

KHOA CÔNG NGHỆ THÔNG TIN VÀ TRUYỀN THÔNG
BỘ MÔN MẠNG MÁY TÍNH VÀ TRUYỀN THÔNG

LẬP TRÌNH WEB



Chương 4: CSS

Biên soạn : ThS. Hà Duy An

Cập nhật : ThS. Nguyễn Cao Hồng Ngọc

NỘI DUNG



- Giới thiệu CSS
- Cú pháp CSS
- Vị trí đặt CSS
- Nguyên tắc áp dụng
- Thuộc tính CSS
- Các thiết kế nâng cao

GIỚI THIỆU CSS



- CSS viết tắt từ **C**ascading **S**tyle **S**heets
- HTML được thiết kế để mô tả cấu trúc, nội dung của trang web
- CSS được thiết kế để định dạng cách hiển thị nội dung của trang web
- Các kiểu (styles) xác định cách thức trình bày các phần tử HTML
- Các kiểu được bổ sung vào HTML 4.0 để giải quyết vấn đề về hiển thị
- Việc dùng các bảng kiểu ngoài (External Style Sheets) làm giảm khối lượng công việc đáng kể
- Các bảng kiểu ngoài được lưu vào các tập tin có phần mở rộng là `.css`

Tại sao là CSS?

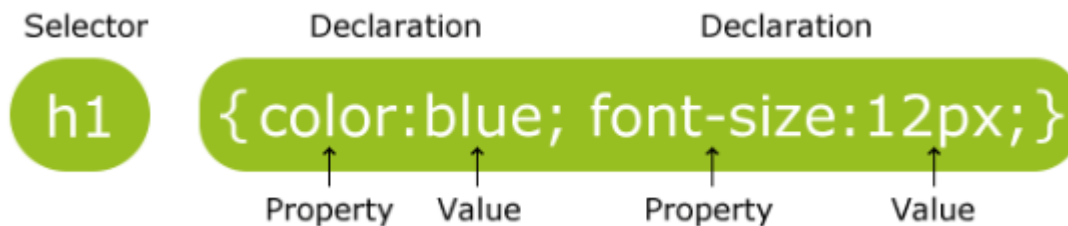


- HTML không chứa các thẻ định dạng tài liệu
- HTML chỉ định nghĩa nội dung của tài liệu dưới các dạng như:
`<h1>This is a heading</h1>`
`<p>This is a paragraph</p>`
- Khi thẻ `` và thuộc tính `color` được thêm tới đặc tả HTML 3.2, nó phát sinh ra vấn đề làm đau đầu các nhà phát triển web: Việc phát triển các website lớn, với thẻ `font` và thuộc tính `color` thêm vào mỗi trang, trở nên tốn kém thời gian và chi phí nhiều hơn
- Để giải quyết vấn đề này, World Wide Web Consortium (W3C) tạo ra CSS
- Trong HTML 4.0, tất cả các định dạng được gỡ khỏi tài liệu HTML, và được lưu vào một tập tin CSS riêng.
- Tất cả các trình duyệt ngày nay đều hỗ trợ CSS.

CÚ PHÁP CSS



- **Cú pháp cơ bản:** Một luật css (css rule) có 2 phần: một **bộ chọn** (selector), và một/ nhiều **khai báo** (declaration)



- Trong đó:
 - Selector (bộ chọn): xác định các thành phần HTML sẽ được áp dụng kiểu tương ứng
 - Property (thuộc tính): các thuộc tính định dạng
 - Value: giá trị các thuộc tính tương ứng
- Mỗi khai báo chứa một cặp thuộc tính & giá trị
- Thuộc tính & giá trị được ngăn cách nhau bởi dấu ":", các khai báo được ngăn cách nhau bởi dấu ";".
- Mỗi khai báo nên đặt trên một dòng

Cú pháp cơ bản (tt)



- Nhóm các bộ chọn: khi một kiểu được áp dụng cho nhiều bộ chọn ta có thể kết hợp chúng lại với nhau. Cú pháp: `selector1, selector2,... {property:value}`

Ví dụ:

↪ `h1{color:red;} h2{color:red;} h3{color:red;}
h1,h2,h3{color:red;}`

- Chú thích `/* ...*/` (áp dụng cho cả nhiều dòng và 1 dòng)

Ví dụ: `p{`

```
    color:black;  /* Font color is black */  
    text-align:center;  
    font-family:arial;  
}
```

Các loại bộ chọn



- Có 4 loại bộ chọn cơ bản:
 - HTML Selector
 - Class Selector
 - ID Selector
 - Attribute Selector

Các loại bộ chọn (tt)



■ HTML Selector

- Bộ chọn chính là tên các thẻ trong HTML
- Kiểu sẽ áp dụng cho tất cả các thẻ tương ứng trên toàn bộ trang
- **Ví dụ:** Với khai báo CSS sau toàn bộ nội dung của các thành phần h2 trong trang web sẽ có font chữ màu đỏ

HTML Selector

h2 { color : red; }

Các loại bộ chọn (tt)



■ Class Selector

- Với bộ chọn class có thể định nghĩa các kiểu cho một/ một nhóm các thành phần HTML
- Các kiểu với bộ chọn class sẽ chỉ được áp dụng cho các thành phần HTML có thuộc tính class với giá trị là tên bộ chọn class tương ứng. **Ví dụ:**



- Để áp dụng các kiểu của *bộ chọn class* trên cho thành phần *h2* trong trang HTML ta thực hiện như sau:

```
<h2 class="highlight">Chapter I...</h2>
```

Các loại bộ chọn (tt)



■ Class selector (tt)

- **Dependent Class:** cho phép xác định các kiểu của một class chỉ có thể áp dụng trên một loại thẻ HTML => khả năng tạo một kiểu chung cho một class, và định nghĩa các kiểu riêng cho các thẻ HTML xác định trong class đó.

Ví dụ:



`<h2 class="highlight">Chapter I...</h2>` => Green Background
`<p class="highlight">I should...</p>` => Yellow Background

Các loại bộ chọn (tt)



■ Class selector (tt)

- **Trộn và ghép các class:** có thể thêm nhiều class vào một thẻ HTML để trộn và ghép các kiểu cần sử dụng, bằng cách đưa tất cả các class cần vào thuộc tính class và phân cách nhau với ký tự khoảng trắng.

Ví dụ:



```
<p class="highlight smallprint">I should...</p>
```

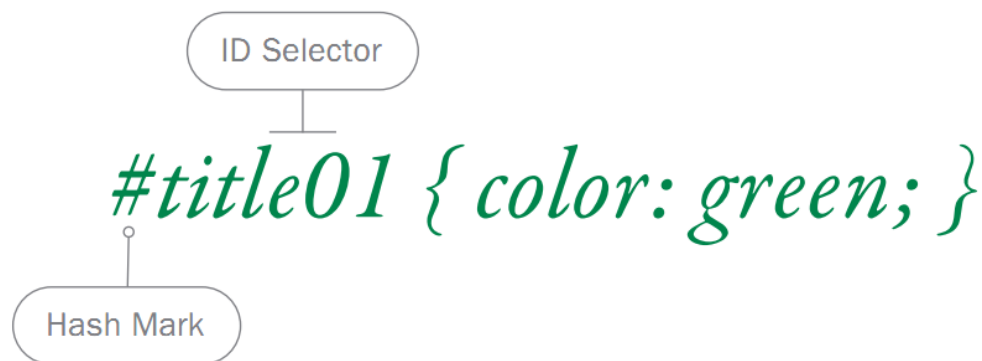
Các loại bộ chọn (tt)



■ ID Selector

- Bộ chọn ID dùng để chỉ định kiểu cho **duy nhất** một thành phần HTML
- Các kiểu của bộ chọn ID chỉ được áp dụng cho duy nhất các thành phần HTML có thuộc tính ID với giá trị là tên bộ chọn ID tương ứng

Ví dụ:



Để áp dụng các kiểu của *bộ chọn ID* trên cho thành phần *h2* trong trang HTML ta thực hiện như sau:

```
<h2 id="title01">Chapter I...</h2>
```

Các loại bộ chọn (tt)



- Quy tắc đặt tên cho bộ chọn class và ID
 - Bắt đầu bằng một ký tự A-Z hay a-z
 - Theo sau bởi các ký tự (A-Z, a-z), số, hay các ký tự "-", "_"
 - Tên phân biệt HOA hay thường
 - Ví dụ các tên hợp lệ: *greenText*, *Font34*, *some_style*, *_newStyle*, *-italicStyle*
 - Ví dụ các tên không hợp lệ: *@RedBorder*, *4_xyz*
 - **Lưu ý:** nên đặt các tên có ý nghĩa

Các loại bộ chọn (tt)



■ Attribute Selector

- Xác định kiểu cho các thành phần HTML có một thuộc tính xác định
- **Lưu ý:** IE7 và IE8 chỉ hỗ trợ attribute selector khi có khai báo !DOCTYPE. IE6 và các phiên bản IE thấp hơn không hỗ trợ attribute selector

| Selector | Example | Example description |
|------------------------------|-----------------|--|
| <u>[attribute]</u> | [target] | Selects all elements with a target attribute |
| <u>[attribute=value]</u> | [target=_blank] | Selects all elements with target="_blank" |
| <u>[attribute~=value]</u> | [title~=flower] | Selects all elements with a title attribute containing the word "flower" |
| <u>[attribute =language]</u> | [lang =en] | Selects all elements with a lang attribute value starting with "en" |

- *Kết hợp giữa HTML Selector & Attribute Selector:* Xác định kiểu cho 1 loại thành phần HTML có 1 thuộc tính xác định

Ví dụ: `input[type="text"] {background-color:blue}`

⇒ tất cả phần tử nhập (input) mà có thuộc tính kiểu với giá trị là "text" sẽ có màu nền là màu xanh

Kiểu trong ngữ cảnh



- **Descendent Selector Context:** Các kiểu có thể được áp dụng cho một thành phần HTML tùy thuộc vào các thẻ HTML, class hay ID của các thành phần cha của chúng

Ví dụ 1:

```
h1 i {color: red;}
```

⇒ nếu thành phần i (chữ nghiêng) nằm trong thành phần h1 (tiêu đề) thì có màu chữ là màu đỏ

Ví dụ 2:

```
#menu li a {color: red;}
```

⇒ liên kết nằm trong phần tử của danh sách của thành phần có ID là menu thì có màu chữ là màu đỏ

Kiểu trong ngữ cảnh (tt)



- **Styles for Children:** Xác định kiểu cho một thành phần HTML là *con trực tiếp* của một thành phần HTML khác

Ví dụ: `p>a {color: red;}`

- Áp dụng cho `<p>I am Obama</p>`
- Không áp dụng cho `<p><i>I am Obama</i></p>`

- **Styles for Siblings:** xác định kiểu cho một thành phần HTML cùng cấp kế cận tiếp theo với thành phần HTML hiện hành

Ví dụ: `i+u {color: red;}`

- Áp dụng cho `<p>I love <i>cats</i> and <u>dogs</u></p>`
- Không áp dụng cho `<p>I love cats and <u>dogs</u></p>`

Kiểu cho các trường hợp đặc biệt



- **Pseudo-class:** dùng để xác định trạng thái (rê chuột lên, phần tử chẵn, phần tử lẻ,...) của một thành phần
- Cú pháp: **<selector>:<psedo-class>{property:value;}**
- **Ví dụ:** Các trạng thái có thể có của một liên kết:
 - **Link:** trạng thái khi liên kết chưa được kích hoạt
 - **Hover:** trạng thái khi người dùng di chuyển chuột qua
 - **Active:** trạng thái khi người dùng click chuột vào
 - **Visited:** trạng thái khi liên kết đã được kích hoạt

```
a:link{color:red;}  
a:visited{color:darkred;}  
a:hover{color:hotpink; }  
a:active{color:fuchsia;}
```

Kiểu cho các trường hợp đặc biệt (tt)



- **Pseudo-element**: dùng để xác định phần đặc biệt (ký tự đầu tiên, dòng đầu tiên,...) của một thành phần
- Cú pháp: `<selector>::<pseudo-elements>{property:value;}`
- Ví dụ:

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

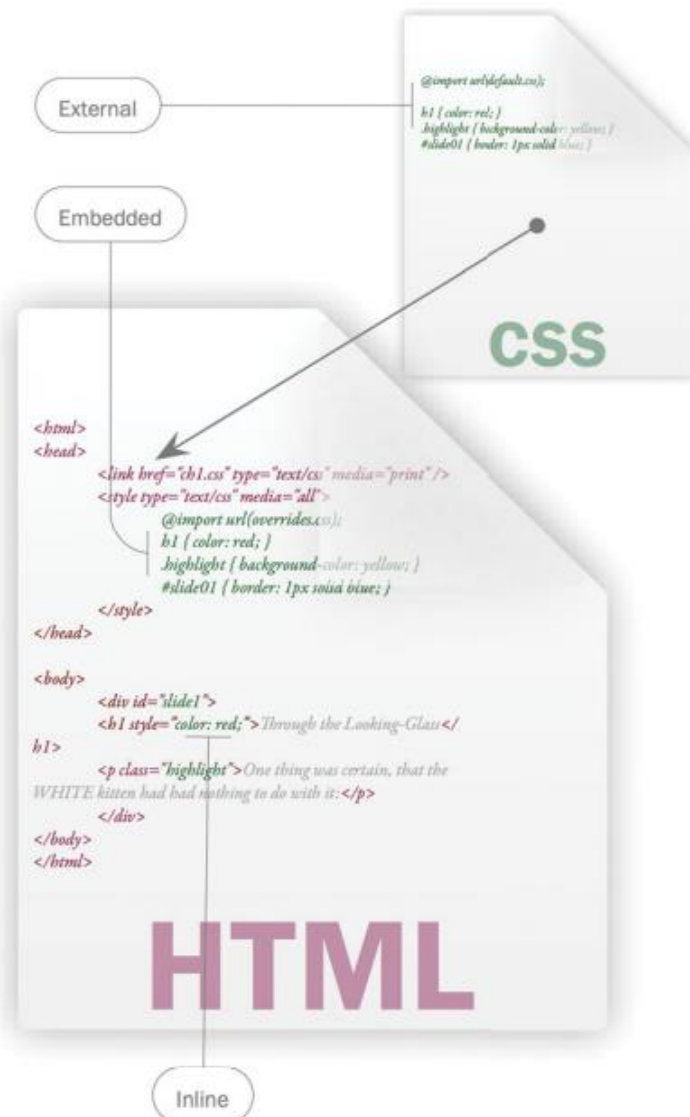
Áp dụng cho `<p>I am Obama.</p>`

- Xem thêm:
 - https://www.w3schools.com/css/css_pseudo_classes.asp
 - https://www.w3schools.com/css/css_pseudo_elements.asp

VỊ TRÍ ĐẶT CSS



- **Inline style:** các luật CSS được đặt trực tiếp vào một thẻ HTML sử dụng thuộc tính *style*
- **Internal style sheet** (embedded): các luật CSS được đặt trong thẻ *<style>* để định dạng cho trang hiện hành
- **External style sheet:** Các luật CSS được đặt trong các file riêng biệt (.css) có thể được truy cập bởi bất kỳ trang web nào bằng cách dùng thẻ *<link>*



Inline style



- Các khai báo được để trực tiếp vào **một thẻ HTML** thông qua thuộc tính *style*
- Không cần bộ chọn, các kiểu chỉ được áp dụng cho thành phần HTML chứa thuộc tính style tương ứng
- Ưu/ khuyết:
 - Độ ưu tiên cao hơn internal style sheet và external style sheet
 - Không thể tạo các kiểu định dạng chung cho toàn trang

Ví dụ:

```
<h1 style="color: red; font-family: Georgia; text-align: center;"> Through the Looking-Glass</h1>
```

Internal style sheet



- Các kiểu sẽ có phạm vi ảnh hưởng trên **toàn bộ 1 trang** web chứa nó
- Các luật CSS nằm trong cặp thẻ `<style> ... </style>`
- Đặt thẻ `<style>` trong thẻ `<head>` và nên đặt trước các đoạn mã JavaScript

```
<style>
body {
    background-color: linen;
}

h1 {
    color: maroon;
    margin-left: 40px;
}
</style>
```

External style sheet



- Có thể định dạng cho **toàn bộ website** bằng cách dùng các bảng kiểu ngoài
- File .css là các file chỉ chứa mã CSS (không có thể `<style>`)
- Có thể dùng bất cứ trình soạn thảo thuần văn bản nào để soạn thảo các file .css
- Sử dụng thẻ `<link>` (đặt trong thẻ `<head>` và nên trước mã lệnh JavaScript) để trỏ đến tập tin .css

```
<link href="URL" type="text/css" media="all" />
```

Ví dụ:

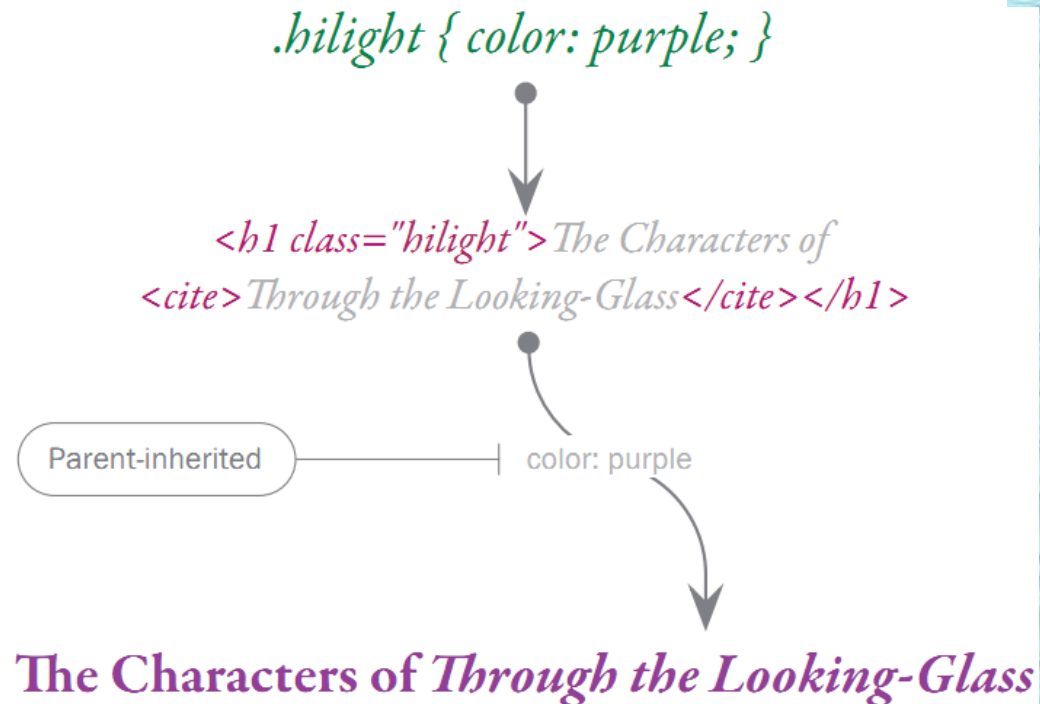
```
<link href="default.css" type="text/css" media="all" />
```


NGUYÊN TẮC ÁP DỤNG CSS



- Việc áp dụng một kiểu định dạng cho một thành phần HTML tuân theo các luật sau: **Inline styles, Media, Importance, Specificity, Order, Parent-inherited, Browser-default**

- Inheritance** (sự thừa kế): hầu như tất cả các thẻ HTML đều bị ảnh hưởng bởi các kiểu định dạng không trực tiếp. Các kiểu này có thể là kiểu mặc định của trình duyệt, hay bị ảnh hưởng bởi kiểu của các thẻ cha (parent tags). Các kiểu được thừa kế thì dễ dàng được ghi đè bằng việc dùng các bảng kiểu CSS.



Order (thứ tự)



- Có thể dễ dàng viết đè lên một kiểu của bất kỳ bộ chọn nào bằng cách khai báo lại bộ chọn với các thuộc tính giống nhau nhưng khác giá trị => các kiểu **khai báo sau cùng sẽ được áp dụng** vào trang

h1 {color: red;} ❌

h1 {color: blue;} ✅

<h1>What color am I?</h1>

Specificity (đặc trưng)



- Xác định thứ tự ảnh hưởng của các kiểu phụ thuộc vào bộ chọn ngữ cảnh => mức độ **đặc trưng** của bộ chọn càng cao độ ưu tiên các kiểu của nó được áp dụng cho một phần tử càng cao (bắt kể thứ tự của bộ chọn)
- Mức độ đặc trưng của bộ chọn ngữ cảnh phụ thuộc vào các kiểu bộ chọn được sử dụng trong bộ chọn ngữ cảnh, mỗi bộ chọn có một trọng số ưu tiên riêng:
 - Bộ chọn HTML: 1
 - Bộ chọn Class & Attribute: 10
 - Bộ chọn ID: 100
 - Inline style: ∞

```
h1 {color: red;}  
#title {color: green;} ✓  
.blue {color: blue;}
```

```
<h1 id="title" class="blue">What color am I?</h1>
```

Importance



- Có thể dùng cú pháp *!important* để xác định một kiểu đặc biệt nào đó sẽ **ưu tiên được sử dụng** khi có sự cạnh tranh thứ tự áp dụng các kiểu CSS lên một thành phần HTML => mức độ ưu tiên cao hơn các luật inheritance, order và specificity.

```
h1 {color: red !important;} ✓  
#title {color: green;}  
.blue {color: blue;}
```

```
<h1 id="title" class="blue">  
What color am I?  
</h1>
```

Media



- Media hay phương tiện dùng để trình bày nội dung một trang web
- Các trang web có thể được xuất ra các thiết bị khác nhau, có 4 giá trị cho thuộc tính *media*:
 - *Screen*: màn hình máy tính cá nhân (laptop, desktop computer) gồm các loại: CRT, LCD, hay Plasma
 - *Print*: máy in
 - *Handheld*: các thiết bị cầm tay
 - *All*: các kiểu sẽ được sử dụng mà không xem xét kiểu thiết bị xuất
- Nếu kiểu thiết bị xuất không tương ứng với kiểu thiết bị đã chỉ định trong thuộc tính *media* thì các kiểu này sẽ bị bỏ qua

Screen



Print



Handheld

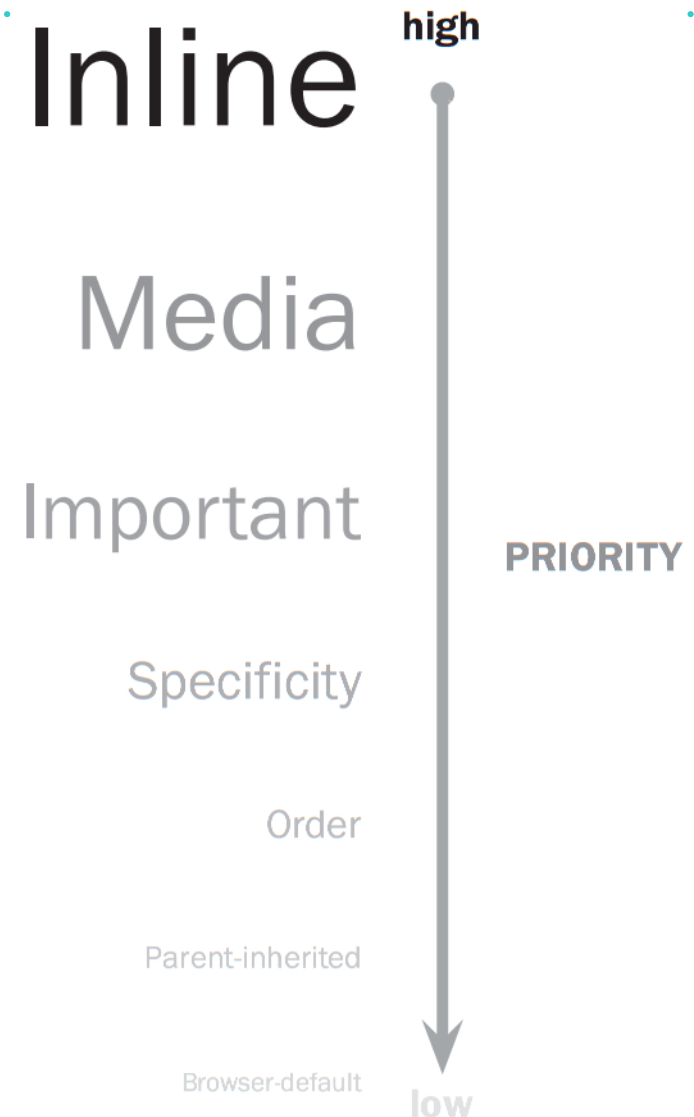


Độ ưu tiên áp dụng các kiểu CSS (Cascade)



- Việc áp dụng các kiểu CSS theo nguyên tắc **ưu tiên theo thứ tự** các luật sau:

1. Inline styles
2. Media
3. Importance
4. Specificity
5. Order
6. Parent-inherited
7. Browser-default



CSS PROPERTIES



1. Font
2. Text
3. Background
4. Box model
 - a. **Display & Visibility**
 - b. **Float**
 - c. **Width and Height**
 - d. **Border**
 - e. **Padding**
 - f. **Margin**
 - g. **Outline**
5. Position
6. Table
7. Lists
8. Cursor

1. Fonts



- Các thuộc tính CSS font định nghĩa: kiểu chữ, cỡ chữ, in đậm, in nghiêng,...
- Trên màn hình máy tính, các font Sans-serif được xem là dễ đọc hơn font Serif



- *font*: là một *shorthand property* (cho phép liệt kê các giá trị cùng lúc)
 - VALUES: `inherit` | `<font-style>` `<font-variant>` `<font-weight>` `<font-size>/<line-height>` `<font-family>`

1. Fonts (tt)



■ Font Style

- *font-style*: dùng để định dạng văn bản in nghiêng (italic) hay xiên (oblique)
- VALUES : **normal** | **italic** | **oblique** | **inherit**
- italic và oblique rất giống nhau, nhưng oblique ít được hỗ trợ hơn

Ví dụ: **font-style: italic;**

■ Font Weight

- *font-weight*: xác định độ đậm của font chữ
- VALUES: **normal** | **bold** | **bolder** | **lighter** | **inherit**

Ví dụ: **font-weight: bold;**

1. Fonts (tt)



■ Font Size

- *font-size*: xác định cỡ chữ của văn bản
- VALUE: <length> | <percentage-parents-font-size> | smaller | larger | xx-small | x-small | small | medium | large | x-large | xx-large | inherit

Ví dụ: `font-size: 12px;`

1. Fonts (tt)



■ Font Families

- *font-family*: xác định họ font thông qua các tên (font family name) như: "Serif" hay "Times New Roman"
- Có 2 loại font family name:
 - *Generic family*: dùng để chỉ một nhóm các họ font gần giống nhau như: "Serif" hay "Monospace"
 - *Font family*: xác định một họ font đặt biệt như: "Times New Roman" hay "Arial"

| Generic family | Font family | Description |
|----------------|-------------------------------|--|
| Serif | Times New Roman Georgia | Serif fonts have small lines at the ends on some characters |
| Sans-serif | Arial Verdana | "Sans" means without - these fonts do not have the lines at the ends of characters |
| Monospace | Courier New Lucida Console | All monospace characters have the same width |

1. Fonts (tt)



■ Font Families (tt)

- Thuộc tính *font-family* có thể quản lý nhiều tên font như hệ thống "fallback".
- Nếu browser không hỗ trợ font thứ nhất, nó thử font kế
- Bắt đầu với font bạn muốn, và kết thúc với font họ chung, để browsers chọn một font tương tự trong họ chung nếu không có font nào khác sẵn có
- Nếu tên của họ font nhiều hơn một từ, nó phải đặt trong dấu nháy kép: " "
- Nhiều hơn một họ font được xác định trong danh sách được ngăn cách bởi dấu phẩy ","

Ví dụ:

font-family: "Times New Roman", Times, serif;

2. Text



- Các thuộc tính text cho phép định dạng một đoạn văn bản, bao gồm: *color*, *letter-spacing*, *word-spacing*, *line-height*, *white-space*, *text-align*, *vertical-align*, *text-indent*, *text-decoration*, *text-transform*, *text-shadow*
- *color*: màu của văn bản
 - VALUES: *<color>* | *inherit*
 - Ví dụ: *color: red;*
- *line-height*: khoảng cách giữa các dòng trong một đoạn văn bản
 - VALUES: *normal* | *<number>* | *<length>* | *<percentage-font-size>%* | *inherit*
 - Ví dụ: *line-height: 1.5;*

3. Text (tt)



- *text-align*: canh lề cho văn bản
 - VALUES: left | center | right | justify | inherit

Ví dụ: *text-align: center;*
- *text-decoration*: tạo đường kẻ gạch qua văn bản (line-through, underline, overline) hay tạo văn bản nhấp nháy (blink)
 - VALUES: none | line-through | underline | overline | blink | inherit

Ví dụ: *text-decoration: none;*

2. Text (tt)



- *text-transform*: tạo văn bản in hoa (**uppercase**), thường (**lowercase**), hay viết hoa các ký tự đầu tiên (**capitalize**)
 - VALUES: **lowercase** | **capitalize** | **uppercase** | **none** | **inherit**

Ví dụ: **text-transform: uppercase;**

- *text-shadow* : tạo hiệu ứng đổ bóng cho văn bản
 - VALUES: **h-shadow** **v-shadow** **blur-radius** **color** | **none** | **initial** | **inherit**;

Ví dụ: **text-shadow: 2px 2px 8px #FF0000;**

Text-shadow with blur effect

3. Background



- Hầu như tất cả các phần tử HTML đều có thuộc tính nền
=> dùng CSS để tùy chỉnh màu nền, ảnh nền
- **background** (shorthand property): cho phép thiết lập tất cả các thuộc tính nền của một phần tử
 - VALUES: `<background-color>` `<background-image>`
`<background-repeat>` `<background-attachment>` `<background-position>` | none

background: red url(bg-01.png) repeat scroll top 0;

```
graph TD
    color --- red
    image --- url["url(bg-01.png)"]
    repeat --- repeat
    attachment --- scroll
    position --- top
    red --- url
    url --- repeat
    repeat --- scroll
    scroll --- top
```


3. Background (tt)



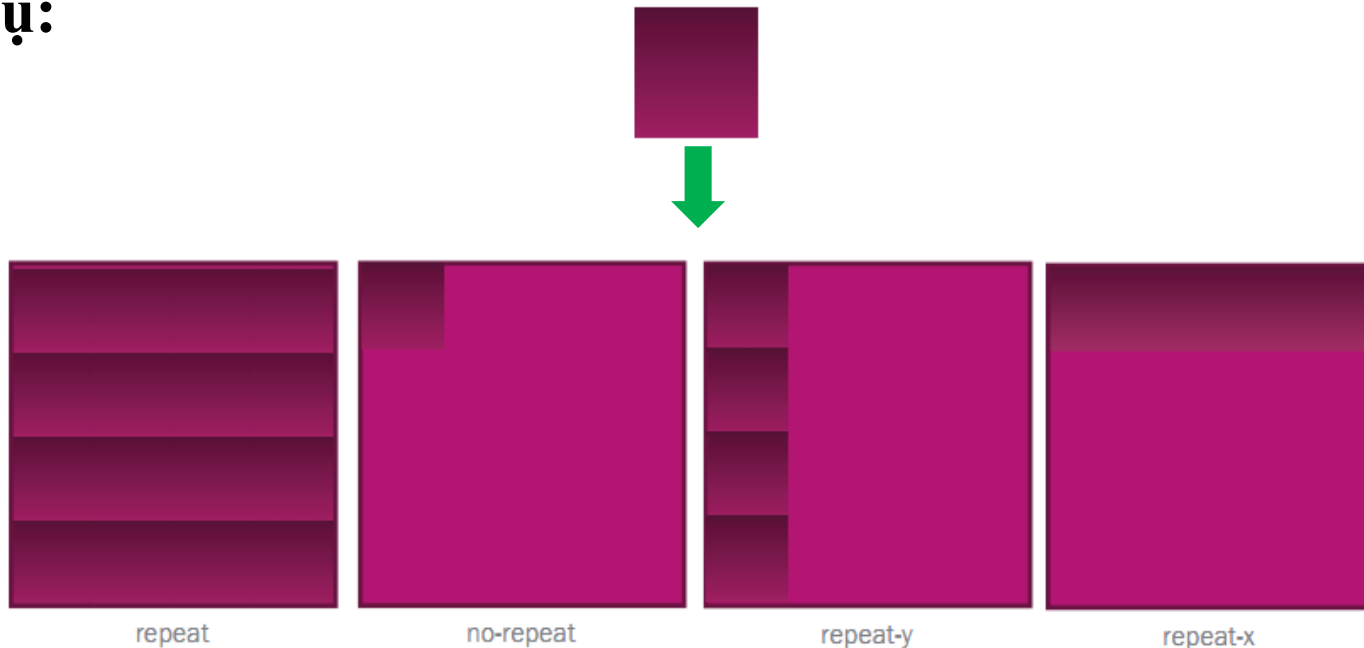
- **background-color**: màu nền của một phần tử, nếu ảnh nền không phủ lấp hết phần tử thì phần còn lại sẽ được phủ lấp bằng màu nền
 - VALUES: **<color>** | **transparent** | **inherit**
- **background-image**: ảnh nền (png, jpeg, gif)
 - VALUES: **url(<url>)** | **none**
- **background-attachment**: thiết lập ảnh nền có được cuộn (**scroll**) theo nội dung hay cố định (**fixed**)
 - VALUES: **scroll** | **fixed** | **inherit**

3. Background (tt)



- **background-repeat**: thiết lập ảnh nền được lặp lại, lặp theo chiều ngang, chiều đứng hay không lặp. Mặc định ảnh nền sẽ được lặp lại để có thể phủ toàn bộ phần tử
 - VALUES: **repeat** | **repeat-x** | **repeat-y** | **no-repeat** | **inherit**

Ví dụ:



3. Background (tt)



- **background-position**: thiết lập vị trí ảnh nền. Có thể thiết lập cùng lúc vị trí ảnh nền ở cả 2 chiều ngang (**left**) và dọc (**top**) bằng cách đặt 2 giá trị cách nhau 1 khoảng trắng
 - VALUES: `<length> | <percentage-box-width+padding>% | left | right | center | top | bottom | center | inherit`

Ví dụ:

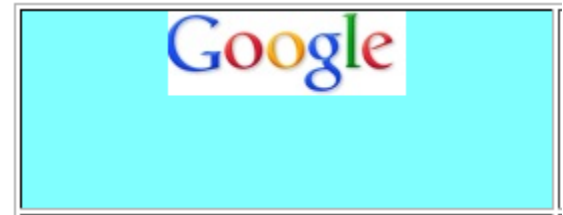
Google



25px bottom



25px 20px



top

4. Box model



- Tất cả phần tử HTML có thể được xem như các hộp (boxes). Trong CSS, thuật ngữ “mô hình hộp” (box model) được dùng khi nói về thiết kế và sắp xếp (layout).
- Mô hình hộp CSS là 1 hộp bao bọc các phần tử HTML, và chứa: lề (margins), đường viền (borders), vùng đệm (padding) và nội dung (content).
- Mô hình hộp cho phép thiết lập các đường viền bao quanh các phần tử và không gian của các phần tử trong mối tương quan với các phần tử khác.

4. Box model (tt)



- **Lề (Margin):** chiếm một vùng quanh đường viền. Lề không có màu nền và nó hoàn toàn trong suốt.
- **Đường viền (Border):** Đường viền nằm giữa padding và nội dung. Đường viền bị ảnh hưởng bởi màu nền của hộp.



- **Padding:** chiếm một vùng quanh nội dung. Padding bị ảnh hưởng bởi màu nền của hộp
- **Nội dung (Content):** nội dung của hộp, nơi văn bản và hình ảnh xuất hiện

Width & height



- **width/ height** dùng để thiết lập chiều rộng/ chiều cao cho khu vực **nội dung** (content) trong mô hình hộp
 - Lưu ý 1: không tác dụng với thành phần kiểu inline
 - Lưu ý 2: không phải là chiều rộng/ cao của nguyên cái hộp
- **width**: độ rộng
 - VALUES: `<length>` | `<percentage-parent-width>` | `auto` | `inherit`
- **height**: chiều cao
 - VALUES: `<length>` | `<percentage-parent-height>` | `auto` | `inherit`
- **overflow**: thiết lập cách trình bày khi nội dung vượt quá sức chứa của một phần tử
 - VALUES: `hidden` | `visible` | `scroll` | `auto` | `inherit`

Width & height (tt)



Ví dụ:

width: 225px;

height: 150px;

overflow: hidden;

Width = 225px

Height = 150px

One thing was certain, that the white kitten had had nothing to do with it: – it was the black kitten's fault entirely. For the white kitten had been having its face washed by the old cat for the last quarter of an hour (and bearing it pretty well, considering); so you see that it couldn't have had any hand in the mischief.



The White Kitten

The way Dinah washed her children's faces was this: first she held the poor thing down by its ear with one paw, and then with the other paw she rubbed its face all over, the wrong way, beginning at the nose: and just now, as I said, she was hard at work on the white kitten, which was lying quite still and trying to purr – no doubt feeling that it was all meant for its good.

Overflow

Border



- **border** (shorthand property): thiết lập giá trị cho tất cả các thuộc tính của đường viền. Thứ tự các thuộc tính cần được đảm bảo như sau:
 - **border:** `<border-width> <border-style> <border-color>`
- **border-width:** độ rộng đường viền
 - VALUES: `<length> | thin | medium | thick | inherit`
- **border-style:** kiểu của đường viền
 - VALUES: `none | dotted | dashed | solid | double | groove | ridge | inset | outset | inherit`
- **border-color:** màu sắc đường viền
 - VALUES: `<color> | transparent | inherit`

Border (tt)



- Có thể điều chỉnh đường viền cho **từng phía** khác nhau
 - `border-top`, `border-right`, `border-bottom`, `border-left` (shorthand properties): thiết lập giá trị cho tất cả các thuộc tính của đường viền theo từng phía
 - `border-width-top`, `border-width-right`, `border-width-bottom`, `border-width-left`: độ rộng từng phía
 - `border-style-top`, `border-style-right`, `border-style-bottom`, `border-style-left`: kiểu đường viền từng phía
 - `border-color-top`, `border-color-right`, `border-color-bottom`, `border-color-left`: màu sắc từng phía
- `border-radius` (shorthand properties): bo tròn các góc
 - `border-radius: 10px 20px 30px 40px;`

Border (tt)



Ví dụ: `border: 5px solid rgb(67,0,37);`

Top = 5px solid rgb(67,0,37)

Left = 5px solid rgb(67,0,37)

One thing was certain, that the white kitten had had nothing to do with it: – it was the black kitten's fault entirely. For the white kitten had been having its face washed by the old cat for the last quarter of an hour (and bearing it pretty well, considering); so you see that it couldn't have had any hand in the mischief.



The White Kitten

The way Dinah washed her children's faces was this: first she held the poor thing down by its ear with one paw, and then with the other paw she rubbed its face all over, the wrong way, beginning at the nose: and just now, as I said, she was hard at work on the white kitten, which was lying quite still and trying to purr -- no doubt feeling that it was all meant for its good.

Right = 5px solid rgb(67,0,37)

Bottom = 5px solid rgb(67,0,37)

Padding



- Padding (vùng đệm) là khoảng không giữa nội dung (content) và đường viền (border) của một phần tử
- Có thể điều chỉnh độ rộng vùng đệm cho từng phía khác nhau
- **padding** (shorthand property): thiết lập độ rộng vùng đệm cho tất cả các phía
 - VALUES: **<length>** | **<percentage-box-width>%** | **inherit**
 - Thuộc tính padding có thể có từ 1 đến 4 giá trị
- **padding-top**, **padding-right**, **padding-bottom**, **padding-left**: thiết lập độ rộng vùng đệm cho từng phía
 - VALUES: **<length>** | **<percentage-box-width>%** | **inherit**

Padding (tt)




padding: 35px 30px;

Top = 35px

Left = 30px

One thing was certain, that the white kitten had had nothing to do with it: – it was the black kitten's fault entirely. For the white kitten had been having its face washed by the old cat for the last quarter of an hour (and bearing it pretty well, considering); so you see that it couldn't have had any hand in the mischief.



The White Kitten

Right = 30px

Bottom = 35px

The way Dinah washed her children's faces was this: first she held the poor thing down by its ear with one paw, and then with the other paw she rubbed its face all over, the wrong way, beginning at the nose: and just now, as I said, she was hard at work on the white kitten, which was lying quite still and trying to purr – no doubt feeling that it was all meant for its good.

Margin

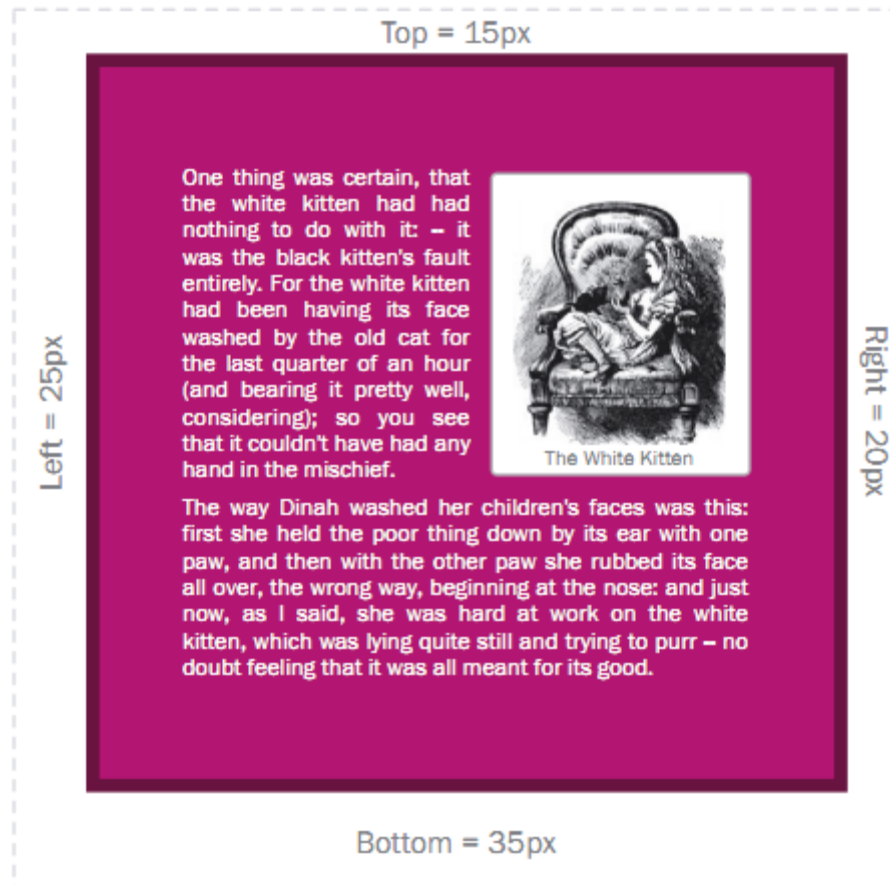


- Margin (lề) là khoảng không giữa đường viền của một phần tử với đường viền của các phần tử lân cận
- Có thể điều chỉnh độ rộng đường viền cho từng phía khác nhau
- **margin** (shorthand property): độ rộng lề cho tất cả các phía
 - VALUES: `<length>` | `<percentage-box-width>%` | `inherit`
 - Thuộc tính margin có thể có từ 1 đến 4 giá trị
- **margin-top, margin-right, margin-bottom, margin-left**: độ rộng lề cho từng phía
 - VALUES: `<length>` | `<percentage-box-width>%` | `inherit`

Margin (tt)



margin: 15px 20px 35px 25px;



Display

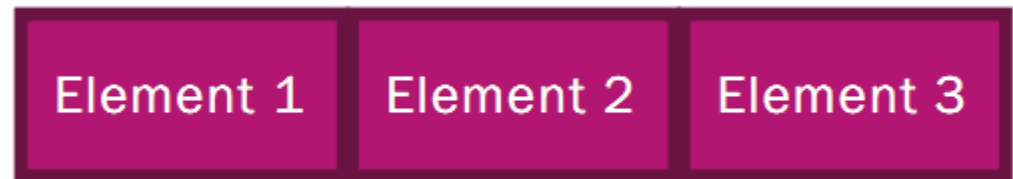
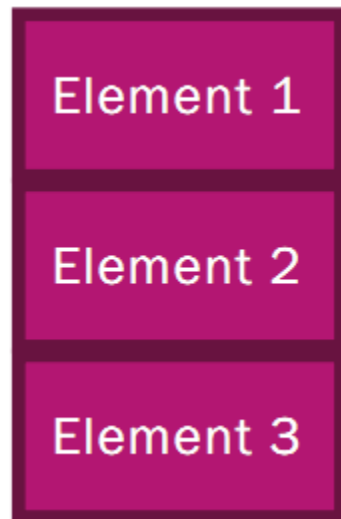


- Tất cả các phần tử HTML sẽ được hiển thị theo một kiểu mặc định => dùng thuộc tính **display** để thiết lập cách hiển thị của một phần tử
- **display**: thiết lập phần tử HTML được hiển thị theo dạng inline, block, danh sách hay không hiển thị trên trình duyệt
 - VALUES: **inline** | **block** | **list-item** | **none** | **inherit**
 - Với giá trị **none**: phần tử HTML sẽ hoàn toàn bị gỡ bỏ khỏi trang web

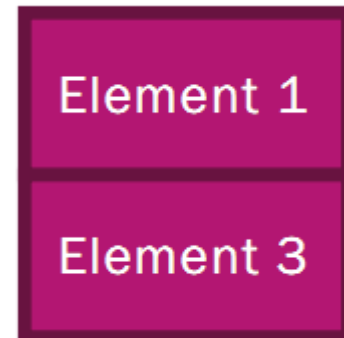
Display (tt)



Ví dụ: `Element 1`
`Element 2`
`Element 3`



`#e1,#e2,#e3 {display:inline;}`



`#e1,#e2,#e3 {display:block;}`

`#e1,#e3 {display:block;}`
`#e2 {display:none;}`

Visibility



- **visibility**: hiển thị hay không hiển thị một phần tử html
 - VALUES: **visible** | **hidden** | **inherit**
 - Với giá trị là **hidden** phần tử sẽ không được trình duyệt hiển thị, tuy nhiên nó vẫn tồn tại trên trang web, và chiếm lấy khoảng không như lúc nó được hiển thị
- **opacity**: độ mờ đục của một phần tử từ 0.0 (trong suốt) đến 1.0 (hiển thị hoàn toàn)
 - VALUES: **<0.0-1.0>** | **inherit**
- **filter: alpha (opacity=<0-100>)**: thiết lập độ mờ đục của một phần tử từ 0 (trong suốt) đến 100 (hiển thị hoàn toàn). Đây không là thuộc tính trong chuẩn css, dùng thay thế cho thuộc tính opacity không được hỗ trợ trong IE 8 và các phiên bản trở về trước => đặt giá trị cho cả **opacity** & **filter** để có được kết quả giống nhau trên tất cả các trình duyệt.

Visibility (tt)



Ví dụ:

```
<span id="e1">Element 1</span>  
<span id="e2">Element 2</span>  
<span id="e3">Element 3</span>
```

Element 1

Element 3

Element 1
Element 2
Element 3

```
#e1 {opacity:1;  
      filter:alpha(100);}
```

```
#e2 {opacity:.5;  
      filter:alpha(50);}
```

```
#e3 {opacity:.25;  
      filter:alpha(25);}
```

```
#e1,#e3 {display:block;}  
#e2 {visibility:hidden;}
```

Float



- **float**: dùng để thiết lập vị trí của một phần tử bên trái, hay phải so với thành phần cha của nó. Tất cả các phần tử bên dưới nó sẽ "nổi" lên ngay bên cạnh nó chiếm lấy khoảng không gian còn trống.
 - VALUES: **left** | **right** | **none** | **inherit**
- **clear**: dùng để ngừng lại việc "nổi" lên của một phần tử
 - VALUES: **none** | **left** | **right** | **both** | **inherit**

Ví dụ:

```
<p id="e1">Element 1</p>
```

```
<p id="e2">One thing was...</p>
```

```
<p id="e3">The way Dinah...</p>
```

Float (tt)



```
#e1 { width: 75px;
      height: 150px;
      float: right; }
#e3 { clear: right; }
```

One thing was certain, that the white kitten had had nothing to do with it: -- it was the black kitten's fault entirely. For the white kitten had been having its face washed by the old cat for the last quarter of an hour (and bearing it pretty well, considering); so you see that it couldn't have had any hand in the mischief.

Element 1

The way Dinah washed her children's faces was this: first she held the poor thing down by its ear with one paw, and...

```
#e1 { width: 75px;
      height: 150px;
      float: right; }
```

One thing was certain, that the white kitten had had nothing to do with it: -- it was the black kitten's fault entirely. For the white kitten had been having its face washed by the old cat for the last quarter of an hour (and bearing it pretty well, considering); so you see that it couldn't have had any hand in the mischief.

The way Dinah washed her children's faces was this: first she held the poor thing down by its ear with one paw, and...

Element 1

5. Position



- **position**: xác định cách định vị cho một phần tử
 - VALUES: **static** | **relative** | **absolute** | **fixed** | **inherit**
 - **static**: phần tử được định vị mặc nhiên, không thể thay đổi vị trí của phần tử (không bị ảnh hưởng bởi các thuộc tính **top**, **bottom**, **left**, **right**)
 - **relative**: vị trí của phần tử được tính tương đối so với vị trí mặc nhiên của nó, phần tử vẫn sẽ chiếm giữa khoảng không gian mà nó định vị như trước khi được di chuyển với các thuộc tính: **top**, **bottom**, **left**, **right**
 - **absolute**: vị trí của phần tử được tính tương đối so với phần tử cha đầu tiên của nó có giá trị của thuộc tính position khác static. Nếu không tồn tại phần tử như thế thì vị trí tương đối được tính so với phần tử `<html>`
 - **fixed**: vị trí của phần tử được tính dựa vào cửa sổ của trình duyệt, phần tử sẽ được cố định tại một vị trí khi màn hình được cuộn
- Một phần tử có thuộc tính position có giá trị: **relative**, **absolute**, **fixed** sẽ có khả năng phủ lấp lên một phần tử khác

5. Position (tt)



- Thuộc tính position không làm thay đổi vị trí của một phần tử, nó chỉ là bước chuẩn bị để ta có thể định vị một phần tử thích hợp bằng các thuộc tính sau:
 - **top**: di chuyển phần tử một khoảng cách tính từ lề phía trên
 - VALUES: **auto** | **<length>** | **<percentage-parents-height>%** | **inherit**
 - **right**: di chuyển phần tử một khoảng cách tính từ lề phía bên phải
 - VALUES: **auto** | **<length>** | **<percentage-parents-width> %** | **inherit**
 - **bottom**: di chuyển phần tử một khoảng cách tính từ lề phía dưới
 - VALUES: **auto** | **<length>** | **<percentage-parents-height>%** | **inherit**
 - **left**: di chuyển phần tử một khoảng cách tính từ lề phía bên trái
 - VALUES: **auto** | **<length>** | **<percentage-parents-width>%** | **inherit**

5. Position (tt)



Ví dụ:

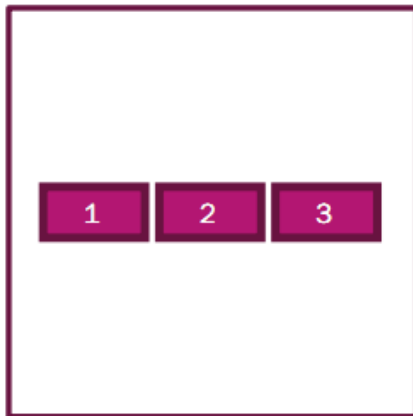
```
<span id="e1">1</span>
```

```
<span id="e2">2</span>
```

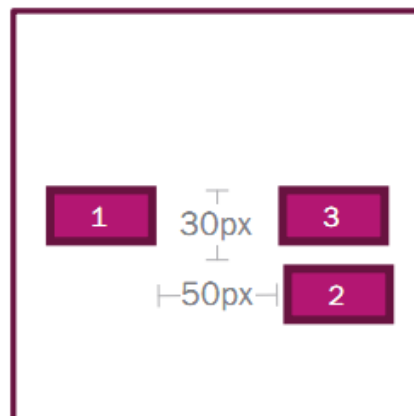
```
<span id="e3">3</span>
```

```
#e2 { top:30px;
```

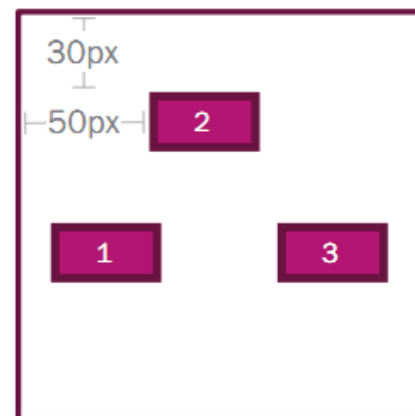
```
left:50px; }
```



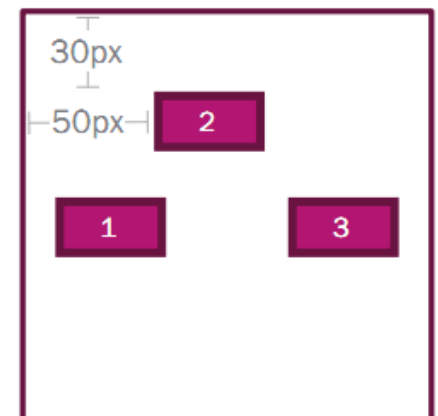
static



relative



absolute



fixed

5. Position (tt)

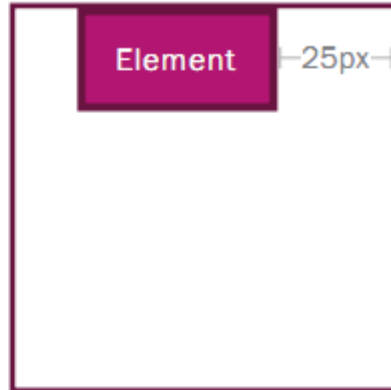


Ví dụ:

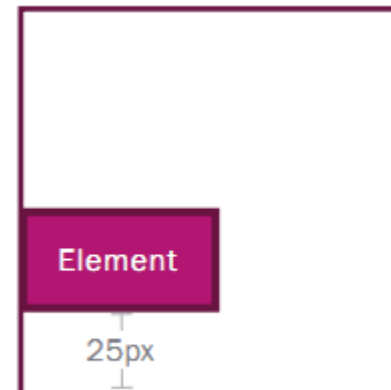
```
<span id="e2">Element</span>  
#e2 { position: absolute; }
```



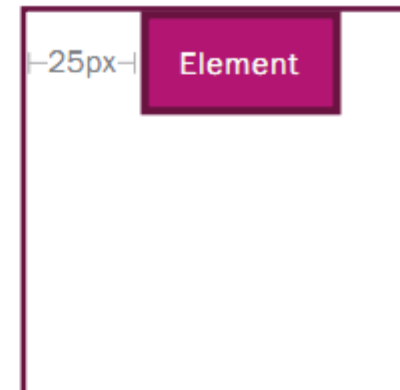
top: 25px



right: 25px



bottom: 25px



left: 25px

6. Tables



- **border-collapse**: xác định đường viền giữa các ô kế nhau trong bảng có thể chia sẻ với nhau (collapse), hay không (separate)
 - VALUES: **collapse** | **separate** | **inherit**
 - Dùng cho thẻ table
- **caption-side**: vị trí tiêu đề của bảng
 - VALUES: **top** | **bottom** | **inherit**
- Định màu nền dòng chẵn khác dòng lẻ
 - `tr:nth-child(even){background-color: #f2f2f2;}`
`/* dòng chẵn */`
 - `tr:nth-child(odd) {background-color: #ffffff;}`
`/* dòng lẻ */`

6. Tables (tt)



- Kẻ đường viền: `border`
- Kích thước bảng hoặc ô: `width, height`
- Canh chữ trong ô: `text-align, vertical-align`
- Khoảng cách giữa chữ và đường viền: `padding`
- Màu nền: `background-color`
- Sự kiện rê chuột lên: `:hover`

7. Lists



- Dùng để tạo các kiểu đánh dấu cho các phần tử danh sách (``, ``)
- `list-style` (shorthand property): giá trị cho tất cả các thuộc tính của danh sách
 - VALUES: `inherit` | `<list-style-type>` `<list-style-image>` `<list-style-position>`
- `list-style-type`: kiểu đánh dấu đầu dòng cho các phần tử trong danh sách
 - VALUES: `disc` | `circle` | `square` | `decimal` | `decimal-leading-zero` | `upper-roman` | `lower-roman` | `upper-alpha` | `lower-alpha` | `lower-greek` | `none` | `inherit`
- `list-style-image`: tạo kiểu đánh dấu đầu dòng cho các phần tử trong danh sách là một ảnh
 - VALUES: `url(<url>)` | `none` | `inherit`
- `list-style-position`: xác định lề, và vị trí đặt các ký tự đánh dấu đầu dòng của các phần tử trong danh sách
 - VALUES: `inside` | `outside` | `inherit`

7. Lists (tt)



list-style-type: square;

| | | | | | |
|---|--------|---|-------------|----|----------------------|
| • | Disc | A | Upper-Alpha | 1 | Decimal |
| ○ | Circle | a | Lower-Roman | 01 | Decimal-Leading-Zero |
| ■ | Square | α | Lower-Greek | I | Upper-Roman |
| | | | | i | Lower-Roman |

list-style-position: inside;

- One thing was certain,
that the white kitten
- had had nothing to do
with it
- it was the black
kitten's fault entirely

Inside

- One thing was certain,
that the white kitten
- had had nothing to do
with it
- it was the black
kitten's fault entirely

Outside

8. Cursor



- **cursor**: thiết lập hình dạng cho con trỏ

- VALUES: default |
 url(<url>) | auto |
 crosshair | pointer | move |
 e-resize | ne-resize | nw-
 resize | n-resize | se-resize |
 sw-resize | s-resize | w-
 resize | text | wait | help |
 progress | inherit



crosshair



pointer



move



text



wait



help



progress



n-resize



e-resize



s-resize



w-resize



ne-resize



nw-resize



se-resize