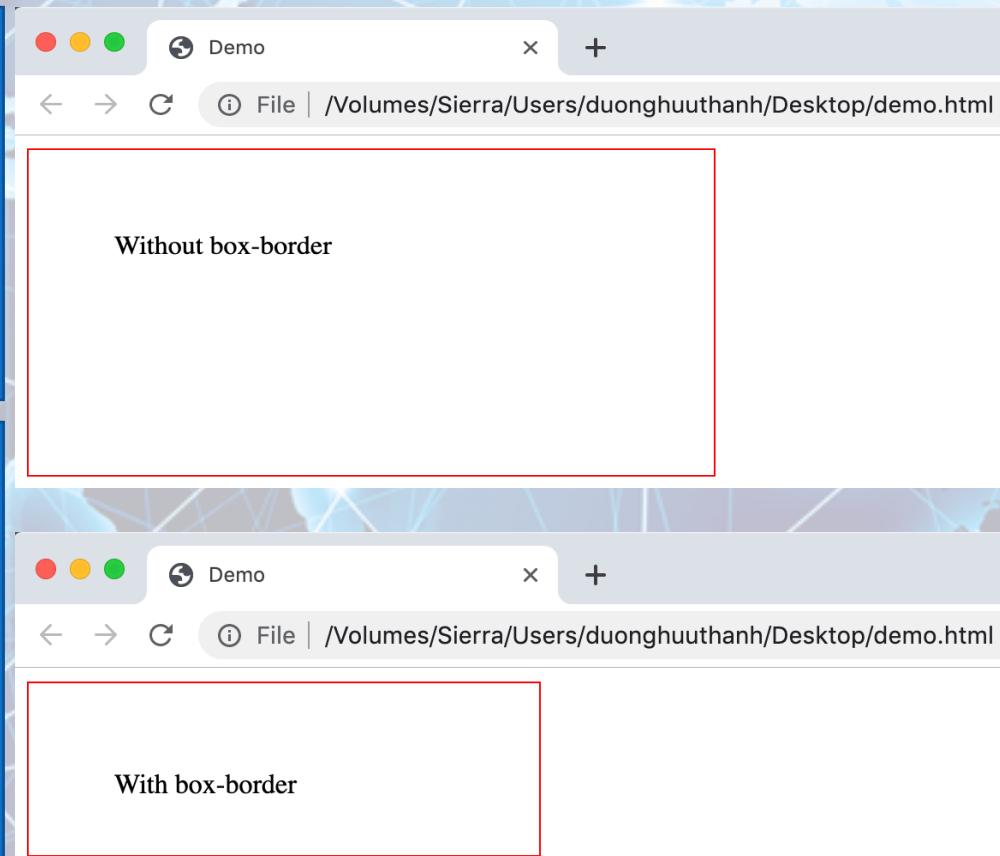


# Box sizing

- Thuộc tính **box-sizing:border-box** sẽ tính border, padding vào kích thước thành phần HTML.

```
div {  
    width:300px;  
    height:100px;  
    border:1px solid red;  
    padding:50px;  
}
```

```
div {  
    width:300px;  
    height:100px;  
    border:1px solid red;  
    padding:50px;  
box-sizing:border-box;  
}
```



# Cascading Style Sheet

**CSS GRDDIENT**



# CSS Gradient



- CSS Gradient trộn nhiều màu để hiển thị, có hai loại gradient:
  - linear-gradient(direction/angle, colors...)
  - radial-gradient(colors...)

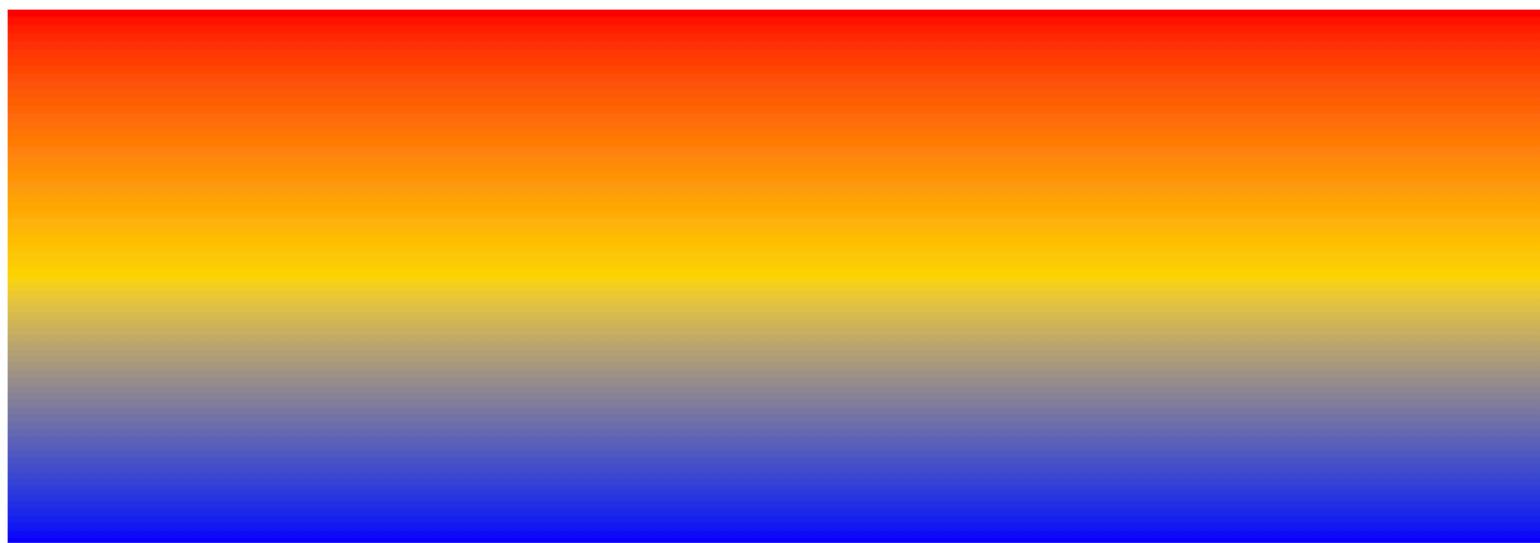




# CSS Gradient

- Ví dụ

```
div {  
    width:100%;  
    height:200px;  
    background-image:linear-gradient(red, gold, blue);  
}
```

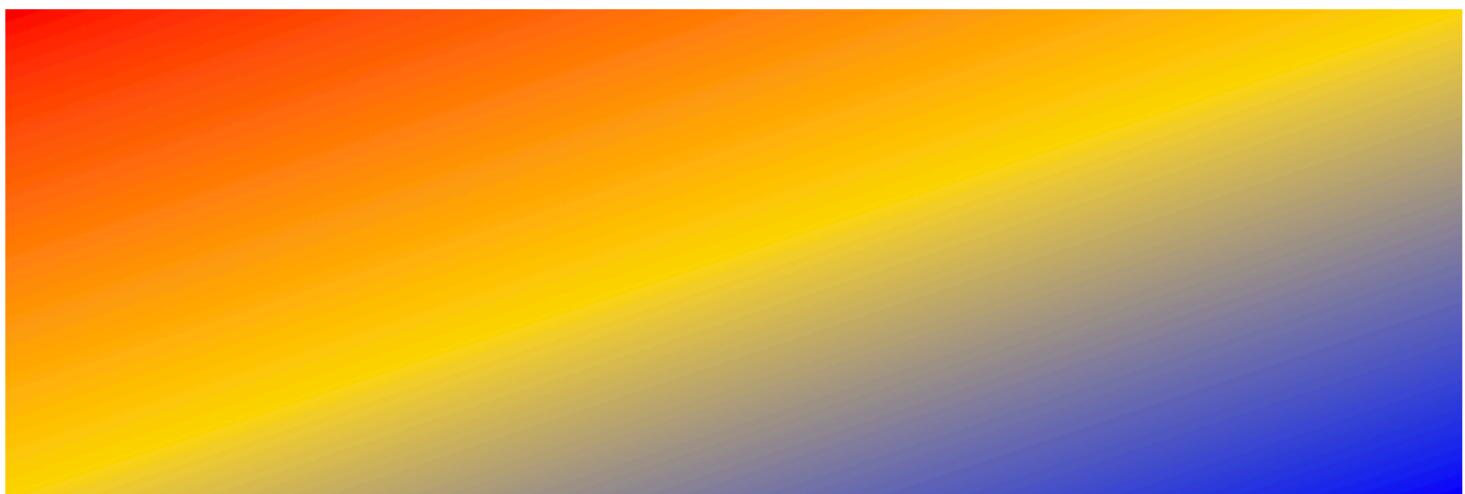




# CSS Gradient

- Ví dụ

```
div {  
    width:100%;  
    height:200px;  
    background-image:linear-gradient(to bottom right,  
    red, gold, blue);  
}
```



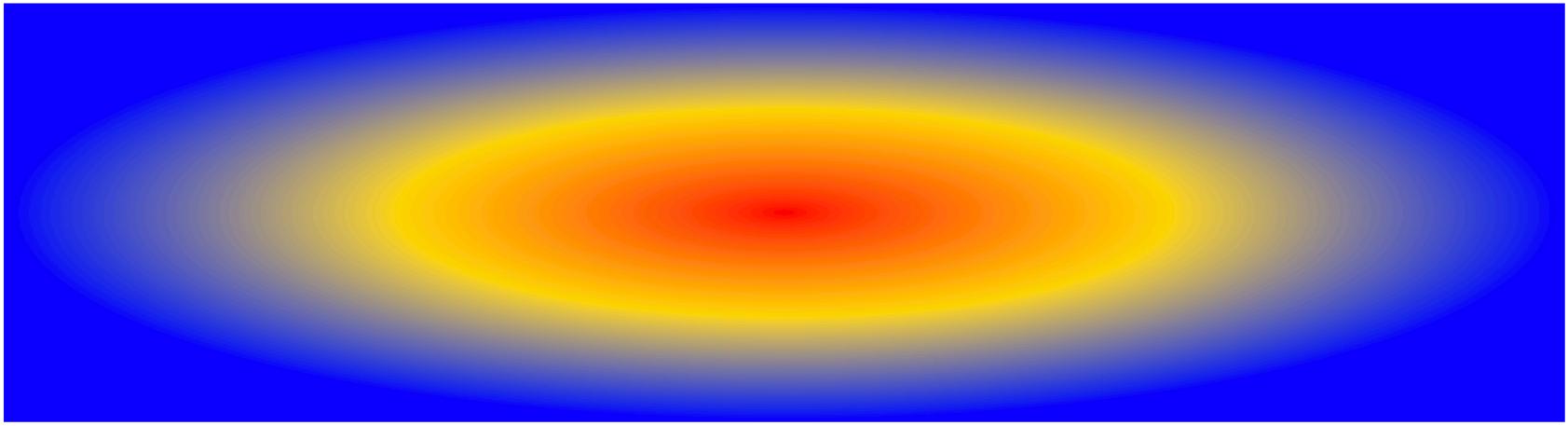


# CSS Gradient



- Ví dụ

```
div {  
    width:100%;  
    height:200px;  
    background-image:radial-gradient(red, gold, blue 70%);  
}
```



# Cascading Style Sheet

**CSS TRANSFORM**



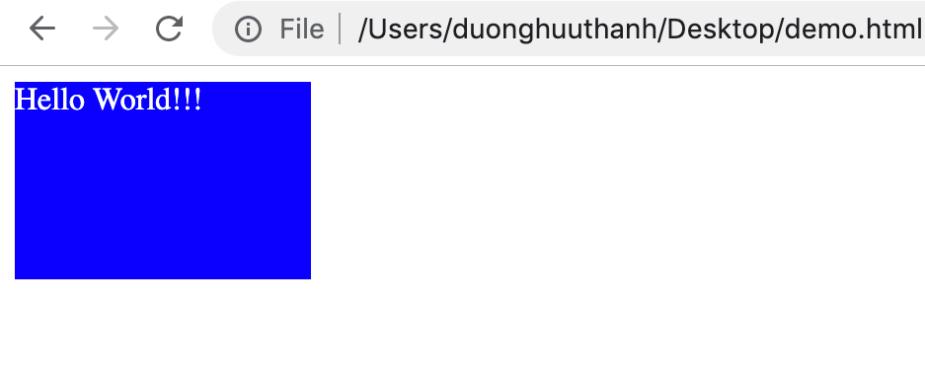
# CSS Transform



- CSS Transform cho phép dịch chuyển (translate), quay (rotate), co dãn (scale), nghiêng (skew) thành phần HTML.
- Ví dụ: Cho HTML và CSS như sau

```
<div>Hello World!!!</div>
```

```
div {  
    width:150px;  
    height:100px;  
    background-color:blue;  
    color:white;  
}
```





# CSS Transform

- Dịch chuyển (translate)

```
div {  
    transform: translate(20px, 50px);  
    /* Chrome, Safari */  
    -webkit-transform: translate(20px, 50px);  
    /* Opera */  
    -o-transform: translate(20px, 50px);  
    /* IE 9 */  
    -ms-transform: translate(20px, 50px);  
    /* Firefox */  
    -moz-transform: t  
}
```

← → ⌂ File | /Users/duonghuuthanh/Desktop/demo.html

Hello World!!!



# CSS Transform

- Quay (rotate)

```
/* Quay một góc 30 độ theo  
chiều kim đồng hồ*/  
div {  
    transform: rotate(30deg);  
}
```



```
/* Quay một góc 30 độ ngược  
chiều kim đồng hồ */  
div {  
    transform: rotate(-30deg);  
}
```





# CSS Transform

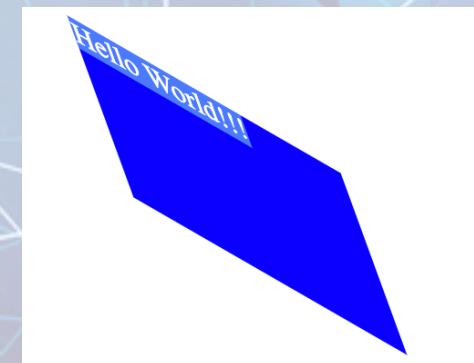


- Co dãn (scale)

```
/* Tăng 2 lần chiều rộng, 3 lần chiều cao so  
với kích thước gốc */  
div {  
    transform:scale(2, 3);  
}
```

- Nghiêng (skew)

```
/* Nghiêng chiều ngang 20 độ,  
chiều dọc 30 độ */  
div {  
    transform:skew(20deg, 30deg);  
}
```



# Cascading Style Sheet

**CSS TRANSITION**



# CSS Transition



- CSS Transition dùng tạo hiệu ứng sự thay đổi giá trị của một thuộc tính CSS nào đó trong khoảng thời gian chỉ định (duration). Các thuộc tính
  - transition-property: tên thuộc tính CSS sẽ thay đổi.
  - transition-duration: khoảng thời gian thực hiện hiệu ứng (giây hoặc mili giây).
  - transition-delay: thời gian delay (giây).
  - transition-timing-function: hàm chỉ định cách thực hiện hiệu ứng (ease, ease-out, ease-in, ...).
  - transition: thuộc tính ngắn gọn thiết lập chung cho các thuộc tính trên.



# CSS Transition



- Ví dụ thực hiện thay đổi background-color của thẻ div khi hover.

```
div {  
    width:150px;  
    height:100px;  
    color:white;  
    background-color:blue;  
    transition: background-color ease 2s;  
}  
div:hover {  
    background-color:red;  
}
```

- Chạy và rê chuột trên thẻ div để xem kết quả.

# Cascading Style Sheet

**CSS ANIMATION**



# CSS Animation



- CSS Animation dùng tạo hiệu ứng trang web mà không cần javascript hay flash.
- Để sử dụng CSS Animation cần định nghĩa các keyframes, dùng chỉ định những thay đổi từ style này sang style khác.
- Ví dụ định nghĩa

```
@keyframes doimau {  
    from { background-color:red; }  
    to { background-color:blue; }  
}
```

```
@keyframes doimau2 {  
    0% { background-color:red; }  
    25% { background-color:blue; }  
    50% { background-color:green; }  
    100% { background-color:gold; }  
}
```



- Các thuộc tính
  - animation-name: tên của keyframes.
  - animation-delay: khoảng thời gian delay để bắt đầu hiệu ứng.
  - animation-duration: khoảng thời gian thực hiện hiệu ứng.
  - animation-iteration-count: số lần thực hiện hiệu ứng (dùng giá trị **infinite** để thực hiện vô hạn).
  - animation-timing-function: hàm chỉ định các thực hiện hiệu ứng (ease, ease-in, ease-out, ...).
  - animation: thuộc tính ngắn gọn.



# CSS Animation



- Ví dụ thực hiện hiệu ứng đổi các màu nền liên tục sử dụng keyframes doimau2 đã định nghĩa ở trên, mỗi lần của hiệu ứng thực hiện trong 3 giây.

```
div {  
    width:150px;  
    height:100px;  
    animation:doimau2 3s ease infinite;  
}
```

- Tạo thẻ div trong HTML và chạy trang web trên trình duyệt để xem kết quả.

# Q&A