More About HTML/CSS

By morriswmz

我们太习惯于滥用

1.更多的HTML的标签

1.1 更多的inline元素

标签	语义
<big>	大字号文本
<small>	小字号文本
<i>	斜体文本
<b>	粗体文本
	强调文本（一般会是斜体）
	强调文本（一般会加粗）
<sub>	下标文本
<sup>	上标文本

使用这些标签可以使**语义更明确**，并且**样式表更容易定义**。

~~~~

1.1 更多的inline元素

- 减少span的滥用


```
<h4>
  <small class="color_meeting">【会议】</small>
  <a href="/event/264">传播政治经济学视野下的微博事件</a>
</h4>
```

- 当然最好还是考虑语义，纯粹为了样式也有些不妥

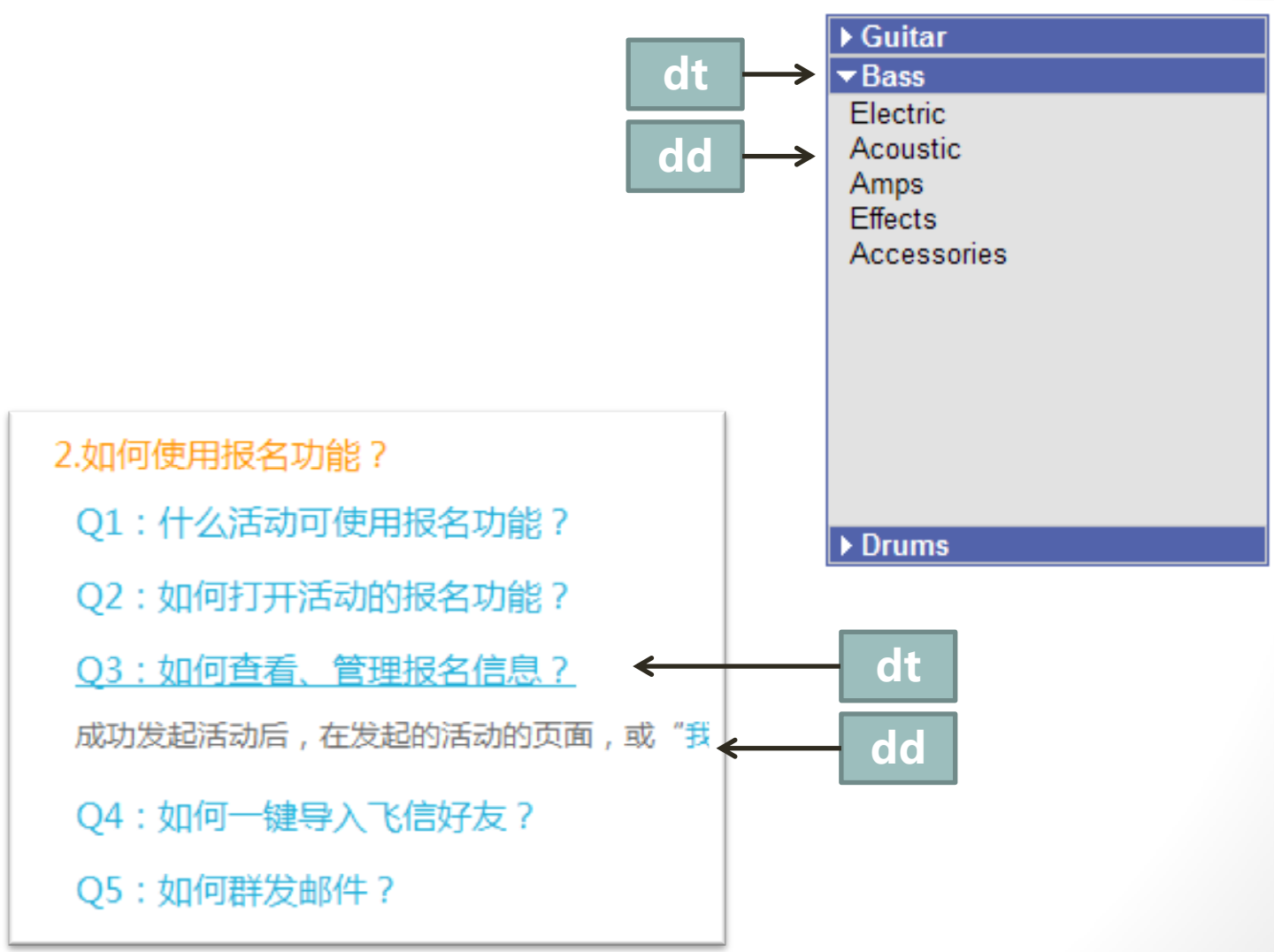
1.2 定义列表<dl>

- 与普通列表不同，子元素成对出现。
- 适用场合：
  - 折叠列表
  - 帮助列表
  - …
  - 标题 – 内容组合列表

```
<dl>
  <dt>Title1</dt>
  <dd>Content1</dd>
  …
  <dt>TitleN</dt>
  <dd>ContentN</dd>
</dl>
```

~~<li class= “title” >ttt
~~
~~<li class= “content” >ccc
~~
~~<li class= “title” >ttt
~~
~~<li class= “content” >ccc
~~

1.2 定义列表<dl>


dt
dd
Guitar
Bass
Electric
Acoustic
Amps
Effects
Accessories
Drums
2.如何使用报名功能？
Q1：什么活动可使用报名功能？
Q2：如何打开活动的报名功能？
Q3：如何查看、管理报名信息？
成功发起活动后，在发起的活动的页面，或“我
Q4：如何一键导入飞信好友？
Q5：如何群发邮件？
dt
dd

1.3 图像热区<map> <area>

- 这两个标签可以定义图像中可以点击的区域（如果你还记得PT站跨年的那张EVE的图的话）

（引自w3school）

```
<img src="planets.jpg" border="0" usemap="#planetmap" alt="Planets" />

<map name="planetmap" id="planetmap">
  <area shape="circle" coords="180,139,14" href="a.html" alt="Venus" />
  <area shape="circle" coords="129,161,10" href ="b.html" alt="Mercury" />
  <area shape="rect" coords="0,0,110,260" href ="c.html" alt="Sun" />
</map>
```

1.4 无js时的一点提示

- 使用<noscript>标签可以放一些提示性信息，这些信息仅在JavaScript失效时才会显示。

```
<noscript>
  您的浏览器禁用了JavaScript，页面功能无法正常工作。
</noscript>
```

层叠是CSS的灵活之处

2. 更灵活的CSS

2.1 id还是class

- 这是个上学期就说过的问题，不过有时候还是挺纠结的
  - 效率　　id > class
  - 灵活性 class > id
- 所以，如果想重复利用某个样式，还是用class的好
- Tip：
  - id的优先级远高于class，所以一个元素用id定义了样式之后是无法通过添加class来覆盖id设定的样式的，只能用style属性覆盖，写js时需要注意这一点。
  - class可以有很多个，这也是其灵活性所在，通过js动态添加/删除class来改变样式是比较推荐的做法，因为这样样式还是定义在css文件中，而不是嵌在了js里，给维护带来麻烦。

2.2 定义一些通用的样式

- body h1~h6 a div 可能会有一些各个页面通用的样式，可以在css开始处定义。
  - 譬如字体、行高，如果大部分情况下是统一的，直接在body中定义样式，而不是分散在其他地方定义
- 一些很常用的样式也可以单独抽出来，比如 btn sprite clearfix 等等，避免重复写大量css代码。
  - 对于按钮来说，对每个按钮都写一遍overflow:hidden之类显然会有大量重复劳动，不如定义一个通用样式，所以按钮都有这个class。

```
.btn {
  display:block;
  overflow:hidden;
  text-indent:-999em;
  border:none;
}
```

但请不要这样：

2.3 灵活地组合选择器

- .class.other 和 .class .other 是不一样的
- 以idea平台的类tab按钮为例：

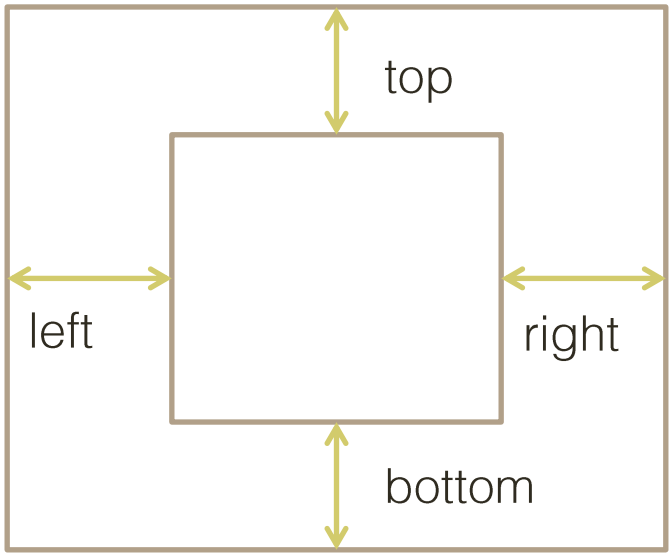

```
.sb_pubtime, .sb_nlike, .sb_njoin {
            ...
}
  .sb_pubtime { ... }
  .sb_pubtime:hover { ... }
  .sb_pubtime:active, .sb_pubtime.current { ... }
  .sb_pubtime.current:hover { ... }
...
```

有时候，必须摆脱文档流的束缚

3 更灵活的排版

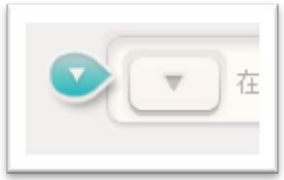
3.1 relative, absolute, fixed

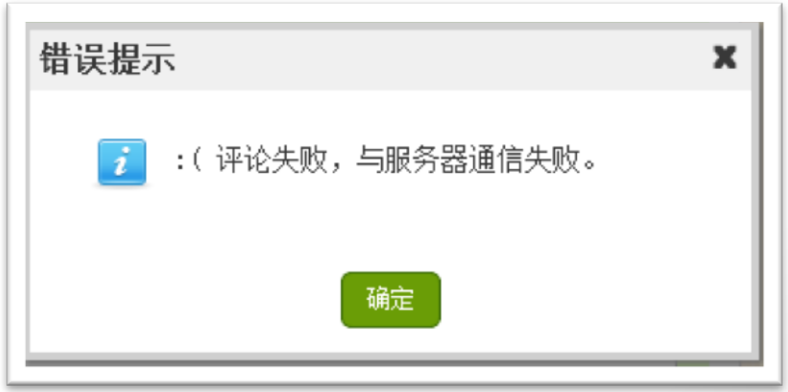
- 这三种方式的排版可以实现更精确，更灵活的排版，不过灵活性相对会差一些。
- 具体请参见上学期所讲Slice & Layout P30-34



3.2 一些脱离文档流的排版

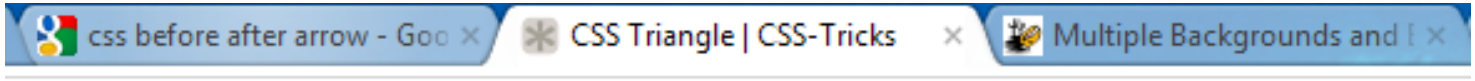
- 浮动的块元素
- 菜单
- 弹出框
- 固定的功能按钮
- 固定的顶栏/底栏
- …





3.3 令人头疼的z-index

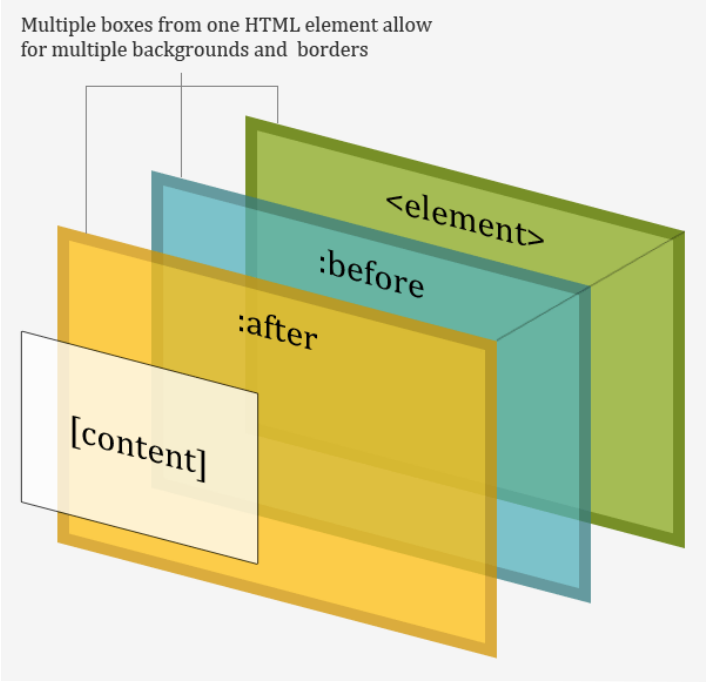
- 虽然大多数情况下没有什么问题，但有时候需要我们人为指定，比如实现tab效果。
  - 若不指定，则按默认顺序层叠：
    - position为relative|absolute|fixed的 > 一般定位的元素
    - 同一级子元素，HTML中在后面的 > 前面的
    - 父元素一般排在其子元素的后面
    - 不再一个容器中的两个元素的层叠顺序比较复杂，需要追溯到同级父元素再比较
      - http://www.cssass.com/blog/index.php/2009/75.html



- 因此，弹出框、功能按钮之类建议放在HTML的最后，最大程度减少对前面排版的干扰。

3.4 :before 和 :after

- 注意：不支持IE6
- 这两个伪类可以在不影响现有HTML结构的情况下在相应元素周围添加内容，可以实现：
  - 无图片参与的小箭头
  - 多边框
  - 多背景
  - 圆角矩形
  - 额外的装饰性元素



http://nicolasgallagher.com/multiple-backgrounds-and-borders-with-css2/

3.4 :before 和 :after

- 最简单的css见右方
- 这样在#element附近便创建了两个块级元素，可以为它们指定背景、边框等等，实现一个块级元素实现不了的效果

- 用border的特性制作小箭头：
  - http://www.yuiblog.com/blog/2010/11/22/css-quick-tip-css-arrows-and-shapes-without-markup/

```
#element:before,
#element:after {
    content:"";
    display:block;
    position:absolute;
    z-index:N;
    top:X;
    left:X;
    right:X;
    bottom:X;
}
```