



CT428. LẬP TRÌNH WEB

CHƯƠNG 4 - CSS VÀ JAVASCRIPT

Giảng viên: Trần Công Ân (tcan@cit.ctu.edu.vn)

Bộ môn Mạng máy tính & Truyền thông
Khoa Công Nghệ Thông Tin & Truyền Thông
Đại học Cần Thơ

2015

Cascading Style Sheet (CSS)

NỘI DUNG

GIỚI THIỆU CASCADING STYLE SHEET (CSS)

CÁC CÁCH SỬ DỤNG CSS TRONG HTML

SELECTOR (BỘ CHỌN)

PROPERTY

BỐ CỤC (LAYOUT)

BOX MODEL

MỘT SỐ CHÚ Ý

CASCADING STYLE SHEET (CSS) LÀ GÌ?

- ▶ Là một ngôn ngữ dùng để mô tả hình thức, bố cục và cách trình bày của thông tin trên trang web.
(vs. HTML: dùng để thể hiện nội dung và cấu trúc trang web)
- ▶ Dùng để mô tả thông tin sẽ được thể hiện thế nào.
(vs. HTML: dùng để định nghĩa thông tin gì sẽ được thể hiện)
- ▶ Cho phép tách rời phần trình bày và phần nội dung của một trang web, giúp việc bảo trì, thay đổi hay cập nhật dễ dàng hơn.
- ▶ CSS có thể được nhúng bên trong trang web hay định nghĩa trong 1 tập tin riêng biệt (.css) để sử dụng chung cho nhiều trang web.

(HTML: Structure) + (CSS: Presentation) = Web page

LỢI ÍCH VÀ NHƯỢC ĐIỂM CỦA CSS

- ▶ Cấu trúc của các trang web **rõ ràng hơn** do tách rời giữa trình bày và nội dung.
- ▶ Có thể thừa kế style đã định nghĩa cho nhiều trang web hoặc website \Rightarrow **tiết kiệm thời gian thiết kế** website.
- ▶ Giảm kích thước trang web \Rightarrow **tăng tốc độ download** trang web.
- ▶ **Bảo trì và thay đổi cấu trúc**, bố cục và hình thức của trang web dễ dàng hơn.
- ▶ Tuy nhiên, CSS có một nhược điểm là **sự tương thích của trình duyệt** đối với CSS: các trình duyệt có sự hỗ trợ khác nhau đối với CSS.

CẤU TRÚC CỦA MỘT CSS

- ▶ Một CSS chứa một **tập các luật CSS** (CSS rules).
- ▶ Một luật CSS bao gồm một (hoặc vài) **bộ chọn** (selector) và 1 tập các **khai báo về các thuộc tính** (property) áp dụng cho bộ chọn.

```
selector {  
  property: value;  
  property: value;  
  ...  
}
```

Ví dụ:

```
body {  
  color: yellow;  
  background-color: black;  
}
```

- ▶ selector: chọn **phần tử** (element) HTML sẽ được áp dụng style.
- ▶ property: **thuộc tính** của thành phần HTML sẽ được áp dụng style.
- ▶ value: **giá trị** được gán cho thuộc tính tương ứng.

NHỮNG CSS TRONG TẬP TIN HTML (INTERNAL)

- Cách thông dụng nhất là đặt các CSS rules trong cặp thẻ `<style>` và đặt trong phần **head** của tập tin HTML (internal style sheet).

```
<html>
  <head>
    <style type="text/css">
      body {
        color: yellow;
        background-color: black;
      }
    </style>
  </head>
  <body> ... </body>
</html>
```

- **Inline:** `<p style="...">Paragraph</p>` (không khuyến khích).

LIÊN KẾT TẬP TIN CSS VÀO TẬP TIN HTML

- ▶ Các CSS rules được định nghĩa trong một tập tin với phần mở rộng **.css** (external style sheet).
- ▶ Dùng thẻ **<link>** để liên kết tập tin CSS vào tập tin HTML.

```
<link rel="stylesheet" type="text/css" href="filename"/>
```

- ▶ Thẻ **<link>** được đặt trong phần **head** của tập tin HTML.

```
<html>  
  <head>  
    <link rel="stylesheet" type="text/css" href="mystyle.css"/>  
  </head>  
  <body> ... </body>  
</html>
```


THỨ TỰ ƯU TIÊN TRONG VIỆC ÁP DỤNG CSS

- ▶ Nếu trong tập tin HTML có sử dụng CSS theo nhiều cách, trong đó có sự **động độ về định dạng**, thì độ ưu tiên được áp dụng như sau:
 1. Inline CSS
 2. Internal CSS
 3. External CSS
 4. Định dạng mặc định của trình duyệt

CÁC LOẠI SELECTOR TRONG CSS

1. HTML-element selector (hay gọi tắt là element selector)
2. Class selector
3. ID selector
4. Attribute selector
5. Các selector đặc biệt như: descendant (con cháu/hậu duệ), pseudo-class, group, ...

```
selector {  
  property: value;  
  ...  
}
```

HTML-ELEMENT SELECTOR

- Chọn **tất cả** các phần tử (thẻ) HTML trong trang web được chỉ định bởi selector của CSS rule. **Cú pháp:** `HTML-tag {property: value; ...}`

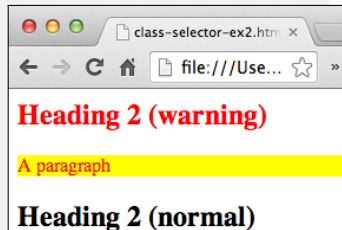
```
<html>
  <head>
    <style type="text/css">
      p {color: red; background-color: yellow} /*rule for <p>*/
      h1 {color: blue} /*rule for <h1>*/
    </style>
  </head>
  <body>
    <h1>Heading 1 (redefined)</h1>
    <p>This is a paragraph</p>
    <h2>Heading 2 (default)</h2>
  </body>
</html>
```



CLASS SELECTOR

- Chọn tất cả các phần tử trong trang web **thuộc class** được chỉ định bởi selector. **Cú pháp:** `.classname {property: value;...}`

```
<html>
<head>
  <style type="text/css">
    .wrn {color: red;} /*rule for class wrn*/
    .hlight {background-color: yellow;} /*rule for class hlight*/
  </style>
</head>
<body>
  <h2 class="wrn">Heading 2 (warning)</h2>
  <p class="wrn hlight">A paragraph</p>
  <h2>Heading 2 (normal)</h2>
</body>
</html>
```

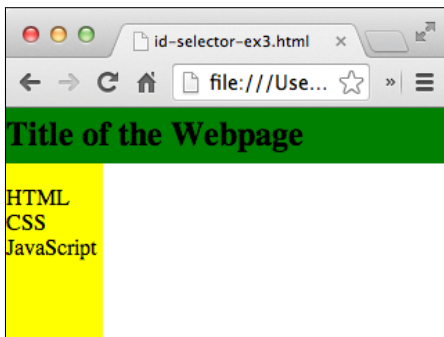


ID SELECTOR

- Chọn 1 phần tử duy nhất trong trang web có id chỉ định bởi selector.
Cú pháp: `#id {property: value;...}`

```
<head>
  <style type="text/css">
    #header {background-color:green; width:100%; height:40px;
      position:absolute; left:0;
      top:0;padding:0px;margin:0px;margin:auto}
    #menu {background-color:yellow; height:100%; width:70px;
      position:absolute; left:0; top:40;float:left;}
    h2 {margin:0px}
  </style>
</head>
<body style="margin:0px">
  <div id="header"> <h2>Title of the Webpage</h2>
</div>
  <div id="menu"> <p>HTML<br>CSS<br>JavaScript</p> </div>
```

ID SELECTOR



ATTRIBUTE SELECTOR

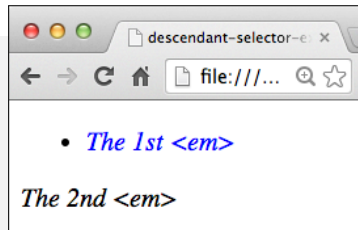
- ▶ Dùng để lựa chọn các phần tử HTML **có thuộc tính** được chỉ định bởi selector. **Một số cú pháp:**
 - ▶ `[attribute]`: chọn tất cả các phần tử có thuộc tính attribute.
 - ▶ `[attribute="avalue"]`: chọn tất cả các phần tử có thuộc tính attribute **có giá trị** là avalue.
 - ▶ `[attribute~="avalue"]`: chọn tất cả các phần tử có thuộc tính attribute với giá trị **có chứa từ** avalue.
 - ▶ `[attribute*="avalue"]`: giá trị thuộc tính **có chứa** avalue.
- ▶ Có thể k/hợp HTML-element selector với attr selector như sau:
 - ▶ `HTML-tag[attribute-selector]`: chọn tất cả các phần tử là HTML-tag có thuộc tính được chỉ định bởi attribute-selector.

VÍ DỤ

DESCENDENT SELECTOR

- ▶ **Cú pháp:** dùng khoảng trắng `ances-selector desc-selector {...}`
- ▶ **Hoạt động:** chọn tất cả các thành phần `desc-selector` là “con cháu” (descendant) của `ances-selector`.

```
<head>
  <style type="text/css">
    ul em {color:blue;}
  </style>
</head>
<body>
  <ul>
    <li><em>The 1st &lt;em>&gt;</em></li></ul>
    <p><em>The 2nd &lt;em>&gt;</em></p>
  </body>
</html>
```

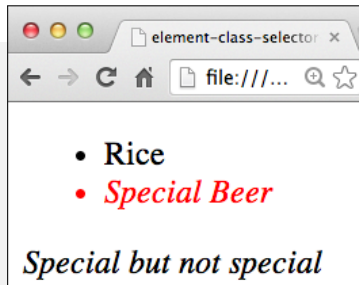


Chú ý: dùng ký hiệu `>` để chọn con trực tiếp.

HTML-ELEMENT + CLASS SELECTOR

- ▶ **Cú pháp:** `HTML-tag.className {...}`
- ▶ **Hoạt động:** chọn tất cả các thành phần **HTML-tag** có thuộc tính **class** có giá trị là **className**.

```
<head>
  <style type="text/css">
    li.special {color: red;}
    .special {font-style: italic;}
  </style>
</head>
<body>
  <ul>
    <li>Rice</li>
    <li class="special">Special Beer</li></ul>
  <p class="special">Special but not special</p>
</body>
```



KẾT HỢP NHIỀU LOẠI SELECTOR

- ▶ Các loại selector trên có thể lồng nhau nhiều cấp.
- ▶ Ví dụ:

```
div#header p em.required {  
    color: white;  
}
```

⇔ Chọn các phần tử `` có thuộc tính `class` có giá trị chứa từ `required` (thuộc lớp `required`) và là con/cháu của các phần tử `<p>`; với các phần tử `<p>` phải là con cháu của một phần tử `<div>` có `id` là `header`.

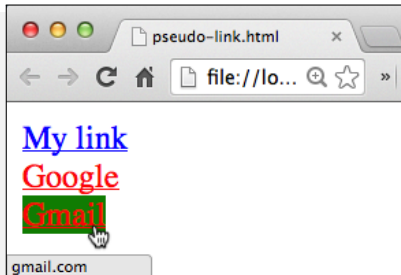
PSEUDO-CLASSES VÀ PSEUDO-ELEMENTS

- ▶ **Pseudo-classes** (lớp giả) và **pseudo-element** (thành phần giả) cho phép truy cập một số t/phần đặc biệt của các phần tử HTML, nằm ngoài cấu trúc cây của trang web (document tree).
- ▶ Các pseudo-classes/elements bắt đầu bằng dấu :
- ▶ Cú pháp: **selector:pseudo-class** hoặc **selector:pseudo-element**

Speudo	Ý nghĩa
:link	Chọn tất cả các l/kết chưa viếng thăm
:visited	Chọn tất cả các l/kết đã được viếng thăm
:hover	Chọn t/phần đang nằm dưới trỏ chuột
:focus	Chọn t/phần đang được focused (pointer)
:active	Chọn t/phần đang được kích hoạt
:firstchild	Chọn t/phần là con đầu tiên của một t/phần khác

PSEUDO-CLASSES/ELEMENTS – Ví Dụ 1

```
<!DOCTYPE html>
<html>
  <head>
    <style type="text/css">
      a:link {color:blue;}
      a:visited {color:red;}
      a:hover {background-color:green;}
    </style>
  </head>
  <body>
    <a href="index.html">My link</a> <br>
    <a href="http://www.google.com">Google</a><br>
    <a href="http://gmail.com">Gmail</a>
  </body>
</html>
```

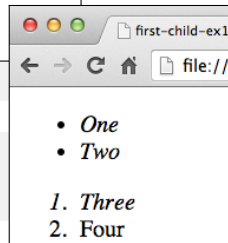
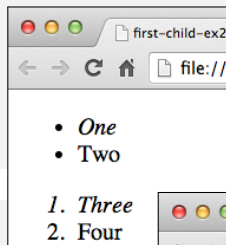


PSEUDO-CLASSES/ELEMENTS – Ví Dụ 2

```
<body>
  <ul><li>One</li>
    <li>Two</li></ul>
  <ol><li>Three</li>
    <li>Four</li></ol>
</body>
```

```
<style type="text/css">
  li:first-child {font-style: italic;}
</style>
```

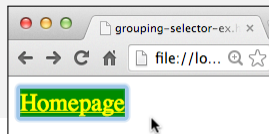
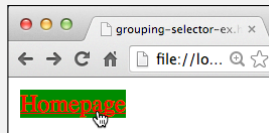
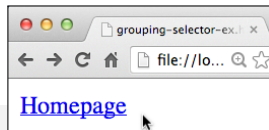
```
<style type="text/css">
  :first-child {font-style: italic;}
</style>
```



NHÓM CÁC SELECTOR (GROUPING SELECTOR)

- Các selector có **chung rules** có thể được nhóm lại với nhau bằng cách dùng dấu phẩy ,

```
<head>
  <style type="text/css">
    a:hover, a:focus {
      background-color: green;
    }
    a:hover {color:red;}
    a:focus {color:yellow;}
  </style>
</head>
<body>
  <a href="index.html">Homepage</a>
</body>
```



CASCADE RULES

- ▶ id selector thì càng đặc trưng hơn (more specific) class selector, class selector thì cụ thể hơn HTML-element selector.
- ▶ Thông thường, selector càng dài thì càng đặc trưng.
- ▶ Nếu các rules có cùng mức độ đặc trưng, những rules khai báo cuối cùng sẽ được áp dụng.

GÁN GIÁ TRỊ CHO THUỘC TÍNH

- ▶ Mỗi thuộc tính có thể được gán một hoặc nhiều giá trị được ngăn cách nhau bằng dấu phẩy.
- ▶ Ví dụ:

```
color: blue;  
background-color: red;  
font: italic 12px sans-serif;  
font-family: Arial, sans-serif;
```

- ▶ Một số giá trị được đ/nghĩa sẵn (được đ/nghĩa như một keyword):
 - ▶ Auto: trình duyệt tự động gán giá trị (mặc nhiên).
 - ▶ Inherit: thừa kế giá trị từ thành phần cha.
 - ▶ Normal: sử dụng giá trị mặc định của thuộc tính.

COLOR

- ▶ Thuộc tính **color** dùng để chỉ định **màu sắc của text**.
- ▶ Giá trị cho thuộc tính color (**color value**) có thể được chỉ định bằng:
 - ▶ tên (cho một số màu thông dụng): `color: red;`
 - ▶ giá trị hexa (`#rrggbb`): `color: #ff0000;`
 - ▶ hàm `rgb(r, g, b)`: `color: rgb(255, 0, 0);`
 - ▶ hàm `rgba(r, g, b, alpha)`: `color: rgba(255, 0, 0, 0.5);`
- ▶ Thuộc tính này có tính thừa kế nhưng có thể bị phủ chồng (overridden) bởi các rules đặc trưng hơn (more specific).

BACKGROUND (NỀN)

- ▶ Các thuộc tính thường sử dụng để chỉ định **hiệu ứng cho nền** (background effect) của một thành phần.
 - ▶ **background-color: <color value>**: chỉ định màu nền.
 - ▶ **background-image:url(<image url>)**: chỉ định hình nền.
 - ▶ **background-repeat: repeat | repeat-x | repeat-y | no-repeat | inherit**: thiết đặt sự lặp lại của hình nền.
 - ▶ **background-attachment: scroll | fixed**: thiết đặt hành vi của hình nền khi cuộn trang.
 - ▶ **background-position: <position value> | <%-x>% <%-y>% | <xpos ypos>**: qui định vị trí đặt hình nền.
- ▶ Shorthand: **background: blue url('img.png') no-repeat right top;**

TEXT

Các thuộc tính cho phép định dạng một đoạn văn bản:

- ▶ **color:** <color value>: chỉ định màu chữ.
- ▶ **text-align:** center | right | justify: canh lề chữ.
- ▶ **text-decoration:** none | overline | line-through | underline: chỉ định đường kẻ cho text.
- ▶ **text-transformation:** uppercase | lowercase | capitalize: chỉ định chữ hoa/thường.
- ▶ **text-indent:** <length> | <percentage-width>%: khoảng cách lùi vào trong của dòng đầu tiên.

TEXT

- ▶ **text-shadow:** <h-shadow v-shadow> [blur] [color]: chỉ định bóng của văn bản.
- ▶ **letter-spacing:** <length> | <percentage-font-size>%: khoảng cách giữa 2 ký tự.
- ▶ **word-spacing:** normal | <length>: khoảng cách giữa 2 từ.
- ▶ **line-height:** normal | <number> | <length> | <percentage>%: khoảng cách giữa 2 dòng trong đoạn văn bản.
- ▶ **vertical-align:** baseline | super | sub | top | text-top | middle | bottom | text-bottom | <length> | <percent-line-height>%: canh lề dọc.

FONT

- ▶ **font-family:** <list-of-font>: chỉ định kiểu chữ.
 - ▶ generic family: serif | sans-serif | monospace
 - ▶ font family: “Time new roman” | Georgia | Arial | Verdana | “Courier New” | “Lucida Console” | ...
- ▶ **font-style:** normal | italic: định dạng nghiêng/bình thường.
- ▶ **font-size:** <length> | <percent-parent> | smaller | larger | xx-small | x-small | small | medium | x-large | xx-large: k/thước chữ.
- ▶ **font-weight:** normal | bold | bolder | lighter: độ đậm của chữ.
- ▶ **font-variant:** normal | small-caps: hiển thị b/thường hay SMALL-CAPS.

LIST

- ▶ **list-style-type: circle | square | upper-roman | lower-roman:** chỉ định kiểu đánh dấu/số (marker) cho danh sách.
- ▶ **list-style-image: url(<image-url>):** chỉ định hình dùng làm marker cho danh sách.
- ▶ **list-style-position: outside | inside:** chỉ định marker được xem như nằm ngoài nội dung của list hay là 1 phần của nội dung.

CÁC THUỘC TÍNH ĐỊNH DẠNG TABLE

- ▶ **border-spacing**: `<length>`: khoảng cách giữa các ô.
- ▶ **border-collapse**: `collapse` | `separate`: chia sẻ border hay riêng.
- ▶ **caption-side**: `top` | `bottom`: vị trí tiêu đề.
- ▶ **empty-cell**: `show` | `hide`: hiển thị hay dấu border, nền của ô trống.
- ▶ **table-layout**: `auto` | `fixed`: cách trình bày một table.
- ▶ Các định dạng như `color`, `background-color`, `font`, ... cũng có thể được sử dụng.
- ▶ Ngoài ra, các định dạng khác như `border`, `padding`, ... sẽ được giới thiệu sau trong phần trình bày bố cục.

BỐ CỤC VÀ TRÌNH BÀY TRANG WEB (LAYOUT)

1. Block element (phần tử khối) và inline element (phần tử trong hàng)
2. Grouping element (nhóm các phần tử)
3. Display (chế độ hiển thị/trình bày)
4. Visibility (độ nhìn thấy)
5. Floating
6. Positioning (định vị trí)

PHẦN TỬ BLOCK VÀ INLINE

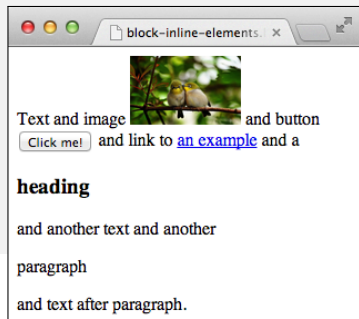
- ▶ Xét về mặt bố cục (layout), có 2 loại phần tử HTML: **block** (khối) và **inline** (trong hàng).
 - ▶ Một phần tử block luôn được **hiển thị trên một hàng mới** (new line) còn những phần tử inline thì có thể được hiển thị trên cùng hàng.
 - ▶ inline: input, a, img, br, ...
 - ▶ block: p, h, ul, li, table, ...
- ⇒ Các phần tử khối không được lồng vào nhau.

BLOCK VÀ INLINE

```

<html>
  <body>
    <p>Text and image 
      and a button <input type="button" value="Click me!">
      and a link to <a href="list.html">
        an example</a>
      and a <h3>heading</h3>
      and another text
      and another <p>paragraph</p>
      and text after paragraph.
    <p>
  </body>
</html>

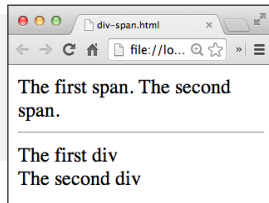
```



NHÓM CÁC PHẦN TỬ (GROUPING ELEMENTS)

- ▶ Nhóm các phần tử cho phép **định dạng nhiều phần tử cùng lúc**.
- ▶ **Phần tử thùng chứa** (container element) dùng để nhóm các phần tử:
 - ▶ thẻ ``: tạo một phần tử thùng chứa inline
 - ▶ thẻ `<div>`: tạo một phần tử thùng chứa block
- ▶ Cách hiển thị các phần tử **có thể thay đổi** bằng CSS.

```
<span>The first span.</span>  
<span>The second span.</span>  
<hr>  
<div>The first div</div>  
<div>The second div</div>
```

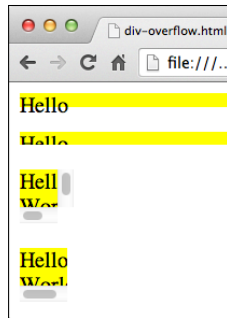


KÍCH THƯỚC CỦA PHẦN TỬ

- ▶ Thuộc tính **width** và **height** dùng để định kích thước của một phần tử.
- ▶ Các thuộc tính **overflow**, **overflow-x** và **overflow-y** dùng để điều khiển cách hiển thị nội dung của một phần tử khi kích thước nội dung lớn hơn kích thước của phần tử.
 - ▶ các giá trị: **visible** (mặc nhiên) | **hidden** | **auto**

KÍCH THƯỚC CỦA PHẦN TỬ – VÍ DỤ

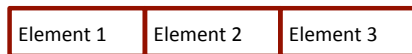
```
<html>
  <body>
    <div style="height:10px; background-color:yellow;
      ">Hello</div><br>
    <div style="height:10px; overflow:hidden;
      background-color:yellow;">Hello</div><br>
    <div style="width:40px; height:40px;
      background-color: yellow; overflow:auto;">
      Hello World</div><br>
    <div style="width:35px;
      height:40px;background-color: yellow;
      overflow-x:auto; overflow-y:hidden;">Hello
      World</div>
  </body>
</html>
```



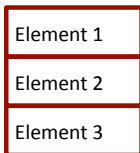
DISPLAY (CHẾ ĐỘ HIỂN THỊ/TRÌNH BÀY)

- ▶ Thuộc tính **display** dùng để chỉ định một phần tử sẽ được **hiển thị** như thế nào (block hay inline).
- ▶ Cú pháp: **display: inline | block | list-item | none | ...**

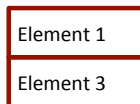
```
<span id="s1">Element 1</span>
<span id="s2">Element 2</span>
<span id="s3">Element 3</span>
```



#s1, #s2, #s3 {display: inline;}



#s1, #s2, #s3 {display: block;}

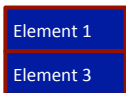


#s1, #s3 {display: block;}
#s2 {display: none;}

VISIBILITY (SỰ NHÌN THẤY)

- ▶ Một số thuộc tính dùng để thiết đặt độ nhìn thấy của một phần tử:
 - ▶ **visibility: visible | hidden:** sự nhìn thấy.
 - ▶ **opacity: <value [0.0, 1.0]>:** độ nhìn thấy (0.0: trong suốt, 1.0: bình thường).

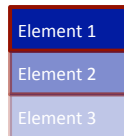
```
<span id="s1">Element 1</span>
<span id="s2">Element 2</span>
<span id="s3">Element 3</span>
```



```
#s1, #s3 {display: block;}
#s2 {display: none;}
```



```
#s1, #s2, #s3 {display: block;}
#s2 {visibility: hidden;}
```



```
#s1 {opacity: 1;}
#s1 {opacity: 0.5;}
#s1 {opacity: 0.25;}
```



FLOAT (SỰ NỔI LÊN)

- ▶ Float trong CSS cho phép một phần tử nổi lên và trôi (di chuyển) về bên trái hoặc phải của phần tử chứa nó (containing element).
- ▶ Các phần tử bên dưới phần tử nổi lên (floating element) sẽ di chuyển lên và bao bọc xung quanh (wrap) phần tử nổi lên.
- ▶ Một số thuộc tính dùng để thiết đặt sự nổi lên của các phần tử:
 - ▶ **float: none | left | right**: chỉ định sự nổi lên của một phần tử.
 - ▶ **clear: none | left | right | both**: chỉ định sự di chuyển của các phần tử phía dưới phần tử floating.

FLOATING – Ví Dụ

Phong Lan

`float: left;`




Họ Phong lan là một họ thực vật có hoa, thuộc bộ Măng tây, lớp thực vật một lá mầm.

Họ Phong lan là một trong những họ lớn nhất của thực vật và có các thành viên mọc trên toàn thế giới.

Các loài lan chủ yếu mọc trên cây cao, sống biểu sinh lâu năm. Chúng được gọi chung là Phong lan. Bên cạnh đó cũng có các loài mọc trong đất, tức là Địa lan và có một số loài mọc trên đá tức Thạch lan.

`float: right;`



`clear: right;`

`clear: left;`

POSITION (VỊ TRÍ)

- ▶ Mặc nhiên, các phần tử trong trang web được sắp xếp tự động từ trái sang phải (inline), từ trên xuống dưới (inline + block).
- ▶ **Vị trí** của các phần tử có thể được thiết đặt bằng các thuộc tính **top**, **right**, **bottom**, **left**. Tuy nhiên, ý nghĩa của các giá trị này phụ thuộc vào thuộc tính **position**.
- ▶ Thuộc tính **position** dùng để chỉ định **sơ đồ định vị** (positioning schema) cho một phần tử:
 - ▶ **position**: static | relative | absolute | fixed
- ▶ Thuộc tính **z-order** được dùng để chỉ định **thứ tự xếp chồng** (stack order) các phần tử khi chúng chồng lên nhau (overlapping). Phần tử có giá trị **z-order** cao hơn sẽ được xếp trên.

CÁC SƠ ĐỒ ĐỊNH VỊ (POSITIONING SCHEMA)

- ▶ **static**: là sơ đồ định vị mặc nhiên
 - ▶ các phần tử inline được sắp xếp từ trái sang phải và tự động xuống dòng khi cần
 - ▶ các phần tử block được xếp từ trên xuống, mỗi phần tử trên 1 dòng riêng
 - ▶ vị trí các phần tử không phụ thuộc giá trị các thuộc tính top, right, bottom, left
- ▶ **relative**:
 - ▶ các phần tử cũng được định vị **tương tự như static**
 - ▶ các thuộc tính top, right, bottom, left sẽ được sử dụng để làm **độ dời** (offset) cho vị trí của phần tử so với vị trí mặc nhiên

CÁC SƠ ĐỒ ĐỊNH VỊ (POSITIONING SCHEMA)

► absolute:

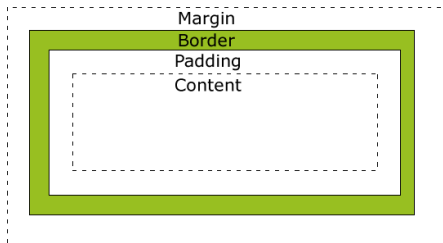
- Vị trí của phần tử (top, right, bottom, left) được tính **tương đối** so với **phần tử cha đầu tiên** có thuộc tính position **không phải static**.
- Nếu không có phần tử cha nào như thế, phần tử cha được xem là phần tử `<html>`.

► fixed:

- vị trí của phần tử được tính tương đối so với cửa sổ trình duyệt
⇒ phần tử sẽ cố định tại một vị trí trên màn hình (không thay đổi kể cả khi nội dung trang web bị cuộn lên/xuống).

BOX MODEL (MÔ HÌNH KHUNG)

- ▶ Mỗi phần tử HTML có thể được coi là một **khung**.
- ▶ Trong CSS, mô hình khung là một khái niệm căn bản trong việc thiết kế và bố trí giao diện.
- ▶ Trong mô hình khung CSS, **một khung bao bọc các phần tử HTML**, nó bao gồm: lề (margins), đường viền (borders), vùng đệm (paddings) và nội dung của khung (content).



MARGIN (LỀ)

- ▶ Là vùng bao quanh đường viền (border), không có màu nền, trong suốt (transparency).
- ▶ Chia tách khung với những thành phần khác.
- ▶ Thiết đặt:
 - ▶ `margin: <top> [<right> [<bottom> [<left>]]]>`
 - ▶ `margin-top/right/bottom/left: <value> | auto`
- ▶ Nếu giá trị margin là auto và khung có kích thước cố định, margin sẽ được đặt lớn nhất có thể.

BORDER (ĐƯỜNG VIỀN)

- ▶ Nằm giữa padding và nội dung; có thể có màu, kiểu và độ dày.
- ▶ Thừa kế màu thiết đặt bởi thuộc tính **color** của khung chứa.
- ▶ Thiết đặt: **border: <thickness> <style> <color>**
 - ▶ style: none, dotted, dashed, solid, double, groove, inset, outset, ridge.
- ▶ Có thể định dạng cho từng t/tính hoặc từng cạnh riêng biệt:
 - ▶ **border-width: <width> | <top> <right> <bottom> <left>**
 - ▶ **border-style: <style-value>**
 - ▶ **border-color: <color> | <top> <right> <bottom> <left>**

VÍ DỤ VỀ ĐƯỜNG VIỀN VÀ LỀ

```
width:340px; margin:auto; border-style: solid
```

```
width:340px; margin-left:auto; margin-right: 5px;
```

```
border-style:dashed; border-color:cyan yellow green blue;
```

```
border-style:groove;
```

```
border-style:outset;
```

```
border-style:inset;
```

```
border-style:ridge;
```

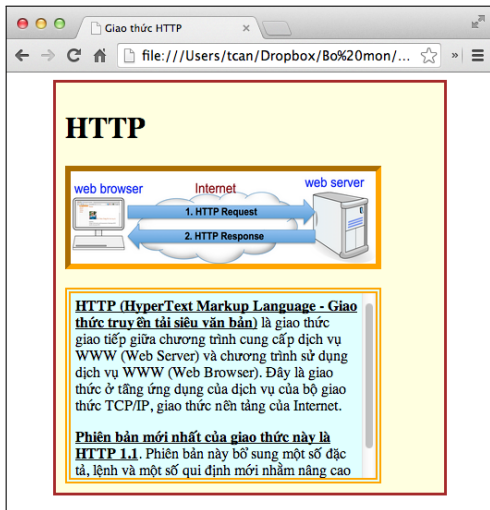
PADDING (VÙNG ĐỆM)

- ▶ Là vùng giữa nội dung và border của khung.
- ▶ Bị ảnh hưởng bởi màu nền của khung.
- ▶ Thiết đặt:
 - ▶ `padding: <size> | <top> <right> <bottom> <left>`
 - ▶ `padding-top/right/bottom/left: <size>`

BÀI TẬP - BOX MODEL

Thiết kế trang web có bố cục và nội dung tương tự như hình bên:

- ▶ chiều ngang của khung: 400px
- ▶ hình: “/figures/http-model.png”
- ▶ tổng chiều ngang khung hình: 340px
- ▶ khoảng cách từ hình đến khung text: 20px
- ▶ chiều cao khung text: 200px



PHẠM VI ÁP DỤNG CSS

- ▶ Việc sử dụng CSS tuân theo **thứ tự** (độ ưu tiên) sau: inline CSS, important, specificity, order, parent-inherit, browser default.
- ▶ **Thừa kế** (inheritance): Các phần tử HTML sẽ thừa kế các thuộc tính từ phần tử cha. Các thuộc tính kế thừa có thể được **định nghĩa lại** cho phần tử con bằng CSS.
- ▶ **important**: thuộc tính này dùng để xác định một CSS rule là quan trọng và sẽ **được ưu tiên sử dụng** khi có nhiều rule cạnh tranh nhau. Mức độ ưu tiên của thuộc tính này **chỉ thấp hơn inline CSS**.

QUI ƯỚC VIẾT CODE

- Ta có thể viết đoạn code CSS như thế này:

```
selector {property: value} selector {property: value}
```

- Tuy nhiên, cách viết sau được khuyến khích:

```
selector {  
  property: value;  
}  
  
selector {  
  property: value;  
}
```

- Các **chú thích** cũng nên được sử dụng để giúp đoạn mã rõ ràng và dễ bảo trì: */* comment (can be multi-line) */*

QUI ƯỚC ĐẶT TÊN

Một số qui ước để đặt ID và tên lớp:

- ▶ Nên **mô tả nội dung**, không phải cách trình bày.

Ví dụ: `warning` thay vì `redbox`

- ▶ Dùng **chữ thường** (lowercase) và dùng **dấu gạch nối** để phân tách các từ.

Ví dụ: `header-info` thay vì `headerInfo`

- ▶ Dùng **tiếp đầu ngữ** (prefix) cho những phần tử con.

Ví dụ: `footer`, `footer-copyright`, `footer-logo`

