Mục lục

Contents

Cơ bản về CSS

Trong bài mở đầu này chúng ta sẽ đi tìm hiểu một số khái niệm và đặc tính của CSS, mà chúng ta cần chú ý trong suốt quá trình làm việc với CSS

I. CSS là gì

CSS (Cascading Style Sheets) được hiểu một cách đơn giản đó là cách mà chúng ta thêm các kiểu hiển thị (font chữ, kích thước, màu sắc...) cho một tài liệu Web

II. Một số đặc tính cơ bản của CSS

1. CSS quy định cách hiển thị của các thẻ HTML bằng cách quy định các thuộc tính của các thẻ đó (font chữ, màu sắc). Để cho thuận tiện bạn có thể đặt toàn bộ các thuộc tính của thẻ vào trong một file riêng có phần mở rộng là ".css"

   CSS nó phá vỡ giới hạn trong thiết kế Web, bởi chỉ cần một file CSS có thể cho phép bạn quản lí định dạng và layout trên nhiều trang khác nhau. Các nhà phát triển Web có thể định nghĩa sẵn thuộc tính của một số thẻ HTML nào đó và sau đó nó có thể dùng lại trên nhiều trang khác.

2. Có thể khai báo CSS bằng nhiều cách khác nhau. Bạn có thể đặt đoạn CSS của bạn phía trong thẻ <head>...</head>, hoặc ghi nó ra file riêng với phần mở rộng ".css", ngoài ra bạn còn có thể đặt chúng trong từng thẻ HTML riêng biệt

   Tuy nhiên tùy từng cách đặt khác nhau mà độ ưu tiên của nó cũng khác nhau. Mức độ ưu tiên của CSS sẽ theo thứ tự sau.

   1. Style đặt trong từng thẻ HTML riêng biệt
   2. Style đặt trong phần <head>
   3. Style đặt trong file mở rộng .css
   4. Style mặc định của trình duyệt

   Mức độ ưu tiên sẽ giảm dần từ trên xuống dưới.

3. CSS có tính kế thừa: giả sử rằng bạn có một thẻ đã được khai báo ở đầu file css với các thuộc tính như sau:

```
4.  #vidu {
5.    width: 200px;
6.    height: 300px;
7.  }
```

Ở một chỗ nào đó trong file css bạn lại khai báo một lần nữa thẻ với các thuộc tính.

```
#vidu {
  width: 400px;
  background-color: #CC0000;
}
```

Sau đoạn khai báo này thì thẻ sẽ có thuộc tính:

```
#vidu {
  width: 400px; /* Đè lên khai báo cũ */
  height: 300px;
  background-color: #CC0000;
}
```

Cú pháp của CSS

Sau khi hiểu là nắm bắt được một số đặc tính của CSS chúng ta tiếp tục đi tìm hiểu về cú pháp và cách khai báo của các thẻ CSS

Cú pháp của CSS được chia làm 3 phần. phần thẻ chọn (selector), phần thuộc tính (property), phần nhãn (value).

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

Nếu nhãn của bạn có nhiều từ bạn nên đặt nhãn của bạn vào trong dấu nháy kép

```
   p {font-family: "sans serif"}
```

Trong trường hợp thẻ chọn của bạn nhiều thuộc tính thì các thuộc tính sẽ được ngăn cách bởi dấu (;).

```
p {text-align:center;color:red}
```

Khi thẻ chọn có nhiều thuộc tính thì chúng ta nên để mỗi thuộc tính ở trên một dòng riêng biệt.

```
p {
  text-align: center;
  color: black;
  font-family: arial
}
```

Làm sao chèn CSS vào trong trang Web

Bạn đã có một file CSS của bạn, bây giờ công việc tiếp theo là làm thế nào để chèn những đoạn CSS của bạn vào trong trang, Và xem chúng hoạt động như thế nào. Trong phần này chúng ta sẽ đi tìm hiểu chi tiết về cách chèn một đoạn style trong trang HTML hay liên kết tới một file CSS viết sẵn.

Khi trình duyệt đọc đến CSS, thì toàn bộ Website sẽ được định dạng theo các thuộc tính đã được khai báo trong phần CSS. Có ba cách cho phép chúng ta chèn định dạng CSS vào trong Website.

1. CSS được khai báo trong file riêng.

Toàn bộ mã CSS được chứa trong file riêng có phần mở rộng .css là một ý tưởng được dùng khi một file CSS sẽ được áp dụng cho nhiều trang khác nhau. Bạn có thể thay đổi cách hiển thị của toàn bộ site mà chỉ cần thay đổi một file CSS. Trong cách này thì file CSS sẽ được chèn vào văn bản HTML thông qua thẻ <link>...</link>. Ta có cú pháp như sau:

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

  <body>
  </body>
</html>
```

Trình duyệt sẽ đọc toàn bộ các định dạng được quy định trong file mystyle.css và định dạng cho văn bản HTML.

File CSS có thể được soạn thảo bằng một số trình duyệt khác nhau. Trong file không được chứa mã HTML, khi ghi lại chúng ta bắt buộc phải ghi lại với phần mở rộng là **.css**. Giả sử chúng trong file mystyle.css ở trên chứa đoạn mã sau:

```
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
```

Không bao giờ sử dụng khoảng trắng trong nhãn, giả sử rằng nếu bạn dùng margin-left: 20 px; thay cho margin-left: 20px; thì IE6 sẽ hiểu còn các trình duyệt như Firefox, Opera sẽ không hiểu

2. Chèn CSS trong tài liệu HTML

Chèn thẳng CSS trong tài liệu được áp dụng trong trường hợp những định dạng CSS này chỉ giành riêng cho tài liệu HTML đó. Khi bạn chèn trực tiếp thì đoạn mã của bạn phải đặt trong thẻ <style> và đặt trong phần <head>.

```
<head>
<style type="text/css">
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
</style>
</head>
```

Có một số trình duyệt cũ sẽ không hiểu thẻ <style>, nó sẽ bỏ qua thẻ này. Tuy nhiên thì nội dung trong thẻ <style> vẫn hiển thị ra trang HTML. Vì vậy mà chúng ta sẽ phải dùng định dạng chú thích trong HTML để chứa phần nội dung của thẻ <style>.

```
<head>
<style type="text/css">
<!--
hr {color: sienna}
p {margin-left: 20px}
body {background-image: url("images/back40.gif")}
-->
</style>
</head>
```

3. Chèn trực tiếp vào thẻ của HTML(inline style)

Inline style được sử dụng nhiều trong trường hợp một thẻ HTML nào đó cần có style riêng cho nó.

Inline style được áp dụng cho chính thẻ HTML đó, cách này sẽ có độ ưu tiên lớn nhất so với hai cách trên. Dưới đây là một ví dụ mà chúng ta dùng Inline style

```
<p style="color: sienna; margin-left: 20px">
This is a paragraph
</p>
```

4. Nhiều Stylesheet

Trong trường hợp mà có một số thẻ có cùng định dạng, chúng ta có thể gộp chúng lại với nhau. Giả sử như sau:

```
h1, h2, h3 {
  margin-left: 10px;
  font-size: 150%;
  ...
}
```

Giống đoạn mã trên thì cả ba thẻ h1, h2, h3 đều có cùng 3 thuộc tính như trên.

Các vấn đề về văn bản và cách định dạng văn bản

Thuộc tính CSS text cho phép bạn hoàn toàn có thể quản lí được các thuộc tính của văn bản, bạn có thể quản lí được sự ẩn hiện của nó, thay đổi màu sắc, tăng hoặc giảm khoảng cách giữa các ký tự trong một đoạn, căn chỉnh việc dóng hàng (align),...

Các thuộc tính của text mà CSS hỗ trợ

Đặt màu cho một đoạn văn bản

Để đặt màu cho một đoạn văn bản chúng ta có thể dùng thuộc tính: **color: #mã màu;**

```
p {
  color: #333333;
}
```

Đặt màu nền cho đoạn văn bản.

Bạn có thể đặt màu nền (background) cho đoạn văn bản bằng thuộc tính **background-color: #mã màu;**

```
p {
  background-color: #FFFF00;
}
```

Căn chỉnh khoảng cách giữa các ký tự.

Khoảng cách giữa các ký tự trong một đoạn văn bản có thể được tăng hoặc giảm bởi thuộc tính **letter-spacing: khoảng cách;**

```
h3 {
  letter-spacing: 2em;
}

h1 {
  letter-spacing: -3em;
}
```

Căn chỉnh khoảng cách giữa các dòng.

Thuộc tính **line-height: khoảng cách;** sẽ giúp bạn căn chỉnh khoảng cách giữa các dòng trong một đoạn văn bản.

```
p {
  line-height: 150%; // line-height: 15px;
```

```
}
```

Dóng hàng

Để gióng hàng cho một đoạn văn bản chúng ta sẽ dùng thuộc tính **text-align: vị trí;**

```
p {
  text-align: left; /* left | center | right */
}
```

Trang hoàng thêm cho đoạn văn bản.

Một đường gạch chân hoặc đường gạch ngang dòng văn bản sẽ làm cho đoạn văn bản của bạn thêm sinh động. Để tô điểm thêm cho đoạn văn bản chúng ta sẽ dùng thuộc tính **text-decoration: thuộc tính;**

```
h3 {
  text-decoration: underline; /* Gạch chân */
}

h2 {
  text-decoration: line-through; /* Gạch ngang */
}

h1 {
  text-decoration: overline; /* kẻ trên */
}
```

Chỉnh vị trí của đoạn văn bản (indent).

Thuộc tính **text-indent: vị trí;** sẽ căn chỉnh vị trí của dòng văn bản theo chiều ngang.

```
h1 {
  text-indent: -2000px; /* text-indent: 30px; */
}
```

Điều kiển các ký tự trong một đoạn văn bản.

Bạn có thể điều khiển toàn bộ đoạn văn bản là chữ hoa hay chữ thường bởi thuộc tính **text-transform: kiểu chữ;**

```
p.uppercase {
  text-tranform: uppercase;
}

p.lowercase {
  text-tranform: lowercase;
}

p.capitalize {
  text-tranform: capitalize;
```

```
}
```

Đặt hướng cho đoạn văn bản.

Hướng của đoạn văn bản có thể từ trái qua phải hay từ phải qua trái chúng ta có thể điều khiển bởi thuộc tính direction: hướng;

```
div.rtl {
  direction: rtl; /* Right to left */
}

div.ltr {
  direction: ltr; /* Left to right */
}
```

Tăng khoảng cách giữa các từ.

Khoảng cách giữa các từ có thể được tăng bởi thuộc tính word-spacing: khoảng cách;

```
  word-spacing: 30px;
```

Làm mất tác dụng của đường bao của một thẻ HTML.

Để làm mất tác dụng đường bao của một thẻ HTML chúng ta dùng thuộc tính **white-space: giá trị;**

```
p {
  white-space: nowrap;
}
```

Thuộc tính **white-space** sẽ làm cho toàn bộ đoạn văn bản ở trên một dòng.

Các thuộc tính của font chữ và định nghĩa font chữ cho văn bản

Các thuộc tính về font chữ sẽ cho phép bạn thay đổi họ font (font family), độ đậm (boldness), kích thước (size) và kiểu font (style).

01Đặt font cho đoạn văn bản.

Để đặt một loại font chữ nào đó cho đoạn văn bản thì chúng ta sẽ sử dụng thuộc tính **font-family:**

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

Thông thường bạn cần phải khai báo họ của font ở cuối (trong ví dụ trên thì sans-serif là chỉ tới 1 họ font) để trong trường hợp máy của người duyệt Web không có các font như mình đã đặt thì nó sẽ lấy font mặc định của họ font trên.

02Đặt đoạn văn bản sử dụng font nhãn caption.

```
p.caption {
  font: caption;
}
```

03Đặt kích thước font cho đoạn văn bản.

Khi chúng ta muốn những đoạn văn bản hoặc tiêu đề có kích thước của chữ khác nhau, chúng ta có thể sử dụng thuộc tính **font-size:**

```
h1 {
  font-size: 20px;
}

h3 {
  font-size: 12px;
}
```

04Định lại kích thước font bằng thuộc tính font-size-ajust:

```
p {
  font-size-ajust: 0.60;
}
```

05Đặt kiểu font cho đoạn văn bản.

Chữ đậm, chữ nghiêng,... được đặt với thuộc tính **font-style:**

```
p {
  font-style: italic; /* normal  |  italic |  oblique */
}
```

06

Muốn hiển thị font ở dạng small-caps hoặc ở dạng normal thì chúng ta sẽ sử dụng thuộc tính **font-variant**. Thuộc tính này có hai giá trị normal và small-caps

```
p {
  font-variant: normal; /* normal  |  small-caps */
}
```

07Đặt độ đậm nhạt của font.

Khi chúng ta muốn thay đổi độ đậm nhạt của văn bản chúng ta sẽ dùng thuộc tính **font-weight:**. Chúng ta có thể đặt giá 3 loại giá trị cho thuộc tính này 1. normal(bình thường), 2. bold(đậm), 3. 300(đặt dạng số)

```
h3 {
  font-weight: bold;
}
```

08Khai báo các thuộc tính font ở dạng shorthand.

```
p {
  font: italic small-caps 900 12px arial;
}
```

Đường viền và các thuộc tính của đường viền

Các thuộc tính của đường viền (border) sẽ cho phép đặt các giá trị đặc biệt cho đườn viền như kiểu đường viền, kích thước, màu sắc. Thuộc tính này sẽ được áp dụng cho các thẻ HTML như , <li>, <table>,...

Trong thuộc tính đường viền (border) chúng ta có 3 giá trị cơ bản đó là:

1. border-color:
2. border-width:
3. border-style:

01Thuộc tính màu của đường viền

Để đặt màu cho đường viền chúng ta sẽ đặt thông số màu cho thuộc tính **border-color:**

```
div.color {
  border-color: #CC0000;
}
```

02Đặt chiều rộng cho đường viền (border)

Nếu muốn đặt chiều rộng của đường viền chúng ta sẽ đặt giá trị cho thuộc tính **border-width:**

```
div.borerwidth {
  border-width: 2px;
}
```

STT	Giá trị
1	thin
2	medium
3	thick
4	length

03Chọn kiểu của đường viền

Bạn có thể sử dụng thuộc tính **border-style** để đặt kiểu cho đường viền. Chúng ta có thể gán cho thuộc tính này 9 giá trị khác nhau tương ứng với 9 kiểu đường viền khác nhau.

```
div.borderstyle {
  border-style: solid;
}
```

STT	border-style
1	none

2 hidden
3 dotted
4 dashed
5 solid
6 double
7 groove
8 ridge
9 inset
10 outset

Với 4 phía của đối tượng ta có 4 thuộc tính border tương ứng:

1. border-top:
2. border-right:
3. border-bottom:
4. border-left:

Ứng với đường viền của mỗi phía chúng ta đều có 3 giá trị (color, width, style)

STT	Phía	Thuộc tính
1	top	border-top-color: border-top-width: border-top-style:
2	right	border-right-color: border-right-width: border-right-style:
3	bottom	border-bottom-color: border-bottom-width: border-bottom-style:
4	left	border-left-color: border-left-width: border-left-style:

Chúng ta có thể dùng phương pháp viết mã giản lược (shorthand) để viết các thuộc tính của đường viền gọn hơn. Giả sử chúng ta đặt thuộc tính border của thẻ với độ rộng bằng 1, kiểu solid và màu là #CC0000

```
div.border {
  border: 1px solid #CC0000;
}
```

Các thuộc tính của margin

Thuộc tính **margin** sẽ định nghĩa khoảng trắng bao quanh một phần tử HTML. Nó có thể dùng giá trị âm để lồng nội dung vào với nhau. Tương ứng với 4 phía của một phần tử chúng ta có 4 thuộc tính tương ứng. Mặt khác để viết cho gọn chúng ta cũng có thể dùng cách viết giản lược để định nghĩa các giá trị cho thuộc tính **margin**.

Đối với các trình duyệt Netcape và IE thì giá trị mặc định của thuộc tính **margin** là 8px. Nhưng Opera thì không hỗ trợ như vậy. Để cho thống nhất chúng ta có thể đặt **margin** mặc định cho toàn bộ các phần tử.

Các giá trị mà thuộc tính **margin** có thể nhận được đó là: auto, length, %. Chúng ta đặt giá trị nào là tùy thích cộng với việc tương ứng tỉ lệ với các phần tử khác.

Tương ứng với 4 phía ta có 4 thuộc tính:

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

Để cho gọn chúng ta cũng có thể việt thuộc tính **margin** dưới dạng shorthand

```
div.margin {
  margin: 10px 4px 5px 9px; /* top  |  right  |  bottom  |  left*/
}
```

Thuộc tính đường bao ngoài (Outline)

Thuộc tính **Outline** sẽ vẽ một đường bao ngoài toàn bộ một phần tử HTML, đối với phần tử có đường viền (border) thì đường bao này sẽ bao trọn đường viền của phần tử đó. Cũng tương tự như đường viền bạn có thể đặt cho nó các thuộc tính về màu sắc, độ lớn và kiểu.

Có một điều cần chú ý là các thuộc tính của đường bao này có thể không được hỗ trợ trên IE

01Đặt thuộc màu cho đường bao

Nếu muốn đặt màu cho đường bao chúng ta có thể sử dụng thuộc tính **outline-color:**

```
p {
  outline-color: #CC0000;
}
```

02Đặt độ rộng cho đường bao.

Tương tự như đường viền, để đặt độ rộng cho đường bao chúng ta đặt giá trị độ lớn cho thuộc tính **outline-width:**

```
p {
  outline-width: 2px;
}
```

03Chọn kiểu đường bao

Để chọn kiểu cho đường bao chúng ta sẽ đặt lần lượt các giá trị cho thuộc tính **outline-style:**

```
p {
  outline-style: dotted;
}
```

STT	outline-style
1	none
2	hidden
3	dotted
4	dashed
5	solid
6	double
7	groove
8	ridge
9	inset

10 outset

Để cho gọn chúng ta cũng có thể viết các giá trị của thuộc tính **Outline** dưới dạng shorthand như sau:

```
div.outline {
  outline: 1px solid #000;
}
```

CSS padding

CSS padding sẽ định nghĩa khoảng trống giữa mép của các phần tử tới các phần tử con hoặc nội dung bên trong. Chúng ta không thể gán giá trị âm cho thuộc tính này. Cũng giống như margin thuộc tính padding cũng tương ứng với 4 phía của phần tử.

Tương ứng với 4 phía của phần tử chúng ta có 4 thuộc tính **padding** tương ứng đó là:

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

Các giá trị có thể gán cho các thuộc tính này là : % hoặc length

Để viết cho gọn chúng ta cũng có thể viết thuộc tính **padding** dưới dạng shorthand.

```
div.padding {
  padding: 5px 3px 2px 8px;
}
```

Làm thẻ div có thanh cuộn (scrollbar) giống iFrame

Bài viết dưới đây sẽ giới thiệu với các bạn cách làm cho thẻ có thanh cuộn giống khi ta sử dụng iFrame để load một trang khác.

Trước hết chúng ta có ví dụ về đoạn mã HTML như sau:

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Strict//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-strict.dtd">
<html xmlns="http://www.w3.org/1999/xhtml" xml:lang="en" lang="en">
<head>
  <title>example page</title>
</head>
<body>
<div id="page">
  <h1>Title</h1>
  <div id="scroll_box">
  <p>
    Put a long text in here. It will be crollable.<br/>
    Put a long text in here. It will be crollable.<br/>
    Put a long text in here. It will be crollable.<br/>
    Put a long text in here. It will be crollable.<br/>
    Put a long text in here. It will be crollable.<br/>
    Put a long text in here. It will be crollable.<br/>
    Put a long text in here. It will be crollable.<br/>
  </p>
</div>
<p>
  This content follows after the scrollable box.
</p>
</div>
</body>
</html>
```

Bây giờ chúng ta tiến hành style cho thẻ , trước hết chúng ta cần phải đặt cố định chiều rộng và chiều cao của cho thẻ

```
#scroll_box {
  height: 120px;
  width: 400px;
  ...
}
```

Để cho thẻ hiển thị thanh cuộn chúng ta sẽ đặt thuộc tính **display** của thẻ là **auto**

```
#scroll_box {
  height: 120px;
  width: 400px;
  display: auto;
  border: 1px solid #CCCCCC;
```

```
  margin: 1em 0;
}
```

Khi bạn đưa nội dung dài hơn chiều cao và chiều rộng của thẻ thì trình duyệt sẽ tự động sinh ra thanh cuôn ngang và thanh cuộn dọc giống như iFrame. Kỹ thuật này hiện có khá nhiều Website sử dụng để có cách thể hiện nội dung riêng biệt của mình.

Style một kiểu Bubble đơn giản

Thông thường khi chúng ta duyệt Web, chúng ta sẽ không để ý tới cách trình bày của trang Web đó, mà chỉ để ý đến thông tin của nó. Nhưng nếu chúng ta để ý một chút thì nhận thấy rằng, nếu Website nào biết vận dụng các kiểu Typography hợp lý thì sẽ giúp người đọc phân biệt các mức độ thông tin rất nhanh. Điều đó sẽ có tác động tốt đối với người đọc

Để giúp các bạn có thói quen áp dụng các kiểu Typography vào trong Website của mình. Trong bài viết này tôi xin hướng dẫn các bạn làm một kiểu bubble đơn giản

Dạng bubble thường được sử dụng trong việc trích dẫn lời nói của một tác giả nào đó, hoặc trong các lời bình luận trong các **blog**.

1. Định dạng HTML

Đây là ví dụ của một kiểu bubble đơn giản, các bạn có thể ứng dụng chúng vào ngay trong site của mình. Chúc các bạn thành công!

Tác giả: CSSYeah.com

Để làm kiểu **bublle** đơn giản này chúng ta sẽ sử dụng hai thẻ . Thẻ chứa lời bình luận, còn thẻ <div class="footercomment"< sẽ được dùng chứa thông tin tác giả.

Thẻ trong trường hợp này có tác dụng giới hạn kích thước cho hai thẻ bên trong. Khi đưa vào trong Website bạn có thể thay đổi kích thước của nó để cho phù hợp với nội dung Website.

2. Định dạng CSS

Ta có định dạng CSS của kiểu **bublle** trên như sau:

```
#wrapper {
        width: 400px;
        margin: 0 auto;
        padding: 0;
}
```

```
div.comment {
        margin: 0;
        padding: 10px;
        background: #E8ECEF;
}

div.commentfooter {
        padding: 8px 0 0 22px;
        background: url(arrow-down.gif) no-repeat 20px 0 #FFFFFF;
}

span.author {
        padding-left: 15px;
        background: url(author.gif) no-repeat left center;
        font-weight: bold;
}

div.commentfooter a {
        color: #CC0000;
}

div.commentfooter a:hover,
div.commentfooter a:active {
        text-decoration: none;
}
```

Trong phần nội dung do chúng ta có thể dùng nhiều lần đạng **bubble** này, do vậy chúng ta định dạng thuộc tính class cho thẻ

Căn bảng vào giữa màn hình.

Bình thường thì bạn muốn đặt một table có chiều rộng cố định vào giữa màn hình thì bạn làm thế nào? có phải bạn đặt thuộc tính **align="center"** vào trong thẻ <table>?. Nếu bạn làm như vậy thì bạn chỉ được kết quả đúng như ý bạn trên một số trình duyệt, còn một số trình duyệt sẽ không thể hiện kết quả như bạn muốn.

Sau đây tôi muốn giới thiệu với bạn một tips nhỏ trong CSS để đặt một table có kích thước cố định vào giữa trang.

Giả sử rằng bạn có một table như sau: <table class="center">. Nếu bây giờ bạn muốn đặt table đó vào giữa trang màn hình bạn chỉ việc đặt cho hai thuộc tính **margin-left** và **margin-right** của table có giá trị bằng **auto**.

Chúng ta có mã CSS như sau:

```
table.center {
  width: 780px;
  margin-left: auto;
  margin-right: auto;
}
```

Menu dạng tab - Phần I.

Trong chúng ta tôi giám chắc rằng không còn ít người vẫn còn mặc cảm với những menu dạng tab, không phải vì nó xấu mà luôn nghĩ rằng làm nó khó và cấu trúc của nó rất phức tạp. Chính bản thân tôi trước đây cũng như vậy. Nhưng thực ra có phải như vậy không? để trả lời câu hỏi đó bây giờ chúng ta cùng nhau thử Style cho một kiểu tab đơn giản để xem nó có thực sự khó như chúng ta vẫn thường nghĩ không?.

Trong menu dạng này bao giờ cũng gồm 2 phần, phần 1 để chứa các tab và phần thứ hai là phần để chứa nội dung của các tab. để có hình dung rõ hơn chúng ta sẽ xem hình ảnh minh họa dưới đây.

Đầu tiên chúng ta sẽ định dạng cho phần khung của tab như sau:

```
div#wrapper {
  margin: 50px;
  padding: 0;
}
```

Chúng ta sẽ dùng định dạng của thẻ <ol> để tạo lên các tab ở phần 1. Về nguyên tắc thì chúng ta có thể dùng một trong ba thẻ <ul>, <ol>, <dl>. Nhưng tại sao tôi lại dùng thẻ <ol>, là bởi vì một lý do nào đó mà trình duyệt của người xem không tải được CSS từ Website của bạn thì họ vẫn hiểu được cấu trúc tab của bạn. Định dạng của phần một như sau

```
<ol class=”tab”>
<li class=”active”>Trang chủ</li>
<a href=”gioithieu.html” title=”Giới thiệu”>Giới thiệu</a>
<a href=”sanpham.html” title=”Sản phẩm”>Sản phẩm</a>
<a href=”tintuc.html” title=”Tin tức”>Tin tức</a>

<a href=”lienhe.html” title=”Liên hệ”>Liên hệ</a>
</ol>
```

Bây giờ chúng ta sẽ định dạng sao cho các tab nằm trên cùng một hàng và có hình dáng của tab.

```
ol.tab {
  background: url(../dot.gif) repeat-x left bottom; /* ảnh 1px */
  list-style: none;
  margin: 0;
  padding: 6px 0;
  position: relative;
}

ol.tab li {
  background: #F2F5FA;
  border: 1px solid #D3DDED;
  display: inline; /* các thẻ li ở trên một dòng */
  margin-right: 5px;
```

```
  padding: 0;
}
```

Trong cách định dạng này ta có sử dụng một kỹ thuật nhỏ, đó là chúng ta có sử dụng một ảnh .gif có kích thước 1px x 1px để thay thế border-bottom của

Khi đang ở trong trang nào đó thì tab của trang đó sẽ có màu khác với những tab khác, do vậy chúng ta sẽ đặt cho tab đó một class là **active**. Vì vậy chúng ta cần phải định dạng cho riêng cho các tab **active** như sau:

```
ol.tab li.active {
  background: #FFF;
  border-bottom: 1px solid #FFF;
  font-weight: bold;
  padding: 5px 10px;
}
```

Tiếp theo chúng ta sẽ định dạng cho các link nằm trong thẻ

```
ol.tab a {
  font-weight: bold;
  margin: 0;
  padding: 5px 10px;
}
```

Cuối cùng bây giờ chúng ta sẽ định dạng phần chứa nội dung của mỗi tab. Phần này chỉ đơn giản là chúng ta định dạng cho thẻ chứa nội dung.

```
div#content {
  border: 1px solid #D3DDED;
  border-top: none;
  padding: 10px;
}
```

Kỹ thuật làm chữ hoa đầu dòng (Drop cap)

Drop cap là một trong những typography hay được dùng trong các tạp chí trên giấy cũng như các tạp chí điện tử. Nó thường được dùng ở đầu mỗi bài báo với font chữ lớn và màu sắc nổi trội. Theo quy luật nhìn của mắt nếu một đối tượng nổi bật hơn các đối tượng xung quanh, nhì nó sẽ có tác động mạnh tới thị giác của người quan sát. Chính vì lý do đó mà người ta vẫn thường dùng cách này cho các bài báo nổi bật cần sự chú ý cao.

Tuy nhiên trong khuôn khổ của lĩnh vực nghiên cứu chúng ta sẽ đi tìm hiểu một số kỹ thuật làm chữ hoa đầu dòng (Drop cap).

1. Cách thứ nhất

Đầu tiên chúng ta sẽ đi tìm hiểu cách làm Drop cap chính thống. Trong cách này chúng ta sẽ sử dụng đến kỹ thuật selectors trong CSS. cụ thể là chúng ta sẽ dùng đến thuộc tính **:first-letter**.

Ta có định dạng HTML:

```
<p class="pdropcap">Lorem ipsum dolor sit amet consectetuer at et
Aenean ac dolor. Ligula nulla ac id pede sit consectetuer ipsum
malesuada convallis habitant. Neque at pellentesque pharetra Aenean
accumsan orci Proin leo tellus eu. Dictumst Integer ut adipiscing
porttitor dolor Morbi ut id lorem auctor. Massa tellus Morbi enim
tellus pede vel suscipit hac sapien Cras. Cursus velit hendrerit
lobortis elit elit sed ut In.</p>
```

Định dạng CSS:

```
p.pdropcap:first-letter {
  float: left;
  padding: 4px 8px 0 0;
  display: block;
  color: #336699;
  font: 60px/50px Georgia, Times, serif;
}
```

Kết quả hiển thị:

Lorem ipsum dolor sit amet consectetuer at et Aenean ac dolor. Ligula nulla ac id pede sit consectetuer ipsum malesuada convallis habitant. Neque at pellentesque pharetra Aenean accumsan orci Proin leo tellus eu. Dictumst Integer ut adipiscing porttitor dolor Morbi ut id lorem auctor. Massa tellus Morbi enim tellus pede vel suscipit hac sapien Cras. Cursus velit hendrerit lobortis elit elit sed ut In.

Chúng ta cần phải đặt **class** cho thẻ <p class="pdropcap"> nhằm mục đích phân biệt nó với các thẻ <p> khác trong trang. Trong trường hợp bạn không đặt class thì toàn bộ các ký tự đầu dòng của thẻ <p> trong trang sẽ hiển thị định dạng chữ hoa, mà điều đó thì không phải là điều mong muốn.

2. Cách thứ hai

Trong cách thứ hai này ta sẽ dùng thêm một thẻ **** để bao ký tự chữ hoa ở đầu dòng

Định dạng HTML:

```
<p><span class="pdropcap">N</span>orem ipsum dolor sit amet
consectetuer at et Aenean ac dolor. Ligula nulla ac id pede
sit consectetuer ipsum malesuada convallis habitant. Neque
at pellentesque pharetra Aenean accumsan orci Proin leo
tellus eu. Dictumst Integer ut adipiscing porttitor dolor Morbi
ut id lorem auctor. Massa tellus Morbi enim tellus pede vel
suscipit hac sapien Cras. Cursus velit hendrerit lobortis
elit elit sed ut In.</p>
```

Định dạng CSS:

```
.dropcap {
  float: left;
  padding: 4px 8px 0 0;
  display: block;
  color: #336699;
  font: 60px/50px Georgia, Times, serif;
}
```

Kết quả hiển thị:

Norem ipsum dolor sit amet consectetuer at et Aenean ac dolor. Ligula nulla ac id pede sit consectetuer ipsum malesuada convallis habitant. Neque at pellentesque pharetra Aenean accumsan orci Proin leo tellus eu. Dictumst Integer ut adipiscing porttitor dolor Morbi ut id lorem auctor. Massa tellus Morbi enim tellus pede vel suscipit hac sapien Cras. Cursus velit hendrerit lobortis elit elit sed ut In.

Cách này có một ưu điểm là ký tự hoa đó có thể đặt ở bất cứ đâu trong bài viết miễn là chúng ta thêm thẻ vào ký tự mà ta muốn chuyển thành dang drop cap.

Để cho phong phú chúng ta cũng có thể thêm một số các thuộc tính như màu nền, màu chữ,... vào cho thêm sinh động. Ta có ví dụ như sau:

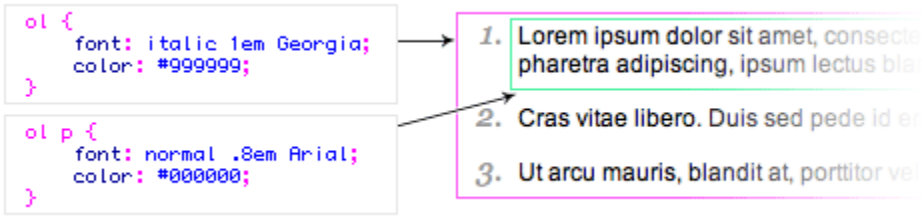
Ac dolor. Ligula nulla ac id pede sit consectetuer ipsum malesuada convallis habitant. Neque at pellentesque pharetra Aenean accumsan orci Proin leo tellus eu. Dictumst Integer ut adipiscing

porttitor dolor Morbi ut id lorem auctor. Massa tellus Morbi enim tellus pede vel suscipit hac sapien Cras. Cursus velit hendrerit lobortis elit elit sed ut In.

Trang trí cho danh sách có thứ tự

Mặc định hầu hết các trình duyệt (Browser) đều hiển thị chữ số của danh sách có thứ tự (order list) theo font mặc định mà chúng ta sử dụng cho nội dung site. Trong bài chỉ dẫn ngắn này tôi xin hướng dẫn các bạn làm thế nào để sử dụng hai thẻ <ol> (orderlist) và <p> để thêm trang trí thêm cho danh sách có thứ tự orderlist.

Để có hình dung rõ hơn các bạn hãy nhìn vào hình vẽ dưới đây:



Chúng ta sẽ sử dụng font cho các chữ số là font "Georgia", còn font cho thẻ <p> là font Arial.

Với hình vẽ trên chúng ta có mã nguồn HTML như sau:

```
<ol>
<li>
<p>This is line one</p>
</li>
<li>
<p>Here is line two</p>
</li>
<li>
<p>And last line</p>
</li>
</ol>
```

Dưới đây là mã CSS cho hai thẻ <ol> và <p>

```
ol {
  font: italic 1em Georgia, Times, serif;
  color: #999999;
}

ol p {
  font: normal .8em Arial, Helvetica, sans-serif;
  color: #000000;
}
```

Bài viết được tổng hợp từ **Web Designer Wall**

Kỹ thuật tải ảnh trước bằng CSS

Trong file CSS của bạn có sử dụng đến một số ảnh **background**, nhưng những ảnh đó sẽ không được trình duyệt tải về trong bộ đệm (cache) trên máy ngay từ đầu, mà khi nào sử dụng đến thì nó mới được tải về. Chính vì vậy đôi khi nó sẽ sinh ra độ trễ khi chúng ta lần đầu tiên khi vào trang Web.

Chúng ta lấy một ví dụ như sau: Giả sử rằng trong thẻ <a> của bạn có sử dụng hai ảnh, một ảnh làm dùng để hiển thị ngay từ đầu, còn một ảnh khác sử dụng cho thuộc tính **:hover**. KHi lần đầu tiên bạn vào Website khi **hover** chuột lên thẻ <a> đó thì nó sẽ không hiển thị anh thứ hai ngay mà sau một khoảng thời gian khoảng 2 đến 3 giây sau thì mới hiện tùy theo tốc độ đường truyền.

Để khắc phục điều đó chúng ta có một thủ thuật (tips) nhỏ như sau:

Đối với một số ảnh mà chúng ta cần sử dụng làm background trong file css, mà chúng ta không muốn có độ trễ như ví dụ trên thì chúng ta cần có biện pháp tải trước những ảnh đó về máy.

Chúng ta sẽ đặt toàn bộ các ảnh mà muốn tải trước về vào trong một thẻ CSS như sau:

```
#preLoadImages {
  width: 0px;
  height: 0px;
  background: url(../images/anh1.gif);
  background: url(../images/anh2.gif);
  background: url(../images/anh3.gif);
  ...
}
```

Bạn có thể đặt đoạn mã CSS trên vào trong file CSS của bạn, hoặc đặt vào trong thẻ <head> của trang. Ví dụ

```
  <html>
    <head>
      <style type="text/css">
        #preLoadImages {
          width: 0px;
          height: 0px;
          background: url(../images/anh1.gif);
          background: url(../images/anh2.gif);
          background: url(../images/anh3.gif);
          ...
        }
      </style>
    </head>
```

Sau đó chúng ta đặt đoạn thẻ trên vào ngay dưới thẻ <body> của Website của bạn. Khi trình duyệt đọc tới thẻ đó nó sẽ load toàn bộ những ảnh được liệt kê trong danh sách trên vào trong bộ đệm của trình duyệt.

```
<body>
  <div id="preLoadImages"></div>
...
```

Fix min-height cho IE

Hiện tại hầu hết các trình duyệt như Firefox, Opera, Safari,... đều hiểu được thuộc tính **min-height:** trong CSS, nhưng IE lại không thể hiểu được thuộc tính đó. Chính vì vậy mà khi chúng ta muốn đặt **min-height** cho một đối tượng nào đó thì chúng ta cần phải Fix để sao cho IE hiển thị được giống như những trình duyệt khác.

Giả sử rằng chúng ta có một thẻ chúng ta muốn đặt min-height cho thẻ này. Khi đó chúng phải làm như sau:

```
/* Cho tất cả các trình duyệt */
.box {
  width:20em;
  padding:0.5em;
  border:1px solid #000;
  min-height:8em;
  height:auto;
}

/* Fix cho IE */
/*\*/
* html .box {
  height: 8em;
}
/**/
```

Update

Chúng ta cũng có thể dùng giải pháp như sau:

```
selector {
  min-height:500px;
  height:auto !important;
  height:500px;
}
```

Do IE6 không hiểu được thuộc tính **min-height** do đó chúng ta phải cố định chiều cao cho nó. Trong trường hợp nội dung có chiều cao lớn hơn **height** thì chiều cao trong IE6 sẽ tự động kéo dài xuống còn trong Firefox và một số trình duyệt khác sẽ sử dụng thuộc tính **height: auto;**.

Đặt min-width cho IE6

Như chúng ta đã biết với sự phát triển nhanh chóng của ngành công nghiệp phần cứng, ngày càng có nhiều màn hình có độ phân giải cao. Nó cho phép chúng ta có một không gian làm việc rộng hơn, nhưng đôi khi nó cũng là vấn đề bất cập đối với Website của chúng ta.

Đối với một số Website vẫn thường để chiều ngang là auto theo chiều rộng của màn hình thì bây giờ tôi nghĩ rằng cần có một chút thay đổi nhỏ.

Chúng ta chỉ đặt chiều rộng Website của mình auto tới một khoảng nhất định (giả sử rằng tới 1024px,...) thì chúng ta sẽ cố định chiều rộng của nó. Điều đó sẽ giúp bạn chủ động hơn trong việc thiết kế giao diện đồ họa của mình, tránh tình trạng Website sẽ bị vỡ khi chiều ngang của màn hình quá lớn.

Để thực hiện điều đó chúng ta sẽ sử dụng thuộc tính **max-width** trong CSS, nhưng có một vấn đề ở đây đó là các trình duyệt như Firefox, Opera, Safari,.. thì hiểu nhưng riêng IE6 lại không thể hiểu được thuộc tính này.

Để có thể thực hiện điều đó trên IE6 chúng ta có một giải pháp là sử dụng câu lệnh điều kiện của Javascript và nhúng vào CSS:

```
content {
  height: 75px;
  background-color: #000;
  color: #fff;
  width: expression(document.body.clientWidth < 742? "740px" : document.body.clientWidth >
1202? "1200px" : "auto");
  min-width: 740px;
  max-width: 1200px;
}
```

CSS Transparency trên toàn bộ các trình duyệt

Trong bài viết "Hiển thi ảnh PNG trên IE" chúng tôi đã đề cập đến vấn đề làm sao để transparent toàn bộ ảnh PNG trong Website trên IE. Trong bài viết này chúng ta sẽ đề cập đề cập đến vấn đề làm cho transparent background của một thẻ nào đó mà chúng ta muốn. Mặt khác trong kỹ thuật này chúng ta đơn thuần dùng kỹ thuật trong CSS.

Chúng ta sẽ áp dụng các thuộc tính sau cho một phần tử HTML nào đó muốn transparent.

```
.transparent_class {
  filter:alpha(opacity=50);
  -moz-opacity:0.5;
  -khtml-opacity: 0.5;
  opacity: 0.5;
}
```

1. **opacity: 0.5;** Đây là một thuộc tính rất quan trọng bởi nó là một thuộc tính chuẩn của CSS. Nó sẽ làm việc trên hầu hết các phiên bản của Firefox, Opera và Safari. Thuộc tính trên rất cần thiết cho những trình duyệt hỗ trợ các thuộc tính chuẩn của CSS.
2. **filter:alpha(opacity=50);** Thuộc tính trên được dùng cho IE.
3. **-moz-opacity:0.5;** Nó cần thiết cho các phiên bản cũ của Mozilla cũng như Netscape Navigator.
4. **-khtml-opacity: 0.5;** Được dùng cho phiên bản cũ của Safari (1.x).

Border và những điều bạn chưa biết

Giả sử rằng tôi có một thẻ với định dạng CSS như sau:

```
div.border {
  display: block;
  width: 1px;
  height: 1px;
  background: #FFF;
  border-top: 1px solid #F00;
  border-right: 1px solid #0F0;
  border-bottom: 1px solid #00F;
  border-left: 1px solid #000;
  overflow: hidden;
}
```

Các bạn có nghĩ rằng nó sẽ hiển thị giống nhau trên mọi trình duyệt hay không?

Câu trả lời của chúng ta là không, điều này có thể bạn không tin nhưng đó là một thực tế và đã được kiểm trực qua thực nghiệm. Nếu chúng ta nhìn thoáng qua thì sẽ không thấy điều đó, nhưng khi bạn có thể phóng to thẻ nhiều lần thì bạn sẽ thấy điều đó. Dưới đây là một số kết quả thấy được khi xem bằng nhiều trình duyệt khác nhau.

Đôi khi những điều này lại làm chúng ta rất đau đầu trong việc fix các lỗi giữa các trình duyệt. Khi bạn gặp trường hợp thừa hoặc thiếu một pixel trên layout của bạn thì khi đó bạn hãy nhớ tới bài viết này của chúng tôi.

Kỹ thuật đưa footer xuống cuối trang

Khi chúng ta thiết kế layout của trang Web dạng bảng (table) thì việc đưa phần footer lúc nào cũng ở phía dưới là một việc hết sức đơn giản. Nhưng khi bạn làm việc với layout dạng thì việc đưa phần footer lúc nào cũng ở phía dưới là một điều rất khó khăn. Mặc dù như vậy các Designer CSS vẫn nghĩ ra được các thủ thuật (trick) để thực hiện việc đó.

Sau đây chúng ta sẽ đi chi tiết về thủ thuật này:

Trước khi đi vào chi tiết để có một hình dung cụ thể chúng ta cùng xem ví dụ minh họa sau.

Ý tưởng chủ đạo của kỹ thuật này là chúng ta sẽ dùng một thẻ có chiều cao là 100% để đẩy phần footer xuống phía dưới, khi đó thì phần footer sẽ bị đẩy ra ngoài phạm vi của trang và không thể nhìn thấy được.

Để có thể nhìn thấy được phần footer chúng ta sẽ đặt thuộc tính **margin-bottom** của phần wrapper bằng âm với mục đích là giảm chiều cao của phần wrapper một khoảng đúng bằng chiều cao của phần footer. Từ đó chúng ta có thể hoàn toàn nhìn thấy phần footer.

Dựa trên ý tưởng như vậy chúng ta có định dạng HTML như sau:

```
/* BEGIN: WRAPPER */
<div class="wrapper">

  <div class="header">
  <h1>CSS Sticky Footer</h1>
  </div>

  <div class="content">
    Nội dung content...
  </div>

  <div class="push"></div>
</div>
/* END: WRAPPER */

/* BEGIN: FOOTER */
<div class="footer">
  <p>Nội dung footer...</p>
</div>
/* END: FOOTER*/
```

Để cho phần **wrapper** có chiều cao 100% chúng ta định dạng các thuộc tính CSS của nó như sau:

```
.wrapper {
  ...
```

```
  min-height: 100%; /* Fix cho firefox */
  height: auto !important;
  height: 100%;
  ...
}
```

Khi đó phần footer sẽ bị đẩy ra ngoài trang, để cho nó có thể nhìn thấy được ta cần phải đặt **margin-bottom** của phần wrapper bằng âm.

```
.wrapper {
  ...
  margin: 0 auto -4em;
  ...
}
```

Phần margin âm này phải có chiều cao đúng bằng chiều cao của phần footer. Khi đó phần footer sẽ được hiển thị hoàn toàn.

Đặt dòng text vào giữa (theo chiều cao)

Từ trước tới giờ khi bạn muốn đặt một dòng chữ (giả sử như) tiêu đề vào giữa một thẻ HTML có một chiều cao xác định thì bạn làm thế nào? Theo truyền thống dùng <table> thì chúng ta chỉ việc đặt thuộc tính valign="middle" vào trong một thẻ <td> là xong. Nhưng nếu đó không phải là thẻ <td> thì bạn sẽ làm thế nào?

Đề trả lời thắc mắc đó sau đây tôi xin mách nước bạn một thủ thuật nhỏ trong CSS.

Giả sử rằng bạn dùng một thẻ HTML dạng khối (Block Element) có chiều cao xác định height: 100px;. Bây giờ bạn muốn đặt một dòng tiêu đề có nội dung Tin mới nhất vào giữa (theo chiều cao) của thẻ HTML đó.

Bạn có thể dùng thuộc tính **padding** để đẩy dòng chữ đó vào giữa, nhưng đó không phải là giải pháp đúng đắn nhất. Trong trường hợp này thuộc tính line-heigh sẽ là một giả pháp tốt nhất, Chúng ta sẽ dùng thuộc tính **line-height** cho thẻ HTML đó.

Ví dụ

```
div.textcenter {
  margin: 0;
  padding: 0;
  height: 100px;
  line-height: 100px;
  border: 1px solid #CCCCCC;
}
```

Hiển thị ảnh PNG trên IE

Hiện định dạng ảnh PNG chỉ được hỗ trợ bởi một số trình duyệt mới như Firefox, Opera, IE7. Còn một số trình duyệt cũ như từ IE6 trở xuống đều không hỗ trợ định dạng ảnh này. Trong bài viết này chúng ta sẽ dùng một số thủ thuật để định dạng ảnh PNG có thể hiển thị tốt trong IE.

Trong khuôn khổ bài viết chúng ta sẽ không đi sâu vào phân tích nguyên lý cũng như là cách thức để làm cho định dạng PNG hiển thị tốt trên IE. Mà chúng ta chỉ dừng lại ở cách làm để đạt được những điều ở trên.

Trong tài liệu HTML của bạn các ảnh .png vẫn được chèn vào như thông thường.

```
<img scr="test.png" alt="Ảnh ví dụ" title="Ảnh ví dụ" />
```

Để ảnh .png có thể hiển thị tốt khi bạn dùng IE để duyệt thì bạn cần phải chèn file script vào trong tài liệu HTML của bạn. Giả sử rằng file script bạn để ở trong thư mục cùng cấp với tài liệu HTML.

```
<script language="javascript" type="text/javascript" src="PieNG.js">
</script>
</body>
```

Tiếp đó bạn copy ảnh blank.gif vào cùng thư mục với file script.

Style cho thẻ hr

<hr> là một thẻ tự đóng, điều đó có nghĩa là nó không cần thẻ đóng như những thẻ HTML khác. Thẻ <hr> sẽ tạo ra một đường kẻ ngang trên trình duyệt và khoảng cách giữa đường kẻ và nội dung thì phụ thuộc vào các trình duyệt khác nhau.

Tuy nhiên bạn hoàn toàn có thể thay đổi cách hiển thị của thẻ <hr> theo ý mình trên một số trình duyệt. Đối với một số trình duyệt mới (IE6, IE7, Firefox, Opera, Mozilla...) thì cách định dạng của bạn hoàn toàn có thể tương thích.

Trong cách định dạng thẻ <hr> chúng ta cũng cần phải chú ý một chút về cách mà cách trình duyệt xử lí đối với đối tượng thẻ <hr>. Với Internet Explore (IE) sẽ sử dụng thuộc tính color để hiển thị màu của đường kẻ và thuộc tính **background** sẽ không có tác dụng. Tuy nhiên thì Mozilla(Netscape) và Opera thì lại dùng thuộc tính **background** để hiển thị màu của đường kẻ. Chúng ta cũng có thể sử dụng thuộc tính **border** để hiển thị định dạng, nhưng đó không phải là một cách mà chúng ta muốn.

Chúng ta có đoạn mã CSS định dạng thẻ <hr> như sau:

```
hr {
  color: red;
  background: red;
  border: 0;
  height: 1px;
}
```

Thêm khoảng cách giữa đường kẻ và phần nội dung.

Để thêm khoảng cách giữa đường kẻ và nội dung chúng ta sẽ dùng thuộc tính **margin** để quy định khoảng cách này.

```
hr {
  color: red;
  background: red;
  border: 0;
  height: 1px;
  margin: 10px 0 20px; }
```

Kỹ thuật slicing door và ứng dụng

Đối với những desiger thường xuyên làm việc với CSS thì kỹ thuật Sliding Doors không có gì mới mẻ. Nhưng đối với những bạn mới làm quen với CSS thì tôi nghĩ rằng đây là một kỹ thuật mà bạn nên tìm hiểu nó.

Kỹ thuật Sliding Doors(kỹ thuật cửa kéo), là một kỹ thuật được ứng dụng rất nhiều trong việc tạo ra các button, hay các menu ngang kiểu tab,…Trong bài viết này chúng ta sẽ đi tìm hiểu ý tưởng chủ đạo của kỹ thuật và tạo ra một ví dụ nhỏ có sử dụng kỹ thuật trên.

Tôi chắc rằng ít nhất chúng ta đã một lần nhìn thấy chiếc cửa kéo kiểu nhật bản. Một bên cửa cố định và một bên cửa sẽ linh động có thể kéo ra hoặc kéo vào tùy thích. Bản chất của kỹ thuật Sliding Doors cũng giống như vậy.

Chúng ta sẽ chia đối tượng 2 phần(phần bên trái và phần bên phải). Phần bên trái sẽ là phần cố định, còn phần bên phải sẽ là phần động(Việc phân chia này tùy theo ý thích của bạn, bạn có thể phân chia phần bên phải cố định và phần bên trái là phần động). Điều đó cũng có nghĩa là phần bên trái sẽ có kích thước vừa phải, còn phần bên phải cần có kích thước luôn lớn hơn hoặc bằng nội dung mà bạn dự định đưa vào(để đảm bảo rằng đối tượng của chúng ta giống như bị đứt gãy).

Để bạn có một hình dung rõ hơn tôi xin đưa ra hình vẽ mô tả ý tưởng của kỹ thuật này như sau:

Như trên hình vẽ bạn đã thấy, phần bên trái sẽ cố định và phần bên phải sẽ có ra hoặc dãn vào tùy theo nội dung bên trong của đối tượng cần tạo. Do vậy với cách làm như trên thì chúng ta cần tạo phần bên phải cần phải đủ dài để tránh trường hợp bị thiếu ảnh khi nội dung dài.

Các bạn thấy không nghe thì có vẻ ghê gớm nhưng về bản chất thì kỹ thuật này chỉ có vậy thôi, nhưng các bạn đừng coi thường, tuy vậy nhưng nó khá quan trọng và được ứng dụng rất rộng rãi trong việc tạo button, tạo menu kiểu tab,…Để minh họa cho điều tôi vừa trình bày bày sau đây tôi xin đưa ra một ví dụ có sử dụng kỹ thuật này.

Trong ví dụ này chúng ta sẽ tạo ra một số button theo phong cách web 2.0, và có độ tùy biến nội dung cao(chúng ta có thể thay đổi nội dung ở trong một cách tùy ý). Sau đây là chi tiết từng bước làm cụ thể.

1. Tạo nút bằng Photoshop

Việc đầu tiên chúng ta cần làm đó là chúng ta cần tạo ra hình dáng chiếc nút thân yêu của ta, trong ví dụ này tôi tạo ra một nút Sign Up Now! như sau:

2. Cắt nút ra thành hai phần.

Như tôi đã trình bày ở trên, để thực hiện được kỹ thuật này chúng ta cần phải cắt đối tượng ra làm 2 phần. Để có thể cắt hình được chính xác tôi khuyên bạn nên sử dụng công cụ Slice(Slice Tools) trong Adobe Photoshop. Button của chúng ta sẽ được slice như sau:

Chúng ta cần chú ý là phần bên phải cần phải đủ dài để đảm bảo rằng không bị thiếu.

3. Thực hiện viết mã HTML và CSS.

Chúng ta có định dạng mã HTML như sau:

```
<a href=”you link” title=”Your title”>
<span>…Nội dung…</span>
</a>
```

Giải thích:

1. Sở dĩ chúng ta cần phải sử dụng 2 phần thử HTML trong việc này là do đối tượng của chúng ta được chia làm 2 phần, và mỗi phần tử sẽ làm nhiệm vụ giữ một phần của đối tượng.
2. Chúng ta sẽ dùng thẻ <a> bao ở ngoài cùng để đảm bảo rằng toàn bộ nút sẽ link được khi di chuột lên trên nút. Đồng thời thẻ <a> này sẽ chứa phần động bên phải.
3. Thẻ sẽ làm nhiệm vụ giữ phần cố định bên trái, đồng thời sẽ có nhiệm vụ che phần dư của ảnh nền bên phải.

Để thỏa mãn với yêu cầu đặt ra chúng ta có mã định dạng CSS như sau:

```
/* Sign up button style */
a.sign-up {
  font-family: Arial, Helvetica, sans-serif;
  font-size: 25px;
  font-weight: bold;
  color: #FFF;
  text-decoration: none;
  margin: 0;
  padding: 28px 0;
  display: block;
  background: url(images/signup-right-bg.gif) no-repeat right center;
  width: 270px;
}

a.sign-up:hover,
a.sign-up:active {
```

```
  color: #FF0;
}

a.sign-up span {
  margin: 0;
  padding: 28px 0 28px 90px;
  background: url(images/signup-left-bg.gif) no-repeat left center;
}
```

Cách viết giản lược trong CSS

Nếu bạn là người thực sự muốn tìm hiểu về CSS, thì bạn không thể không tìm hiểu cách viết giản lược(shorthand) trong CSS. Cách viết này thực sự mang lại những lợi ích và tiện lợi khi bạn sử dụng CSS.

Trước khi đi vào phân tích những tiện lợi mà nó mang lại, tôi xin lấy một ví dụ đơn giản như sau. Giả sử rằng chúng ta muốn định dạng cho một thẻ có đường viền bao quanh thì chúng ta cần phải viết.

```
border-width: 1px;
border-style: solid;
border-color: #CC0000;
```

Thay vì phải viết như vậy chúng ta chỉ cần viết

```
border: 1px solid #CC0000;
```

Qua ví dụ đơn giản ở trên ta có thể thấy cách viết này mang lại cho chúng ta một số lợi ích sau. Thứ hai

1. Thứ nhất: nó giúp chúng ta giảm thiểu được đáng kể thời gian phải viết mã CSS.
2. Thứ hai: bạn cứ tưởng tượng rằng file CSS của bạn có tới vài ngìn dòng và dung lượng lên tới vài trăm Kb, thì cách viết này còn giúp bạn giảm thiểu được đáng kể dung lượng của file CSS và giúp bạn dễ dàng theo dõi hơn, khi số lượng dòng của trang được giảm xuống.

Sau đây tôi xin đi vào chi tiết một số thuộc tính trong CSs mà chúng ta có thể dùng cách viết giản lược.

1. Thuộc tính Color

Có rất nhiều cách để định nghĩa một màu trong CSS, chúng ta có thể định nghĩa theo hệ số Hexa(trong hệ màu RGB), hoặc chúng ta có thể viết tên màu (ví dụ: white, red…). Nhưng cách định nghĩa theo hệ số Hexa được sử dụng thông dụng nhất. Để định nghĩa theo hệ Hexa chúng ta sẽ đặt dấu (#) ở trước sau đó đến các thông số màu (ví dụ: #003366). Dãy các thông số màu được chia làm 3 phần tương ứng với ba màu Red, Green, Blue (00: Red | 33: Green | 66: Blue). Trong cách định nghĩa hệ số màu ta cũng có cách viết giản lược như sau: **#000000** có thể viết **#000** hoặc **#003366** có thể viết **#036**

2. Thuộc tính margin và padding.

```
margin-top: 10px;
margin-right: 15px;
```

```
margin-bottom: 20px;
margin-left: 25px;
```

Được thay thế bằng

```
margin: 10px 15px 20px 25px; /* top | right | bottom | left */
```

Tương tự với thuộc tính padding

```
padding-top: 10px;
pading-right: 15px;
padding-bottom: 20px;
padding-left: 25px;
padding: 10px 15px 20px 25px; /* top | right | bottom | left */
```

Cả hai thuộc tính margin và padding có hai chú ý như sau: nếu trong trường hợp có hai thông số.

```
margin: 10px 20px; /* top bottom | right left */
padding: 10px 20px; /* top bottom | right left */
```

Thì thông số thứ nhất tương đương với **top** và **bottom** còn thông số thứ hai tương đương với **right** và **left**

Trong trường hợp **margin** và **padding** có 3 thông số:

```
margin: 10px 20px 15px; /* top | right left | bottom */
padding: 10px 20px 15px; /* top | right left | bottom */
```

Thì thông số thứ nhất tương đương với **top**, thông số thứ hai tương đương với **right** và **left**, thông số thứ 3 tương đương với **bottom**

3. Thuộc tính border.

```
border-with: 1px;
border-style: solid;
border-color: #CC0000;
```

Sẽ viết thành

```
border: 1px solid #CC0000; /* width | style | color */
```

4. Thuộc tính background.

```
background-color: #CC0000;
background-image: (image url);
background-repeat: no-repeat; /* repeat-x, repeat-y */
background-position: top left;
```

Tương đương với

```
background: #CC0000 url(image url) no-repeat top left;
```

5. Thuộc tính font

```
font-size: 1em;
line-height: 1.5em;
font-variant:small-caps;
font-weight: bold;
font-style: italic;
font-famyli: Arial, Verdana, Sans-serif;
```

Dạng viết giản lược

```
font: 1em/1.5em bold italic small-caps Arial,Verdana,Sans-serif;
```

5. List type

```
list-style: none;
```

Có nghĩa là

```
list-style-type: none;
```

Bạn cũng có thể sử dụng thuộc tính list-style-position và list-type-image để định dạng cho danh sách không có thứ tự unordered lists, sử dụng hình ảnh cho mỗi dòng và sử dụng list-type-style là hình vuông trong trường hợp không hiển thị được ảnh. Và hai cách viết dưới đây là như nhau.

```
list-style:square inside url(image.gif);
```

là giản lược cho

```
list-style-type: square;
list-style-position: inside;
list-style-image: url(image.gif);
```

6. Outline

Thuộc tính này rất ít dùng vì có rất ít các trình duyệt hiện tại hỗ trợ thuộc tính này, hiện tại chỉ có một số trình duyệt hỗ trợ thuộc tính này Safari, OmniWeb và Opera. Cách viết giản lược các thuộc tính này như sau:

```
outline-color: #000;
outline-style: solid;
outline-with: 2px;
```

Cách viết giản lược sẽ là

```
outline: #000 solid 2px;
```