一、选择器

（一）选择器的类型

1. 类型选择器：用来寻找特定类型的元素，比如段落或者标题元素等，也被称为“元素选择器”。例如：

```
p{color:red;}
div{border:1px solid black;}
```

2. 后代选择器：后代选择器有其他两个选择器之间的空格表示。例如：

```
div p{color:red}
```

3. id选择器

```
#intro{font-weight: 600;}
```

4. 类选择器

```
.date-posted{color: #ccc;}
```

5. 伪类

```
a:link{color: blue}
div:first-child{color: red;}
```

6. 通用选择器：匹配所有可用元素

```
*{padding:0;margin:0}
```

7. 子选择器：子选择器不同于后代选择器的是——子选择器只选择元素的直接后代，而后代选择器选择一个元素的所有后代。例如下面这个例子只把id为nav的直接子元素的padding设置为10,而在li里面如果有嵌套的li元素并不受影响。

```
#nav>li{padding：10px;}
```

8. 相邻同胞选择器：定义同一个父元素下某个元素之后的元素

```
h2 + p{color: #eee;}
```

9. 属性选择器：属性选择器可以分局某个属性是否存在或属性的值来寻找元素。一个属性可以有多个值，值之间用空格分隔。属性选择器允许分局属性值之一寻找元素。例如:

```
p[rel="nofllow"]{}
a[rel]{}
[id="header"]{}等同于#header
```

（二）伪类和伪元素

1. 伪类：用于向某些元素添加特殊效果；

- 常见伪类：:link、:visited、:hover、:focus、:active:、first-child:、lang
- css3对伪类的定义：
    - 伪类存在的意义是为了通过选择器找到那些不存在与DOM树中的信息以及不能被常规CSS选择器获取到的信息。
    - 伪类由一个冒号:开头，冒号后面是伪类的名称和包含在圆括号中的可选参数。
    - 任何常规选择器可以再任何位置使用伪类。伪类语法不区别大小写。一些伪类的作用会互斥，另外一些伪类可以同时被同一个元素使用。并且，为了满足用户在操作DOM时产生的DOM结构改变，伪类也可以是动态的。

2. 伪元素：用于向某些选择器设置特殊效果。

- 常见伪元素：:first-letter、:first-line、:after、:before
- css3对于伪元素的定义：
    - 伪元素在DOM树中创建了一些抽象元素，这些抽象元素是不存在于文档语言里的（可以理解为html源码）。比如：documen接口不提供访问元素内容的第一个字或者第一行的机制，而伪元素可以使开发者可以提取到这些信息。并且，一些伪元素可以使开发者获取到不存在于源文档中的内容（比如常见的::before,::after）。
    - 伪元素的由两个冒号::开头，然后是伪元素的名称。
    - 使用两个冒号::是为了区别伪类和伪元素（CSS2中并没有区别）。当然，考虑到兼容性，CSS2中已存的伪元素仍然可以使用一个冒号:的语法，但是CSS3中新增的伪元素必须使用两个冒号::。
    - 一个选择器只能使用一个伪元素，并且伪元素必须处于选择器语句的最后。

3. 两者比较

- 伪类本质上是为了弥补常规CSS选择器的不足，以便获取到更多信息；
- 伪元素本质上是创建了一个有内容的虚拟容器；
- CSS3中伪类和伪元素的语法不同；
- 可以同时使用多个伪类，而只能同时使用一个伪元素；

> 参考：https://www.cnblogs.com/ihardcoder/p/5294927.html

（三）选择器的层叠和特殊性

寻找一个元素可能有两个或者多种规则，css通过层叠来处理这种冲突，层叠为每个规则分配一个重要度。层叠采用以下重要度次序：

- 标有!important的用户样式
- 标有!important的作者样式
- 作者样式
- 用户样式
- 浏览器/用户代理应用的样式

然后根据选择器的特殊性决定规则的次序。 选择器的特殊性分为四个等级，我们可以认为分别占四位数a、b、c、d(从高位到低位)。

- 如果样式是行内样式，那么a=1
- b等于id选择器的总数
- c等于类、伪类和属性选择器的总数
- d等于类型选择器和伪元素选择器的数量

例如下面这个例子的特殊性为：0，1，0，1,以10为技术的特殊性为101.

```
div#content{}
```

二、规划、组织和维护演示表

（一）链接样式表link和导入样式表@import的区别

1. 尽管现代浏览器把从同一个域下载文件的最大数量从2提升到了8 但是分局最近的浏览器基准测试表明：导入样式表的速度比链接样式表慢。
2. 分隔代码有利于代码的维护，但是也意味着要打开多个样式表，经常在文件之间切换。
3. 具体使用哪一种还要和具体的项目相关

（二）在css中写注释

1. 设计代码结构：

- 最一般的规则放在前面，包括应用于body标记的，应用于由站点上所有元素继承的样式
- 接下来可能需要的所有全局演示样式
- 然后是链接标题和其他元素

2. 注释风格

```
/*@group general styles
-------------------------------------------------------*/
```

3. 自我提示：例如颜色、元素大小等可以以注释的方式写在文件的开头

（三）可以继承的元素：

1. 不可继承的：display、margin、border、padding、background、height、min-height、max-height、width、min-width、max-width、overflow、position、left、right、top、bottom、z-index、float、clear、table-layout、vertical-align、page-break-after、page-bread-before和unicode-bidi。
2. 所有元素可继承：visibility和cursor。
3. 内联元素可继承：letter-spacing、word-spacing、white-space、line-height、color、font、font-family、font-size、font-style、font-variant、font-weight、text-decoration、text-transform、direction。
4. 终端块状元素可继承：text-indent和text-align。
5. 列表元素可继承：list-style、list-style-type、list-style-position、list-style-image。

三、可视化格式模型

（一）盒模型

1. 盒子模型有两种，分别是 IE 盒子模型和标准 W3C 盒子模型
2. 标准 W3C 盒子模型的范围包括 margin、border、padding、content，并且 content 部分不包含其他部分。
3. IE 盒子模型的范围也包括 margin、border、padding、content，和标准 W3C 盒子模型不同的是：IE 盒子模型的 content 部分包含了 border 和 pading。

（二）外边距叠加

1. 当一个元素出现在另一个元素的上面时，的一个元素的底外边距和第二个元素的顶外边距发生叠加。
2. 当一个元素包含在另一个元素中时（假设没有内边距或者边框江外边距分开），他们的顶和底部也会发生叠加
3. 只有普通文档流中块框的垂直边距才会发生外边距叠加。行内框，浮动框可绝对定位框之间的外边距不会叠加。这是因为触发了BFC，BFC内部的元素不会受到外部元素的影响，所以要想元素之间的外边距不叠加，可以想办法触发BFC就可以了。

（三）定位

css中有三种基本的定位机制：普通流、浮动和绝对定位。除非专门指定，否则所有的框都在普通流中定位。

2. 相对定位：

- position的值为relative；
- 相对于元素自身定位，可以设置垂直或者水平位置，让这个元素相对于它的起点移动；
- 相对定位实际上被看做普通流定位模型的一部分，因为他的流是相对于它在普通流中的位置的。

1. 绝对定位：

- position的值为absolute；
- 绝对定位使元素的位置与文档流无关，因此不占据空间，普通文档流的其他元素的布局就像绝对定位的元素不存在一样；
- 绝对定位的位置是相对于距离它最近的那个已定位的祖先元素确定的，如果元素没有已定位的祖先元素，那么他的位置是相对于初始包含块的，根据用户代理的不同，初始包含块可能是画布或者HTML元素。

3. 固定定位

- position的值为fixed；
- 属于绝对定位的一种，差别在于固定元素的包含块是视口。

（四）浮动

1. 浮动脱离文档流和定位脱离文档流的区别：

- 使用 float脱离文档流时，其他盒子会无视这个元素，但是其他盒子内的文本依然会为这个元素让出位置，环绕在周围；所以我们常常看到的文字环绕就是利用浮动实现的。
- 而对于使用position:absolute脱离文档流的元素，其他盒子与其他盒子内的文本都会无视他。
- 定位中只有absolute和fixed脱离文档流，并且当同时设置了position:absolute和float:right，忽略float:right.

2. 清浮动的方法：

- 使用伪元素。

html代码

```
<div>
    <p>1</p>
    <p>2</p>
</div>
```

css代码

```
p{
    float: left;
}
div:after{
    display: block;
    content: "";
    clear: both;
}
```

2. 使用overflow：原理：触发BFC

css代码

```
p{
    float: left;
}
div:after{
    overflow: hidden
```

```
}
```

3. 使用额外标签：在浮动的盒子之下再放一个标签，在这个标签中使用clear:both，来清除浮动对页面的影响.

（五）块级格式化上下文

1. 定义：“块级格式化上下文”，它是一个独立的渲染区域，只有Block-level box参与，它规定了内部的Block-level Box 如何布局，并且与这个区域外部毫不相干。
2. BFC布局规则：

- 内部的Box会在垂直方向，一个接一个的放置；
- Box垂直方向上的距离由margin决定，属于同一个BFC的两个相邻Box的margin会发生重叠；
- 每个元素的margin box的左边，与包含块border box 的左边相接触（对于从左到右的格式化，否则相反）。即时浮动也是如此；
- BFC的区域不会与float box 重叠；
- BFC就是页面上的一个隔离的独立容器，容器里面的子元素不会影响到外面的元素，反之亦如此；
- 计算BFC高度时，浮动元素也参与计算

3. BFC的应用

- 防止垂直margin重叠
- 清除内部浮动
- s实现自适应两栏布局

> 例子：http://www.cnblogs.com/Anita-meng/p/7803234.html

四、链接

（一）普通链接

1. 链接常用的伪类：

- a:link：用来寻找没有被访问过的链接；
- a:visited：用来寻找被访问过的链接；
- a:hover：表示鼠标悬停在链接上；
- a:focus：表示通过键盘移动到链接上；
- a:active：表示链接被激活，也就是当鼠标按下的时候。

  为了避免样式覆盖，通常按照以下次序应用链接样式：a:link a:visited a:hover a:focus a:active

2. 让下划线更有趣

- 使用text-decoration:none去掉链接的下划线，然后在鼠标经过的时候显示下划线；
- 使用图像创建下划线：

```
a:link,a:visited{
    color:#666;
    text-decoration: none;
    background: url(/img/underline.gif) repeat-x left bottom;
}
```

3. 已访问链接的样式：

- 通过在没每个已访问链接的旁边添加一个复选框，就可以创建一种非常简单的已访问链接的样式。

```
a:visited{
    padding-right: 20px;
    background: url(/img/check.gif) no-repeat right middle;
}
```

4. 为练节目标设置样式：假设希望链接到某个页面上的第三个评论，实现的方法是在href的末尾加上一个#字符。为了突出目标元素，我们使用:target伪类为目标元素设置样式。

```
<a href="http://example.com/story/html#commen3">
    so great!
 </a>
```

```
.commen:target{
    background-color: yellow;
}
```

5. 突出显示不同类型的链接

- 外部链接：一个框加一个箭头；
- 邮件链接；
- 突出显示可以下载的文档和提要。

  这些技术有助于改进用户在站点上的浏览体验。通过提醒用户注意离站链接或可下载的文档，让用户明确的了解在单击的时候发生的情况，避免了不必要的回溯操作和烦恼。

（二）类似按钮的链接

```
a{
    display: block;
    line-height: 1.6em;
    width: 6.6em;
```

```
    text-align: center;
    text-decoration: none;
    border: 1px solid #66a300;
    background-color: #8cca12;
    color: #ffffff;
}
```

1. 简单的翻转：在鼠标悬停的时候设置链接的背景和颜色，从而实现简单的翻转效果。
2. 图像翻转：

```
a:link,a:visited {
    width: 203px;
    height: 72px;
    text-indent: -1000em;
    background: url("./static/pictures/back.jpg") no-repeat left top;
}
a:hover,a:focus{
    background-image: url("./static/pictures/back2.jpg");
}
a:active{
    background-image: url("./static/pictures/back3.jpg");
}
```

这种方法在浏览器第一次加载鼠标悬停的时候图像有短暂的延迟，这会造成闪烁效果。

3. Pixy样式的翻转：使用同一个图像并切换他的背景位置。这样减少了服务器的请求数量。

```
a:link,a:visited {
    width: 203px;
    height: 72px;
    text-indent: -1000em;
    background: url("./static/pictures/back1.jpg") no-repeat -203px 0;
}
a:hover,a:focus{
    background-position:  right top;
}
a:active{
    background-position: left bottom;
}
```

4. css精灵：在Pixy样式翻转的基础上，把所有图标甚至站点都包含在一个图像中。许多大型网站都在使用这种技术。可以减少web浏览器的发出的服务器请求，显著加快下载速度。
5. 用css3实现翻转：使用text_shadow,box-shadow,border-radius等属性。

（三）纯css3工具提示：

- 工具提示是当鼠标悬停在具有title属性的元素上时，一些浏览器弹出的黄色小文本框。可以通过使用js和css创建样式独特的自定义工具提示；也可以通过css的定位技术，实现纯css的工具提示。

```
a.tooltip:hover span{
    display: block;
    position: absolute;
    top: 1em;
    left: 2em;
}
```

（四）background

在本章的最后再拓展一下background的知识。

```
background: [<background-color>][,<background-image>][,<background-repeat>][,
<background-attachment>][,<background-position>]
```

1. 设置背景色：background-color

```
background-color: transparent || <color>
```

2. 设置背景图片:background-image

```
background-image: none || <url>
```

3. 设置背景图片重复：background-repeat

```
background-repeat: repeat || repeat-x || repeat-y || no-repeat
```

4. 设置背景图像是否固定或者随着页面的其余部分滚动:background-attachment

```
background-attachment: scroll || fixed
```

5. 设置背景图片的位置background-position

```
background-position: <percentage> || <length> || [left|center|right]
[,top|center|bottom]
```

6. background-size

```
background-size: auto || <length> || <percentage> || cover || contain
```

- auto:原始值，保持背景图片的原始高度和宽度；
- 设置的是具体值，可以改变背景图片的大小；
- 此值为百分值，可以是0％～100％之间任何值，但此值只能应用在块元素上，所设置百分值将使用背景图片大小根据所在元素的宽度的百分比来计算；
- cover：此值是将图片放大，以适合铺满整个容器，这个主要运用在，当图片小于容器时，又无法使用background-repeat来实现时，我们就可以采用cover;将背景图片放大到适合容器的大小，但这种方法会使用背景图片失真；
- contain:此值刚好与cover相反，其主要是将背景图片缩小，以适合铺满整个容器，这个主要运用在，当背景图片大于元素容器时，而又需要将背景图片全部显示出来，此时我们就可以使用contain将图片缩小到适合容器大小为止，这种方法同样会使用图片失真；
- 当background-size取值为和时可以设置两个值，也可以设置一个值，当只取一个值时，第二个值相当于auto，但这里的auto并不会使背景图片的高度保持自己原始高度，而会与第一个值相同。

7. background-origin:用来决定背景图片的定位原点，即背景图片的起点。

```
background-origin: padding || border || content
background-origin: padding-box || border-box || content-box
```

- padding-box(padding):此值为background-origin的默认值，决定background-position起始位置从padding的外边缘（border的内边缘）开始显示背景图片；
- 此值决定background-position起始位置从border的外边缘开始显示背景图片；
- content-box(content):此值决定background-position起始位置从content的外边缘（padding的内边缘）开始显示背景图片。

8. background-clip:用来确定背景的裁剪区域，换句话说，就是如何控制元素背景显示区域。

```
/*Firefox3.6-*/
-moz-background-clip: border || padding;
/*Webkit*/
-webkit-background-clip: border-box || padding-box || context-box;
/*W3C标准 IE9+ and Firefox4.0+*/
background-clip: border-box || padding-box || context-box;
```

background-clip主要是用来控制背景（背景色和背景图片）的显示区域，其主要配合background-origin来制作不同的效果；而background-origin主要是用来控制背景图片的background-position位置，并且其只能控制背景图片。

五、对刘表应用样式和导航条

1. 基本样式列表：通过background-image设置自定义的导航图标

```
li{
    background: url("./img/bullet.gif") no-repeat 0 50%;
    padding: 30px;
}
```

2. 创建基本的垂直导航条

- 去掉默认的符号并将外边距和内边距设置为0；
- 设置不同的背景色和边框色；
- 创建斜面效果：通过把顶边框设置的比背景色浅，底边框设置的比背景色深；
- 设置li的下边框，并且把最后一个li的底边框设置为border:none.

3. 在导航条中突出显示当前页面

- 为主导航中的每个表项分配一个类名或者ID，从而指出用户当前在哪个页面或者部分中；
- 然后在导航列表中的每个项中添加一个对应的ID名或者类型；
- 使用主体ID和列表ID/类的唯一组合在站点中突出显示当前部分和页面。

```
<body id="home">
    <ul class="nav">
        <li><a href="home.html">Home</a></li>
        <li><a href="about.html">About</a></li>
        <li><a href="service.html">Service</a></li>
        <li><a href="concat.html">Concat</a></li>
    </ul>
</body>
```

```
#home .nav .home a,
#home .nav .home a,
#home .nav .home a,
#home .nav .home a{
    background-position: right bottom;
    color: #ffffff;
}
```

3. 创建简单的水平导航条

- 去掉默认的内外边距和列表样式；
- 浮动让列表水平排列
- 设置背景和边框；
- 使用伪类选择器添加前面和后面的双箭头。

4. 创建图形化的导航条：使用图片作为导航栏的背景。

5. 下拉菜单:利用多层列表嵌套和定位实现。
6. css图像映射：使用定位实现鼠标移动到图片的某个位置，显示信息。这个示例可以让我们发挥充分的想象力去扩展。

六、布局

（一）计划布局：

1. 先把页面划分为大的结构区域，比如容器、页眉、内容区域和页脚；
2. 然后开始建立网格结构，根据内容性和差异性设计网格区域；
3. 最后在各个内容区域中寻找不同的布局结构。
4. 设计完结构之后，要关注不同类型的内容；
5. 查看每个内容块的结构，看看不同的类型中是否有共同的模式；
6. 找出模式并确定命名约定之后，开始定义将使用的元素 ###（二）布局(这是我总结的一些居中的方式和常见布局方式)

> http://www.cnblogs.com/Anita-meng/p/7794464.html

> http://www.cnblogs.com/Anita-meng/p/7793856.html

（三）固定宽度、流式和弹性布局

1. 流式布局：在使用流式布局时，尺寸是百分数而不是像素设置的，这是流式布局能够相对于浏览器窗口进行伸缩。

- 首先，将容器宽度设置为浏览器窗口总宽度的百分数。例如：如果设计者使用的宽度是960px，而大多数用户的浏览器窗口设置为1250px，那么使用的百分数是(960/1250)*100=76.8%。
- 然后，以容器的宽度的百分数形式设置主内容区域和次要内容其余的宽度。因为总宽度为注意可以在导航元素和容器之间留出2.**%的隔离带，从而应对任何取整错误和宽度差异。
- 然后需要设置主内容区域中列的宽度，也要留出20px左右的槽。
- 这样就会产生一个最适合1250px窗口的流式布局，但是在屏幕分辨率更大或者更小的时候阅读起来都比较舒服。
- 因为这个布局会恰当的伸缩，所以不需要添加max-width属性，但是，为了确保文本行的长度适合阅读，最好添加以em为单位的max-width。对于比较小的窗口尺寸，这个布局有点挤，所以还可以添加以em为单位的min-width。
- 虽然流式布局可以充分利用空间，但是在大分辨率显示器上，仍然会过长，让用户不舒服，相反，在窄窗口中或者在增加文本字号的时候，行会变得非常短，内容零碎，对于这种问题，弹性布局是一种解决方案。

2. 弹性布局：弹性布局相对于字号来设置元素的宽度。以em为单位设置宽度，可以确保在字号增加的时候整个布局随之扩大。这可以将行长保持在可阅读的范围，对于实力弱或者有认知障碍的人尤其有用。

- 大多数浏览器上的默认字号的16px，10px大约是16px的62.5%,所以在主体上将字母设置为62.5%就好了。
- 只以em单位设置容器的宽度，内部的宽度可以用百分比，这样内部宽度仍然是相对于字号的。

3. 流式和弹性图像：如果选择使用流式或者弹性布局，那么固定宽度的图像就会对设计产生强烈的影响。当布局的宽度减小时，图像会相对于它移动，可能会导致一些消极的影响。比如有些图像会超出包含他

们的元素，从而破坏精心调整过的设计。

- 对于跨越大区域的图像，使用背景图像而不是图片；
- 如果图像用作页面上的图像元素，可以将容器元素的宽度设置为100%并且将overflow属性设置为hidden；
- 对于常规内容图像，如果不希望被裁剪，可以将图像元素添加到没有设置任何尺寸的页面上，然后设置图像的百分比宽度，并且添加与图像宽度相同的max-width以避免图像失真。

七、bug和修复bug

1. 特殊性和分类次序的问题：不要随便添加更特殊的选择器，因为这可能会给代码的其他部分带来特殊性问题，更好的方法往往是删除额外的选择器，让他们尽可能一般化。
2. 外边距叠加问题：可以通过添加一点内边距或者元素背景颜色相同的小边框来修复。
3. 隔离问题：通过隔离问题和识别症状，有可能查明问题的原因并修复它。
4. 创建基本的测试案例
5. 修复问题，而不是修复症状
6. 请求帮助：活动社区等
7. 拥有布局
8. IE条件注释
9. 明确地使用hack和过滤器
10. 应用IE for Mac带通过滤器
11. 应用星号HTML hack
12. 应用子选择器hack

常见bug及修复方法

1. 双外边距浮动bug：这个windows bug时任何浮动元素上的外边距加倍。针对这个问题，只要将元素的display属性设置为inline就可以了；
2. 3像素文本偏移bug：当文本与一个浮动元素相邻时，这个bug就会表现出来。IE5.x的修复办法：元素本身设置高度，迫使元素拥有布局，左外边距设置为0，浮动元素上、下外边距设置为0，左、右外边距设置为-3px。对于IE6，元素本身设置高度，迫使元素拥有布局，浮动元素外边距设置为0。
3. IE6的重复字符bug：最容易最安全的办法是从HTML代码中删除注释。
4. IE6的藏猫猫bug：在某些条件下文本看起来消失了，只有在重新加载页面的时候才再度出现。出现这个问题的条件是：一个浮动元素后面跟着一些非浮动元素，然后是一个清理元素，所有这些元素包含在设置了背景色或者图像的父元素中，如果清理元素碰到了父元素，那么中间的非浮动元素看起来消失了，隐藏到了父元素的背景后面，只有刷新页面才会显示出来。解决这个问题的办法是：避免清理元素和浮动元素接触；另一个方法是给元素指定行高，这个bug也不会出现。最后，将浮动元素和容器的position设置为relative也会减轻这个问题。
5. 相对容器中的绝对定位：可以为容器布局提供一个任意的高度，这会让容器拥有布局。