

米傻的博客

勿在浮沙筑高楼！

CSS面试题

📅 2016-09-28 | 📁 CSS

CSS 中类 (classes) 和 ID 的区别。

- 书写上的差别：class名用"."号开头来定义，id名用"#"号开头来定义；
- 优先级不同（权重不同）
- 调用上的区别：在同一个html网页页面中class是可以被多次调用的（在不同的地方）。而id名作为标签的身份则是唯一的，id在页面中只能出现一次。在js脚本中经常会用到id来修改一个标签的属性
- id作为元素的标签，用于区分不同结构和内容，而class作为一个样式，它可以应用到任何结构和内容上。
- 在布局思路，一般坚持这样的原则：id是先确定页面的结构和内容，然后再为它定义样式；而class相反，它先定义好一类样式，然后再页面中根据需要把类样式应用到不同的元素和内容上面。
- 在实际应用时，class更多的被应用到文字版块以及页面修饰等方面，而id更多地被用来实现宏伟布局和设计包含块，或包含框的样式。

一般原则：类应该应用于概念上相似的元素，这些元素可以出现在同一页面上的多个位置，而ID 应该应用于不同的唯一的元素

请问“resetting”和“normalizing”CSS 之间的区别？你会如何选择，为什么？

Normalize 相对「平和」，注重通用的方案，重置掉该重置的样式，保留有用的 user agent 样式，同时进行一些 bug 的修复，这点是 reset 所缺乏的。

Reset 相对「暴力」，不管你有没有用，统统重置成一样的效果，且影响的范围很大，讲求跨浏览器的一致性。【摘自知乎】

<http://jerryzou.com/posts/aboutNormalizeCss/>

Normalize.css是一种CSS reset的替代方案。它们的区别有：

- Normalize.css 保护了有价值的默认值，Reset通过为几乎所有的元素施加默认样式，强行使得元素有相同的视觉效果。相比之下，Normalize.css保持了许多默认的浏览器样式。这就意味着你不用再为所有公共的排版元素重新设置样式。当一个元素在不同的浏览器中有不同的默认值时，Normalize.css会力求让这些样式保持一致并尽可能与现代标准相符合。
 - Normalize.css 修复了浏览器的bug，它修复了常见的桌面端和移动端浏览器的bug。这往往超出了Reset所能做到的范畴。关于这一点，Normalize.css修复的问题包含了HTML5元素的显示设置、预格式化文字的font-size问题、在IE9中SVG的溢出、许多出现在各浏览器和操作系统中的与表单相关的bug。
 - Normalize.css 不会让你的调试工具变的杂乱
 - Normalize.css 是模块化的
 - Normalize.css 拥有详细的文档
- 选择Normalize.css，主要是reset.css为几乎所有的元素施加默认样式，所以需要对所有公共的排版元素重新设置样式，这是一件很麻烦的工作。

请解释浮动 (Floats) 及其工作原理

浮动出现的最开始出现的意义是用来让**文字环绕**图片而已。

float可以自动包裹元素。

float会导致父容器高度塌陷。float为什么会高度塌陷：元素含有浮动属性 -> 破坏inline box -> 破坏line box高度 -> 没有高度 -> 塌陷。什么时候会塌陷：当标签里面的元素只要样子没有实际高度时会塌陷。

浮动会脱离文档流。产生自己的块级格式化上下文。

描述z-index和叠加上下文是如何形成的。

首先来看看在CSS中叠加上下文形成的原因：

- 负边距

margin为负值时元素会依参考线向外偏移。margin-left/margin-top的参考线为左边的元素/上面的元素（如无兄弟元素则为父元素的左内侧/上内侧），margin-right和margin-bottom的参考线为元素本身的border右侧/border下侧。一般可以利用负边距来就行布局，但没有计算好的话就可能造成元素重叠。堆叠顺序由元素在文档中的先后位置决定，后出现的会在上面。

- position的relative/absolute/fixed定位

当为元素设置position值为relative/absolute/fixed后，元素发生的偏移可能产生重叠，且z-index属性被激活。z-index值可以控制定位元素在垂直于显示屏方向（Z轴）上的堆叠顺序（stack order），值大的元素发生重叠时会在值小的元素上面。

z-index属性：z-index只能在position属性值为relative或absolute或fixed的元素上有效。

基本原理：z-index值可以控制定位元素在垂直于显示屏方向（Z轴）上的堆叠顺序（stack order），值大的元素发生重叠时会在值小的元素上面。

使用相对性：z-index值只决定同一父元素中的同级子元素的堆叠顺序。父元素的z-index值（如果有）为子元素定义了堆叠顺序（css版堆叠“拼爹”）。向上追溯找不到含有z-index值的父元素的情况下，则可以视为自由的z-index元素，它可以与父元素的同级兄弟定位元素或其他自由的定位元素来比较z-index的值，决定其堆叠顺序。同级元素的z-index值如果相同，则堆叠顺序由元素在文档中的先后位置决定，后出现的会在上面。所以如果你发现一个z-index值较大的元素被值较小的元素遮挡了，请先检查它们之间的dom结点关系，多半是因为其父结点含有激活并设置了z-index值的position定位元素

请描述 BFC(Block Formatting Context) 及其如何工作？

BFC:块级格式上下文。定义：

浮动元素和绝对定位元素，非块级盒子的块级容器（例如 inline-blocks, table-cells, 和 table-captions），以及overflow值不为“visible”的块级盒子，都会为他们的内容创建新的块级格式化上下文。

在一个块级格式化上下文里，盒子从包含块的顶端开始垂直地一个接一个地排列，两个盒子之间的垂直的间隙是由他们的margin 值所决定的。两个相邻的块级盒子的垂直外边距会发生叠加。

在块级格式化上下文中，每一个盒子的左外边缘（margin-left）会触碰到容器的左边缘(border-left)（对于从右到左的格式来说，则触碰到右边缘），即使存在浮动也是如此，除非这个盒子创建一个新的块级格式化上下文。

BFC详解：<http://www.cnblogs.com/lhb25/p/inside-block-formatting-ontext.html>

block，inline和inline-block的概念以及区别

display:block

- block元素会独占一行，多个block元素会各自新起一行。默认情况下，block元素宽度自动填满其父元素宽度。
- block元素可以设置width,height属性。块级元素即使设置了宽度,仍然是独占一行。
- block元素可以设置margin和padding属性。

display:inline

- inline元素不会独占一行，多个相邻的行内元素会排列在同一行里，直到一行排列不下，才会新换一行，其宽度随元素的内容而变化。
- inline元素设置width,height属性无效。
- inline元素的margin和padding属性，水平方向的padding-left, padding-right, margin-left, margin-right都产生边距效果；但垂直方向的padding-top, padding-bottom, margin-top, margin-bottom不会产生边距效果。

display:inline-block

就是将对象呈现为inline对象，但是对象的内容作为block对象呈现。之后的内联对象会被排列在同一行内。

备注：属性为inline-block元素之间的空格或者换行在浏览器上会是一个空白的间隙。且IE6和7是不支持这个属性的，需要通过display:inline;zoom:1做hack处理。

列举不同的清除浮动的技巧，并指出它们各自适用的使用场景。

- 添加新的元素、应用 clear：both；

```
<div class="outer">
  <div class="div1">1</div>
  <div class="div2">2</div>
  <div class="div3">3</div>
  <div class="clear"></div>
```

```
</div>
.clear{clear:both; height: 0; line-height: 0; font-size: 0}
```

优点：简单，代码少，浏览器支持好，不容易出现怪问题

缺点是要增加很多无效布局，但这是清除浮动用的比较多的一种方法。

- 父级div定义overflow：auto或者hidden

```
<div class="outer over-flow"> //这里添加了一个class
<div class="div1">1</div>
<div class="div2">2</div>
<div class="div3">3</div>
</div>
.over-flow{
    overflow: auto; zoom: 1; //zoom: 1; 是在处理兼容性问题
}
```

原理：必须定义width或zoom:1，同时不能定义height，使用overflow属性来清除浮动有一点需要注意，overflow属性共有三个属性值：hidden,auto,visible。我们可以使用hiddent和auto值来清除浮动，但切记不能使用visible值，如果使用这个值将无法达到清除浮动效果。

优点：简单，代码少，浏览器支持好

缺点：使用auto时内部宽高超过父级div时，会出现滚动条，使用hidden时会被隐藏

- after 方法

```
<div class="outer">
<div class="div1">1</div>
<div class="div2">2</div>
<div class="div3">3</div>
</div>
.outer {zoom:1;} /*for IE6/7 Maxthon2==/
.outer :after {clear:both;content:'. ';display:block;width: 0;height: 0;visibility:hidden;}
```

其中clear:both;指清除所有浮动；content: ' '; display:block;对于FF/chrome/opera/IE8不能缺少，其中content（' '）可以取值也可以为空。visibility:hidden;的作用是允许浏览器渲染它，但是不显示出来，这样才能实现清除浮动。

所以总的来说，**推荐使用伪类的办法。**

请解释 CSS sprites，以及你要如何在页面或网站中实现它。

CSS Sprites就是把网页中一些背景图片整合到一张图片文件中，再利用CSS的“background-image”，“background-repeat”，“background-position”的组合进行背景定位，background-position可以用数字能精确的定位出背景图片的位置。

优点：当页面加载时，不是加载每个单独图片，而是一次加载整个组合图片。这是一个了不起的改进，它大大减少了HTTP请求的次数，减轻服务器压力，同时缩短了悬停加载图片所需要的时间延迟，使效果更流畅，不会停顿。

缺点：做图像拼合的时候很麻烦。

你最喜欢的图片替换方法是什么，你如何选择使用。

你会如何解决特定浏览器的样式问题？

浏览器的兼容性：

解决方案：

- 主张向前兼容，不考虑向后兼容，
- 根据产品的用户群中各大浏览器，来考虑需要兼容的浏览器
- 把浏览器分两类，一类历史遗留浏览器，一类是现代浏览器，然后根据这个分类开发两个版本的网站，然后自己定义哪些浏览器是历史遗留版本，历史遗留版本浏览器，是用历史遗留界面，通过通告栏告知用户使用现代浏览器，功能更全面，提供好的用户体验
- 直接在用户的浏览器不能兼容的时候，提示用户至少什么版本的IE和火狐谷歌浏览器才能支持（以上方案都失效）
- 项目开始前就得需要确认兼容支持的最低按本是什么，设计一个对应的兼容方案

如何为有功能限制的浏览器提供网页？

有哪些的隐藏内容的方法 (如果同时还要保证屏幕阅读器可用呢)？

`display:none` 文本图片的隐藏

- 缺陷：搜索引擎可能认为被隐藏的文字属于垃圾信息而被忽略
- 屏幕阅读器（是为视觉上有障碍的人设计的读取屏幕内容的程序）会忽略被隐藏的文字，同时不利于搜索引擎。

`visibility: hidden`

缺陷：隐藏的内容会占据他所应该占据物理空间

`overflow: hidden` 隐藏内容或图片

你用过栅格系统 (grid system) 吗？如果使用过，你最喜欢哪种？

Bootstrap中的流式布局；Bootstrap提供了两种布局方式，**固定式布局**和**流式布局**（用em表示的叫做弹性布局，用百分比表示的叫做流体布局）方式，Bootstrap的布局实际上是在栅格外加个容器 (Container)

因此两种布局方式的唯一区别是：

固定布局加的是固定宽度(width)的容器，

流式布局加的是自适应(或叫可变)宽度的容器。

你用过媒体查询，或针对移动端的布局/CSS 吗？

媒体查询规则是开发者能够在相同的样式中，针对不同的媒介来使用不同的样式规则。在CSS2的时候有media Type的规则，通过不同的媒介来切换不同的CSS样式。通过媒体查询的技术可以实现响应式布局，适应不同终端的开发。媒体查询的具体知识请见 CSS3新属性应用文档。

你熟悉 SVG 样式的书写吗？

如何优化网页的打印样式？

添加打印样式

为屏幕显示和打印分别准备一个css文件，如下所示：

- 用于屏幕显示的css：

```
<link rel="stylesheet" href="css/mainstylesheet.css" media="screen" />
```

- 用于打印的css：

```
<link rel="stylesheet" href="css/printstylesheet.css" media="print" />
```

- import方式：

```
<style type="text/css">
    @import url("css/printstylesheet.css") print;
</style>
```

- 直接把屏幕显示样式和打印样式写在一个css文件中：

```
@media print {}{
    h1 {
        color: black;
    }
}
```


```
h2 {{  
  color: gray;  
}}  
}  
}  
  
@media print里面的内容只对打印出来的内容有效，之外的内容就是屏幕显示的样式。
```

◦ 其他：

创建一个不指定媒体类型的样式表通常很有用（或者利用media="all"）。当你准备好定义一些特别用

于打印的规则时，可以只创建一个单独的样式表，使任何在打印时看起来不好的样式都失效。使用这种方法的一个问题是必须确保打印机样式实际上确实覆盖了主样式表。可以使用！important.

但打印样式表也应有些注意事项：

- 打印样式表中最好不要用背景图片，因为打印机不能打印CSS中的背景。如要显示图片，请使用插入到页面中。
 - 最好不要使用像素作为单位，因为打印样式表要打印出来的会是实物，所以建议使用pt和cm。
 - 隐藏掉不必要的内容。（@print div{display:none;}）
 - 打印样式表中最好少用浮动属性，因为它们会消失。
- 如果想要知道打印样式表的效果如何，直接在浏览器上选择打印预览就可以了。

备注：参考：<http://blog.csdn.net/pangni/article/details/6224533>

在书写高效 CSS 时会有哪些问题需要考虑？

- 样式是：从右向左的解析一个选择器
- ID最快，Universal最慢 有四种类型的key selector，解析速度由快到慢依次是：ID、class、tag和universal
- 不要tag-qualify（永远不要这样做 ul#main-navigation{}ID已经是唯一的，不需要Tag来标识，这样做会让选择器变慢。）
- 后代选择器最糟糕（换句话说，下面这个选择器是很低效的：html body ul li a{}）
- 想清楚你为什么这样写
- CSS3的效率问题（CSS3选择器（比如:nth-child）能够漂亮的定位我们想要的元素，又能保证我们的CSS整洁易读。但是这些神奇的选择器会浪费很多的浏览器资源。）
- 我们知道#ID速度是最快的，那么我们都用ID，是不是很快。但是我们不应该为了效率而牺牲可读性和可维护性

使用 CSS 预处理器的优缺点有哪些？

缺点：简单来说CSS预处理器语言较CSS玩法变得更高级了，但同时降低了自己对最终代码的控制力。更致命的是提高了门槛，首先是上手门槛，其次是维护门槛，再来是团队整体水平和规范的门槛。这也造成了初学学习成本的昂贵。

优点：用一种专门的编程语言，为CSS增加了一些编程的特性，将CSS作为目标生成文件，然后开发者就只要使用这种语言进行编码工作。通俗的说，CSS预处理器用一种专门的编程语言，进行Web页面样式设计，然后再编译成正常的CSS文件，以供项目使用。CSS预处理器为CSS增加一些编程的特性，无需考虑浏览器的兼容性问题，例如你可以在CSS中使用变量、简单的逻辑程序、函数等等在编程语言中的一些基本特性，可以让你的CSS更加简洁、适应性更强、可读性更佳，更易于代码的维护等诸多好处。

如果设计中使用了非标准的字体，你该如何去实现？

Webfonts (字体服务例如：Google Webfonts，Typekit 等等。)

请解释浏览器是如何判断元素是否匹配某个 CSS 选择器？

浏览器先产生一个元素集合，这个集合往往由最后一个部分的索引产生（如果没有索引就是所有元素的集合）。然后向上匹配，如果不符合上一个部分，就把元素从集合中删除，直到真个选择器都匹配完，还在集合中的元素就匹配这个选择器了。

请描述伪元素 (pseudo-elements) 及其用途

伪类用于当已有元素处于的某个状态时，为其添加对应的样式，这个状态是根据用户行为而动态变化的。

伪元素用于创建一些不在文档树中的元素，并为其添加样式。

区别：伪类的操作对象是文档树中已有的元素，而伪元素则创建了一个文档数外的元素。因此，伪类与伪元素的区别在于：有没有创建一个文档树之外的元素

参考：<http://www.alloyteam.com/2016/05/summary-of-pseudo-classes-and-pseudo-elements/>

请解释你对盒模型的理解，以及如何在 CSS 中告诉浏览器使用不同的盒模型来渲染你的布局？

盒子模型分为两类：W3C标准盒子模型和IE盒子模型

这两者的关键区别就在于：

- 宽高的计算：W3C盒子模型——属性高（height）和属性宽（width）这两个值不包含填充（padding）和边框（border）
- IE盒子模型——属性高（height）和属性宽（width）这两个值包含填充（padding）和边框（border）

各浏览器盒模型的组成结构是一致的，区别只是在“怪异模式”下宽度和高度的计算方式，而“标准模式”下则没有区别。

组成结构以宽度为例：总宽度=marginLeft+borderLeft+paddingLeft+contentWidth+paddingRight+borderRight+marginRight（W3C标准盒子模型）。页面在“怪异模式”下，css中为元素的width和height设置的值在标准浏览器和ie系列（ie9除外）里的代表的含义是不同的（IE盒子模型）。

因而解决兼容型为题最简洁和值得推荐的方式是：下述的第一条。

- 将页面设为“标准模式”。添加对应的dtd标识，如：<!DOCTYPE html>
- 使用hack或者在外面套上一层wrapper。前提是页面处于“怪异模式”，“标准模式”不存在兼容性问题。

```
1、hack的方式
#box {
width:100px !important; // ie9,ff,chrome,opera这样的标准浏览器
width:160px; //所有的浏览器；它的本意是只对不认识!important的设置。可是ie7、ie8也认识
+width:160px!important;//ie7
width:160px/0!important;//ie8
padding:0 10px;border:20px solid blue;margin:70px;
}

2、wrapper
#box {
width:100px;
margin:70px;
float:left;
}
.wrapper {
padding:0 10px;border:20px solid blue;
}
```

总结：使用“标准模式”即可实现兼容，不兼容只发生在“怪异模式”下。而且正常的页面基本上都选择前者，如果选择后者，麻烦不止于此，一些css技巧也将失灵，如将div居中：div{margin:0 auto;}

请解释*{ box-sizing: border-box; }的作用,并且说明使用它有什么好处？

设置他以后，相当于以怪异模式解析，border和padding全会在你设置的宽度内部，比如手机端设置两行并且的布局，宽度各为50%。如果不用这个属性，设置border后右边的div会下来错位，设置这个属性，宽度还是50%而不是50%+*px,两行可以并列显示

说到IE的bug，在IE6以前的版本中，IE对盒模型的解析出现一些问题，跟其它浏览器不同，将border与padding都包含在width之内。而另外一些浏览器则与它相反，是不包括border和padding的。对于IE浏览器，当我们设置box-sizing: content-box;时，浏览器对盒模型的解释遵从我们之前认识到的W3C标准，当它定义width和height时，它的宽度不包括border和padding；对于非IE浏览器，当我们设置box-sizing: border-box;时，浏览器对盒模型的解释与IE6之前的版本相同，当它定义width和height时，border和padding则是被包含在宽高之内的。内容的宽和高可以通过定义的“width”和“height”减去相应方向的“padding”和“border”的宽度得到。内容的宽和高必须保证不能为负，必要时将自动增大该元素border box的尺寸以使其内容的宽或高最小为0。

好处：

- 使用*{ box-sizing: border-box; }能够统一IE和非IE浏览器之间的差异。

- 解决排版的问题，每个盒子之间排版时不用考虑padding和border的宽度计算

请罗列出你所知道的 display 属性的全部值？

display 属性规定元素应该生成的框的类型。

值	描述
none	此元素不会被显示。
block	此元素将显示为块级元素，此元素前后会带有换行符。
inline	默认。此元素会被显示为内联元素，元素前后没有换行符。
inline-block	行内块元素。（CSS2.1 新增的值）
list-item	此元素会作为列表显示。
run-in	此元素会根据上下文作为块级元素或内联元素显示。
compact	CSS 中有值 compact，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。
marker	CSS 中有值 marker，不过由于缺乏广泛支持，已经从 CSS2.1 中删除。
table	此元素会作为块级表格来显示（类似 <table>），表格前后带有换行符。
inline-table	此元素会作为内联表格来显示（类似 <table>），表格前后没有换行符。

请解释 inline 和 inline-block 的区别？

都是display 属性规定元素应该生成的框的类型。但是block代表块级元素，元素前后都有换行符；inline是默认的样式，表示该元素被显示为内联元素，元素前后没有换行符号。也就是说，block元素通常被现实为独立的一块，会单独换一行；inline元素则前后不会产生换行，一系列inline元素都在一行内显示，直到该行排满。而inline-block代表行内块元素（css2.0新增）。

display:block

- block元素会独占一行，多个block元素会各自新起一行。默认情况下，block元素宽度自动填满其父元素宽度。
- block元素可以设置width,height属性。块级元素即使设置了宽度,仍然是独占一行。
- block元素可以设置margin和padding属性。

display:inline

- inline元素不会独占一行，多个相邻的行内元素会排列在同一行里，直到一行排列不下，才会新换一行，其宽度随元素的内容而变化。
- inline元素设置width,height属性无效。
- inline元素的margin和padding属性，水平方向的padding-left, padding-right, margin-left, margin-right都产生边距效果；但竖直方向的padding-top, padding-bottom, margin-top, margin-bottom不会产生边距效果。

display:inline-block

简单来说就是将对象呈现为inline对象，但是对象的内容作为block对象呈现。之后的内联对象会被排列在同一行内。比如我们可以给一个link（a元素）inline-block属性值，使其既具有block的宽度高度特性又具有inline的同行特性。

请解释 relative、fixed、absolute 和 static 元素的区别？

各个属性的值：

- static：默认值。没有定位，元素出现在正常的流中（忽略 top, bottom, left, right 或者 z-index 声明）。
- relative：生成相对定位的元素，通过top,bottom,left,right的设置相对于其正常位置进行定位。可通过z-index进行层次分级。
- absolute：生成绝对定位的元素，相对于 static 定位以外的第一个父元素进行定位。元素的位置通过“left”,“top”,“right”以及“bottom”属性进行规定。可通过z-index进行层次分级。
- fixed：生成绝对定位的元素，相对于浏览器窗口进行定位。元素的位置通过“left”,“top”,“right”以及“bottom”属性进行规定。可通过z-index进行层次分级。

relative和absolute进行对比分析：

- relative。定位为relative的元素脱离正常的文本流中，但其在文本流中的位置依然存在。
- absolute。定位为absolute的层脱离正常文本流，但与relative的区别是其在正常流中的位置不存在。
- fixed:定位为绝对定位，脱离正常文本流，相对于浏览器窗口进行定位

relative和absolute与fixed进行对比分析：

- relative定位的层总是相对于其最近的父元素，无论其父元素是何种定位方式。
- absolute定位的层总是相对于其最近的定义为absolute或relative的父层，而这个父层并不一定是其直接父层。如果其父层中都未定义absolute或relative，则其将相对body进行定位，
- fixed：生成绝对定位的元素，相对于浏览器窗口进行定位。

CSS中字母‘C’的意思是叠层 (Cascading)。请问在确定样式的过程中优先级是如何决定的 (请举例)？如何有效使用此系统？

CSS面试题

JS面试题 >

© 2016 ♥ 米傻

由 [Hexo](#) 强力驱动 | 主题 - [NexT.Pisces](#)

