本部分主要是笔者在复习 CSS 相关知识和一些相关面试题时所做的笔记，如果出现错误，希望大家指出！

目录

1.介绍一下标准的 CSS 的盒子模型？低版本 IE 的盒子模型有什么不同的?

相关知识点:

（1）有两种盒子模型：IE盒模型（border-box）、W3C标准盒模型（content-box）
（2）盒模型：分为内容（content）、填充（padding）、边界（margin）、边框（border）四个部分

IE盒模型和W3C标准盒模型的区别：

（1）W3C标准盒模型：属性width，height只包含内容content，不包含border和padding
（2）IE盒模型：属性width，height包含content、border和padding，指的是content
+padding+border。

在ie8+浏览器中使用哪个盒模型可以由box-sizing（CSS新增的属性）控制，默认值为content-box，即标准盒模型；
如果将box-sizing设为border-box则用的是IE盒模型。如果在ie6，7，8中DOCTYPE缺失会将盒子模型解释为IE
盒子模型。若在页面中声明了DOCTYPE类型，所有的浏览器都会把盒模型解释为W3C盒模型。

回答:

盒模型都是由四个部分组成的，分别是margin、border、padding和content。

标准盒模型和IE盒模型的区别在于设置width和height时，所对应的范围不同。标准盒模型的width和height属性的
范围只包含了content，而IE盒模型的width和height属性的范围包含了border、padding和content。

一般来说，我们可以通过修改元素的box-sizing属性来改变元素的盒模型。

详细的资料可以参考:
《CSS 盒模型详解》

2.CSS 选择器有哪些?

（1）id选择器（#myid）
（2）类选择器（.myclassname）
（3）标签选择器（div,h1,p）
（4）后代选择器（h1 p）
（5）子选择器（ul>li）
（6）兄弟选择器（所有）（li~a）
（7）相邻兄弟选择器（li+a）
（8）属性选择器（a[rel="external"]）
（9）伪类选择器（a:hover,li:nth-child）
（10）伪元素选择器（::before、::after）
（11）通配符选择器（*）

3.::before 和:after 中双冒号和单冒号有什么区别？解释一下这 2 个伪元素的作用。

相关知识点：

单冒号（:）用于CSS3伪类，双冒号（::）用于CSS3伪元素。（伪元素由双冒号和伪元素名称组成）
双冒号是在当前规范中引入的，用于区分伪类和伪元素。不过浏览器需要同时支持旧的已经存在的伪元素写法，
比如:first-line、:first-letter、:before、:after等，
而新的在CSS3中引入的伪元素则不允许再支持旧的单冒号的写法。

想让插入的内容出现在其它内容前，使用::before，否者，使用::after；
在代码顺序上，::after生成的内容也比::before生成的内容靠后。
如果按堆栈视角，::after生成的内容会在::before生成的内容之上。

回答：

在css3中使用单冒号来表示伪类，用双冒号来表示伪元素。但是为了兼容已有的伪元素的写法，在一些浏览器中也可以使用单冒号
来表示伪元素。

伪类一般匹配的是元素的一些特殊状态，如hover、link等，而伪元素一般匹配的特殊的位置，比如after、before等。

4.伪类与伪元素的区别

css引入伪类和伪元素概念是为了格式化文档树以外的信息。也就是说，伪类和伪元素是用来修饰不在文档树中的部分，比如，一句
话中的第一个字母，或者是列表中的第一个元素。

伪类用于当已有的元素处于某个状态时，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。比如说，当用户悬停在指定的
元素时，我们可以通过:hover来描述这个元素的状态。

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。它们允许我们为元素的某些部分设置样式。比如说，我们可以通过::be
fore来在一个元素前增加一些文本，并为这些文本添加样式。虽然用户可以看到这些文本，但是这些文本实际上不在文档树中。

有时你会发现伪元素使用了两个冒号（::）而不是一个冒号（:）。这是CSS3的一部分，并尝试区分伪类和伪元素。大多数浏览
器都支持这两个值。按照规则应该使用（::）而不是（:），从而区分伪类和伪元素。但是，由于在旧版本的W3C规范并未对此进行
特别区分，因此目前绝大多数的浏览器都支持使用这两种方式表示伪元素。

详细资料可以参考：
《总结伪类与伪元素》

5.CSS 中哪些属性可以继承?

相关资料：

每个CSS属性定义的概述都指出了这个属性是默认继承的，还是默认不继承的。这决定了当你没有为元素的属性指定值时该如何计算
值。

当元素的一个继承属性没有指定值时，则取父元素的同属性的计算值。只有文档根元素取该属性的概述中给定的初始值（这里的意思应
该是在该属性本身的定义中的默认值）。

当元素的一个非继承属性（在Mozilla code里有时称之为reset property）没有指定值时，则取属性的初始值initial v
alue（该值在该属性的概述里被指定）。

有继承性的属性：

（1）字体系列属性
font、font-family、font-weight、font-size、font-style、font-variant、font-stretch、font-size-adjust

（2）文本系列属性
text-indent、text-align、text-shadow、line-height、word-spacing、letter-spacing、text-transform、direction、color

（3）表格布局属性
caption-side border-collapse empty-cells

（4）列表属性
list-style-type、list-style-image、list-style-position、list-style

（5）光标属性
cursor

（6）元素可见性
visibility

（7）还有一些不常用的：speak，page，设置嵌套引用的引号类型quotes等属性

注意：当一个属性不是继承属性时，可以使用inherit关键字指定一个属性应从父元素继承它的值，inherit关键字用于显式地
指定继承性，可用于任何继承性/非继承性属性。

回答：

每一个属性在定义中都给出了这个属性是否具有继承性，一个具有继承性的属性会在没有指定值的时候，会使用父元素的同属性的值
来作为自己的值。

一般具有继承性的属性有，字体相关的属性，font-size和font-weight等。文本相关的属性，color和text-align等。
表格的一些布局属性、列表属性如list-style等。还有光标属性cursor、元素可见性visibility。

当一个属性不是继承属性的时候，我们也可以通过将它的值设置为inherit来使它从父元素那获取同名的属性值来继承。

详细的资料可以参考：
《继承属性》
《CSS 有哪些属性可以继承？》

6.CSS 优先级算法如何计算?

相关知识点：

CSS的优先级是根据样式声明的特殊性值来判断的。

选择器的特殊性值分为四个等级，如下：

（1）内联样式选择器x,0,0,0
（2）ID选择器0,x,0,0
（3）class选择器/属性选择器/伪类选择器 0,0,x,0
（4）元素和伪元素选择器0,0,0,x

计算方法：

（1）每个等级的初始值为0
（2）每个等级的叠加为选择器出现的次数相加
（3）不可进位，比如0,99,99,99
（4）依次表示为：0,0,0,0
（5）每个等级计数之间没关联
（6）等级判断从左向右，如果某一位数值相同，则判断下一位数值
（7）如果两个优先级相同，则最后出现的优先级高，!important也适用
（8）通配符选择器的特殊性值为：0,0,0,0
（9）继承样式优先级最低，通配符样式优先级高于继承样式
（10）!important（权重），它没有特殊性值，但它的优先级是最高的，为了方便记忆，可以认为它的特殊性值为1,0,0,0,0。

计算实例：

```
（1）#demo a{color: orange;}/*特殊性值：0,1,0,1*/
（2）div#demo a{color: red;}/*特殊性值：0,1,0,2*/
```

注意：
（1）样式应用时，css会先查看规则的权重（!important），加了权重的优先级最高，当权重相同的时候，会比较规则的特殊性。

（2）特殊性值越大的声明优先级越高。

（3）相同特殊性值的声明，根据样式引入的顺序，后声明的规则优先级高（距离元素出现最近的）

（4）部分浏览器由于字节溢出问题出现的进位表现不做考虑

回答：

判断优先级时，首先我们会判断一条属性声明是否有权重，也就是是否在声明后面加上了!important。一条声明如果加上了权重，
那么它的优先级就是最高的，前提是它之后不再出现相同权重的声明。如果权重相同，我们则需要去比较匹配规则的特殊性。

一条匹配规则一般由多个选择器组成，一条规则的特殊性由组成它的选择器的特殊性累加而成。选择器的特殊性可以分为四个等级，
第一个等级是行内样式，为1000，第二个等级是id选择器，为0100，第三个等级是类选择器、伪类选择器和属性选择器，为0010，
第四个等级是元素选择器和伪元素选择器，为0001。规则中每出现一个选择器，就将它的特殊性进行叠加，这个叠加只限于对应的等
级的叠加，不会产生进位。选择器特殊性值的比较是从左向右排序的，也就是说以1开头的特殊性值比所有以0开头的特殊性值要大。
比如说特殊性值为1000的的规则优先级就要比特殊性值为0999的规则高。如果两个规则的特殊性值相等的时候，那么就会根据它们引
入的顺序，后出现的规则的优先级最高。

对于组合声明的特殊性值计算可以参考：
《CSS 优先级计算及应用》
《CSS 优先级计算规则》
《有趣：256 个 class 选择器可以干掉 1 个 id 选择器》

7.关于伪类 LVHA 的解释?

a标签有四种状态：链接访问前、链接访问后、鼠标滑过、激活，分别对应四种伪
类:link、:visited、:hover、:active；

当链接未访问过时：

（1）当鼠标滑过a链接时，满足:link和:hover两种状态，要改变a标签的颜色，就必须将:hover伪类
在:link伪
类后面声明；
（2）当鼠标点击激活a链接时，同时满足:link、:hover、:active三种状态，要显示a标签激活时的样式
（:active），
必须将:active声明放到:link和:hover之后。因此得出LVHA这个顺序。

当链接访问过时，情况基本同上，只不过需要将:link换成:visited。

这个顺序能不能变？可以，但也只有:link和:visited可以交换位置，因为一个链接要么访问过要么没访问
过，不可能同时满足，
也就不存在覆盖的问题。

8.CSS3 新增伪类有那些?

（1）elem:nth-child(n)选中父元素下的第n个子元素，并且这个子元素的标签名为elem，n可以接受具体
的数
值，也可以接受函数。

（2）elem:nth-last-child(n)作用同上，不过是从后开始查找。

（3）elem:last-child选中最后一个子元素。

（4）elem:only-child如果elem是父元素下唯一的子元素，则选中之。

（5）elem:nth-of-type(n)选中父元素下第n个elem类型元素，n可以接受具体的数值，也可以接受函数。

（6）elem:first-of-type选中父元素下第一个elem类型元素。

（7）elem:last-of-type选中父元素下最后一个elem类型元素。

（8）elem:only-of-type如果父元素下的子元素只有一个elem类型元素，则选中该元素。

（9）elem:empty选中不包含子元素和内容的elem类型元素。

（10）elem:target选择当前活动的elem元素。

（11）:not(elem)选择非elem元素的每个元素。

（12）:enabled 控制表单控件的禁用状态。

（13）:disabled 控制表单控件的禁用状态。

(14):checked单选框或复选框被选中。

详细的资料可以参考：
《CSS3 新特性总结(伪类)》
《浅谈 CSS 伪类和伪元素及 CSS3 新增伪类》

9.如何居中 div?

-水平居中：给 div 设置一个宽度，然后添加 margin:0 auto 属性

```
div {
  width: 200px;
  margin: 0 auto;
}
```

-水平居中，利用 text-align:center 实现

```
.container {
  background: rgba(0, 0, 0, 0.5);
  text-align: center;
  font-size: 0;
}

.box {
  display: inline-block;
  width: 500px;
  height: 400px;
  background-color: pink;
}
```

-让绝对定位的 div 居中

```
div {
  position: absolute;
  width: 300px;
  height: 300px;
  margin: auto;
  top: 0;
  left: 0;
  bottom: 0;
  right: 0;
  background-color: pink; /*方便看效果*/
}
```

-水平垂直居中一

```
/*确定容器的宽高宽500高300的层设置层的外边距div{*/
position: absolute;/*绝对定位*/
width: 500px;
height: 300px;
top: 50%;
left: 50%;
margin: -150px 0 0 -250px;/*外边距为自身宽高的一半*/
background-color: pink;/*方便看效果*/
}
```

-水平垂直居中二

```
/*未知容器的宽高，利用`transform`属性*/
div {
  position: absolute; /*相对定位或绝对定位均可*/
  width: 500px;
  height: 300px;
  top: 50%;
  left: 50%;
  transform: translate(-50%, -50%);
  background-color: pink; /*方便看效果*/
}
```

-水平垂直居中三

```
/*利用flex布局实际使用时应考虑兼容性*/
.container {
  display: flex;
  align-items: center; /*垂直居中*/
  justify-content: center; /*水平居中*/
}
.containerdiv {
  width: 100px;
  height: 100px;
  background-color: pink; /*方便看效果*/
}
```

-水平垂直居中四

```
/*利用text-align:center和vertical-align:middle属性*/
.container {
  position: fixed;
  top: 0;
  right: 0;
  bottom: 0;
  left: 0;
  background: rgba(0, 0, 0, 0.5);
  text-align: center;
  font-size: 0;
  white-space: nowrap;
  overflow: auto;
}

.container::after {
  content: '';
  display: inline-block;
  height: 100%;
  vertical-align: middle;
}

.box {
  display: inline-block;
  width: 500px;
  height: 400px;
  background-color: pink;
  white-space: normal;
  vertical-align: middle;
}
```

回答：

一般常见的几种居中的方法有：

对于宽高固定的元素

（1）我们可以利用margin:0 auto来实现元素的水平居中。

（2）利用绝对定位，设置四个方向的值都为0，并将margin设置为auto，由于宽高固定，因此对应方向实现平分，可以实现水
平和垂直方向上的居中。

（3）利用绝对定位，先将元素的左上角通过top:50%和left:50%定位到页面的中心，然后再通过margin负值来调整元素
的中心点到页面的中心。

（4）利用绝对定位，先将元素的左上角通过top:50%和left:50%定位到页面的中心，然后再通过translate来调整元素
的中心点到页面的中心。

（5）使用flex布局，通过align-items:center和justify-content:center设置容器的垂直和水平方向上为居中对
齐，然后它的子元素也可以实现垂直和水平的居中。

对于宽高不定的元素，上面的后面两种方法，可以实现元素的垂直和水平的居中。

10.display 有哪些值？说明他们的作用。

block　块类型。默认宽度为父元素宽度，可设置宽高，换行显示。
none　元素不显示，并从文档流中移除。
inline　行内元素类型。默认宽度为内容宽度，不可设置宽高，同行显示。
inline-block 默认宽度为内容宽度，可以设置宽高，同行显示。
list-item　像块类型元素一样显示，并添加样式列表标记。
table　此元素会作为块级表格来显示。
inherit 规定应该从父元素继承display属性的值。

详细资料可以参考：
《CSS display 属性》

11.position 的值 relative 和 absolute 定位原点是?

相关知识点：

absolute
生成绝对定位的元素，相对于值不为static的第一个父元素的padding box进行定位，也可以理解为离自己这一级元素最近的
一级position设置为absolute或者relative的父元素的padding box的左上角为原点的。

fixed（老IE不支持）
生成固定定位的元素，相对于浏览器窗口进行定位。

relative
生成相对定位的元素，相对于其元素本身所在文档流中的位置进行定位。

static

默认值。没有定位，元素出现在正常的流中（忽略top,bottom,left,right,z-index声明）。

inherit
规定从父元素继承position属性的值。

回答:

relative定位的元素，是相对于元素本身所在文档流中的位置进行定位的。

absolute定位的元素，是相对于它的第一个position值不为static的祖先元素的padding box来进行定位的。这句话
我们可以这样来理解，我们首先需要找到绝对定位元素的一个position的值不为static的祖先元素，然后相对于这个祖先元
素的padding box来定位，也就是说在计算定位距离的时候，padding的值也要算进去。

12.CSS3 有哪些新特性？（根据项目回答）

新增各种CSS选择器 （:not(.input)：所有class不是“input”的节点）
圆角 （border-radius:8px）
多列布局 （multi-column layout）
阴影和反射 （Shadow\Reflect）
文字特效 （text-shadow）
文本修饰 （Text-decoration）
线性渐变 （linear-gradient）
旋转 （transform：rotate）
缩放，平移，倾斜，动画，多背景
例如：transform:\scale(0.85,0.90)\translate(0px,-30px)\skew(-9deg,0deg)\Animation:

13.请解释一下 CSS3 的 Flex box（弹性盒布局模型），以及适用场景?

相关知识点:

Flex是FlexibleBox的缩写，意为"弹性布局"，用来为盒状模型提供最大的灵活性。

任何一个容器都可以指定为Flex布局。行内元素也可以使用Flex布局。注意，设为Flex布局以后，子元素的float、cl
ear和vertical-align属性将失效。

采用Flex布局的元素，称为Flex容器（flex container），简称"容器"。它的所有子元素自动成为容器成员，称为Flex
元素（flex item），简称"元素"。

容器默认存在两根轴：水平的主轴（main axis）和垂直的辅轴（cross axis），元素默认沿主轴排列。

以下6个属性设置在容器上。

flex-direction属性决定主轴的方向（即元素的排列方向）。

flex-wrap属性定义，如果一条轴线排不下，如何换行。

flex-flow属性是flex-direction属性和flex-wrap属性的简写形式，默认值为row nowrap。

justify-content属性定义了元素在主轴上的对齐方式。 可选值：flex-start, flex-end, center, space-around,
space-between, stretch

align-items属性定义元素在辅轴上如何对齐。 可选值：flex-start, flex-end, center, stretch

align-content属性定义了多根轴线的对齐方式。如果元素只有一根轴线，该属性不起作用。

以下6个属性设置在元素上。

order属性定义元素的排列顺序。数值越小，排列越靠前，默认为0。

flex-grow属性定义元素的增长系数，默认为0，即如果存在剩余空间，也不放大。

flex-shrink属性定义了项目的缩减系数，默认为1，即如果空间不足，该项目将缩小。

flex-basis属性定义了在分配多余空间之前，元素占据的主轴空间。浏览器根据这个属性，计算主轴是否有多余空间。它的默认
值为auto，即项目的本来大小。

flex属性是flex-grow，flex-shrink和flex-basis的简写，默认值为0 1 auto。

align-self属性允许单个元素有与其他元素不一样的对齐方式，可覆盖align-items属性。默认值为auto，表示继承父
元素的align-items属性，如果没有父元素，则等同于stretch。

回答:

flex布局是CSS3新增的一种布局方式，我们可以通过将一个元素的display属性值设置为flex从而使它成为一个flex
容器，它的所有子元素都会成为它的弹性元素。

一个容器默认有两条轴，一个是水平的主轴，一个是与主轴垂直的辅轴。我们可以使用flex-direction来指定主轴的方向。
我们可以使用justify-content来指定元素在主轴上的排列方式，使用align-items来指定元素在辅轴上的排列方式。还
可以使用flex-wrap来规定当一行排列不下时的换行方式。

对于容器中的弹性元素，我们可以使用order属性来指定元素的排列顺序，还可以使用flex-grow来指定当排列空间有剩余的时候，
元素的增长系数。还可以使用flex-shrink来指定当排列空间不足时，元素的缩减系数。

详细资料可以参考:
《Flex 布局教程：语法篇》
《Flex 布局教程：实例篇》

14.用纯 CSS 创建一个三角形的原理是什么?

采用的是相邻边框连接处的均分原理。
将元素的宽高设为0，只设置
border，把任意三条边隐藏掉（颜色设为
transparent），剩下的就是一个三角形。

```
#demo {
width: 0;
height: 0;
border-width: 20px;
border-style: solid;
border-color: transparent transparent red transparent;
border-top: none;    // 加上这一行可以让三角形顶格，  不加三角形会下移20px
```

15.一个满屏品字布局如何设计?

简单的方式:
上面的div宽100%,
下面的两个div分别宽50%,
然后用float或者inline使其不换行即可

```
<style>
    *{    /* 去除所有元素默认的内外边距的值 */
      padding: 0;
      margin: 0;
    }

    html, body{    /* 默认HTML，body的高度为0，为其设置高度以使后面的div可以用百分比设置高度 */
      height: 100%;
    }

    .header{
      width: 50%;
      height: 50%;
      margin: 0 auto;
      background-color: orange;
    }

    .main{
      width: 100%;
      height: 50%;
    }

    .main .left{
      float: left;
      width: 50%;
      height: 100%;
      background-color: #bfa;
    }

    .main .right{
      float: left;
      width: 50%;
      height: 100%;
      background-color: pink;
    }
```

```
</style>


  <div class="header"></div>
  <div class="main">
    <div class="left"></div>
    <div class="right"></div>
  </div>
```

16.CSS 多列等高如何实现?

（1）利用padding-bottom|margin-bottom正负值相抵，不会影响页面布局的特点。设置父容器超出隐藏（overflow:
hidden），这样父容器的高度就还是它里面的列没有设定padding-bottom时的高度，当它里面的任一列高度增加了，则
父容器的高度被撑到里面最高那列的高度，其他比这列矮的列会用它们的padding-bottom补偿这部分高度差。

（2）利用table-cell所有单元格高度都相等的特性，来实现多列等高。

（3）利用flex布局中弹性元素align-items属性默认为stretch，如果弹性元素未设置高度或设为auto，将占满整个容器的高度
的特性，来实现多列等高。

详细资料可以参考：
《前端应该掌握的 CSS 实现多列等高布局》
《CSS：多列等高布局》

17.经常遇到的浏览器的兼容性有哪些？原因，解决方法是什么，常用 hack 的技巧?

（1）png24位的图片在iE6浏览器上出现背景
解决方案：做成PNG8，也可以引用一段脚本处理。

（2）浏览器默认的margin和padding不同
解决方案：加一个全局的*{margin:0;padding:0;}来统一。

（3）IE6双边距bug：在IE6下，如果对元素设置了浮动，同时又设置了margin-left或
margin-right，margin值会加倍。

```
#box{float:left;width:10px;margin:0 0 0 10px;}
```

这种情况之下IE会产生20px的距离
解决方案：在float的标签样式控制中加入_display:inline;将其转化为行内属性。(_这个符号只有ie6会识别)

（4）渐进识别的方式，从总体中逐渐排除局部。
首先，巧妙的使用"\9"这一标记，将IE浏览器从所有情况中分离出来。
接着，再次使用"+"将IE8和IE7、IE6分离开来，这样IE8已经独立识别。

```
.bb{
background-color:#f1ee18;/*所有识别*/
background-color:#00deff\9;/*IE6、7、8识别*/
```

```
+background-color:#a200ff;/*IE6、7识别*/
_background-color:#1e0bd1;/*IE6识别*/
}
```

（5）IE下，可以使用获取常规属性的方法来获取自定义属性，也可以使用getAttribute()获取自定义属性；Firefox下，只能使用getAttribute()获取自定义属性
解决方法：统一通过getAttribute()获取自定义属性。

（6）IE下，event对象有x、y属性，但是没有pageX、pageY属性;Firefox下，event对象有pageX、pageY属性，但是没有x、y属性。
解决方法：（条件注释）缺点是在IE浏览器下可能会增加额外的HTTP请求数。

（7）Chrome中文界面下默认会将小于12px的文本强制按照12px显示
解决方法：

1.可通过加入CSS属性-webkit-text-size-adjust:none;解决。但是，在chrome更新到27版本之后就不可以用了。

2.还可以使用-webkit-transform:scale(0.5);注意-webkit-transform:scale(0.75);收缩的是整个span的大小，这时候，必须要将span转换成块元素，可以使用display：block/inline-block/...；

（8）超链接访问过后hover样式就不出现了，被点击访问过的超链接样式不再具有hover和active了
解决方法：改变CSS属性的排列顺序L-V-H-A

（9）怪异模式问题：漏写DTD声明，Firefox仍然会按照标准模式来解析网页，但在IE中会触发怪异模式。为避免怪异模式给我们带来不必要的麻烦，最好养成书写DTD声明的好习惯。

18.li 与 li 之间有看不见的空白间隔是什么原因引起的？有什么解决办法?

浏览器会把inline元素间的空白字符（空格、换行、Tab等）渲染成一个空格。而为了美观。我们通常是一个<li>放在一行，
这导致<li>换行后产生换行字符，它变成一个空格，占用了一个字符的宽度。

解决办法：

（1）为<li>设置float:left。不足：有些容器是不能设置浮动，如左右切换的焦点图等。

（2）将所有<li>写在同一行。不足：代码不美观。

（3）将<ul>内的字符尺寸直接设为0，即font-size:0。不足：<ul>中的其他字符尺寸也被设为0，需要额外重新设定其他
字符尺寸，且在Safari浏览器依然会出现空白间隔。

（4）消除<ul>的字符间隔letter-spacing:-8px，不足：这也设置了<li>内的字符间隔，因此需要将<li>内的字符
间隔设为默认letter-spacing:normal。

详细资料可以参考：
《li 与 li 之间有看不见的空白间隔是什么原因引起的？》

19.为什么要初始化 CSS 样式?

-因为浏览器的兼容问题，不同浏览器对有些标签的默认值是不同的，如果没对CSS初始化往往会出现浏览器之间的页面显示差异。

-当然，初始化样式会对SEO有一定的影响，但鱼和熊掌不可兼得，但力求影响最小的情况下初始化。

最简单的初始化方法：*{padding:0;margin:0;}（强烈不建议）

淘宝的样式初始化代码：

```
body,h1,h2,h3,h4,h5,h6,hr,p,blockquote,dl,dt,dd,ul,ol,li,pre,form,fieldset,legend
,button,input,textarea,th,td{margin:0;padding:0;}
body,button,input,select,textarea{font:12px/1.5tahoma,arial,\5b8b\4f53;}
h1,h2,h3,h4,h5,h6{font-size:100%;}
address,cite,dfn,em,var{font-style:normal;}
code,kbd,pre,samp{font-family:couriernew,courier,monospace;}
small{font-size:12px;}
ul,ol{list-style:none;}
a{text-decoration:none;}
a:hover{text-decoration:underline;}
sup{vertical-align:text-top;}
sub{vertical-align:text-bottom;}
legend{color:#000;}
fieldset,img{border:0;}
button,input,select,textarea{font-size:100%;}
table{border-collapse:collapse;border-spacing:0;}
```

20.什么是包含块，对于包含块的理解?

包含块（containing block）就是元素用来计算和定位的一个框。

（1）根元素（很多场景下可以看成是<html>）被称为“初始包含块”，其尺寸等同于浏览器可视窗口的大小。

（2）对于其他元素，如果该元素的position是relative或者static，则“包含块”由其最近的块级祖先元素的content box
边界形成。

（3）如果元素position:fixed，则“包含块”是“初始包含块”。

（4）如果元素position:absolute，则“包含块”由最近的position不为static的祖先元素建立，具体方式如下：

如果该祖先元素是纯inline元素，则规则略复杂：
•假设给内联元素的前后各生成一个宽度为0的内联盒子（inline box），则这两个内联盒子的padding box外面的包
围盒就是内联元素的“包含块”；
•如果该内联元素被跨行分割了，那么“包含块”是未定义的，也就是CSS2.1规范并没有明确定义，浏览器自行发挥
否则，“包含块”由该祖先的padding box边界形成。

如果没有符合条件的祖先元素，则“包含块”是“初始包含块”。

简单描述：默认情况下包含块就是离当前元素最近的块级祖先元素；
对于开启了绝对定位的元素来说，包含块是离它最近的开启了定位（且position不为static）的祖先元素，
如果所有的祖先元素都没有开启定位，则其包含块就是初始包含块。

21.CSS 里的 visibility 属性有个 collapse 属性值是干嘛用的？在不同浏览器下以后什么区别？

（1）对于一般的元素，它的表现跟visibility：hidden;是一样的。元素是不可见的，但此时仍占用页面空间。

（2）但例外的是，如果这个元素是table相关的元素，例如table行，table group，table列，table column group，它的
表现却跟display:none一样，也就是说，它们占用的空间也会释放。

在不同浏览器下的区别：

在谷歌浏览器里，使用collapse值和使用hidden值没有什么区别。

在火狐浏览器、Opera和IE11里，使用collapse值的效果就如它的字面意思：table的行会消失，它的下面一行会补充它的位
置。

详细资料可以参考：
《CSS 里的 visibility 属性有个鲜为人知的属性值：collapse》

22.width:auto 和 width:100%的区别

一般而言

width:100%会使元素box的宽度等于父元素的content box的宽度。

width:auto会使元素撑满整个父元素，margin、border、padding、content区域会自动分配水平空间。

23.绝对定位元素与非绝对定位元素的百分比计算的区别

绝对定位元素的宽高百分比是相对于临近的position不为static的祖先元素的padding box来计算的。

非绝对定位元素的宽高百分比则是相对于父元素的content box来计算的。

24.简单介绍使用图片 base64 编码的优点和缺点。

base64编码是一种图片处理格式，通过特定的算法将图片编码成一长串字符串，在页面上显示的时候，可以用该字符串来代替图片的
url属性。

使用base64的优点是：

（1）减少一个图片的HTTP请求

使用base64的缺点是：

（1）根据base64的编码原理，编码后的大小会比原文件大小大1/3，如果把大图片编码到html/css中，不仅会造成文件体
积的增加，影响文件的加载速度，还会增加浏览器对html或css文件解析渲染的时间。

（2）使用base64无法直接缓存，要缓存只能缓存包含base64的文件，比如HTML或者CSS，这相比于直接缓存图片的效果要
差很多。

（3）兼容性的问题，ie8以前的浏览器不支持。

一般一些网站的小图标可以使用base64图片来引入。

详细资料可以参考：
《玩转图片 base64 编码》
《前端开发中，使用 base64 图片的弊端是什么？》
《小 tip:base64:URL 背景图片与 web 页面性能优化》

25.'display'、'position'和'float'的相互关系?

（1）首先我们判断display属性是否为none，如果为none，则position和float属性的值不影响元素最后的表现。

（2）然后判断position的值是否为absolute或者fixed，如果是，则float属性失效，并且display的值应该被
设置为table或者block，具体转换需要看初始转换值。

（3）如果position的值不为absolute或者fixed，则判断float属性的值是否为none，如果不是，则display
的值则按上面的规则转换。注意，如果position的值为relative并且float属性的值存在，则relative相对
于浮动后的最终位置定位。

（4）如果float的值为none，则判断元素是否为根元素，如果是根元素则display属性按照上面的规则转换，如果不是，
则保持指定的display属性值不变。

总的来说，可以把它看作是一个类似优先级的机制，"position:absolute"和"position:fixed"优先级最高，有它存在
的时候，浮动不起作用，'display'的值也需要调整；其次，元素的'float'特性的值不是"none"的时候或者它是根元素
的时候，调整'display'的值；最后，非根元素，并且非浮动元素，并且非绝对定位的元素，'display'特性值同设置值。

详细资料可以参考：
《position 跟 display、margincollapse、overflow、float 这些特性相互叠加后会怎么样？》

26.margin 重叠问题的理解。

相关知识点：

块级元素的上外边距（margin-top）与下外边距（margin-bottom）有时会合并为单个外边距，这样的现象称为“margin合
并”。

产生折叠的必备条件：margin必须是邻接的!

而根据w3c规范，两个margin是邻接的必须满足以下条件：

•必须是处于常规文档流（非float和绝对定位）的块级盒子，并且处于同一个BFC当中。
•没有线盒，没有空隙，没有padding和border将他们分隔开
•都属于垂直方向上相邻的外边距，可以是下面任意一种情况
•元素的margin-top与其第一个常规文档流的子元素的margin-top
•元素的margin-bottom与其下一个常规文档流的兄弟元素的margin-top
•height为auto的元素的margin-bottom与其最后一个常规文档流的子元素的margin-bottom
•高度为0并且最小高度也为0，不包含常规文档流的子元素，并且自身没有建立新的BFC的元素的margin-top和margin-bottom

margin合并的3种场景：

（1）相邻兄弟元素margin合并。

解决办法：
•设置块状格式化上下文元素（BFC），即开启BFC

（2）父级和第一个/最后一个子元素的margin合并。

解决办法：

对于margin-top合并，可以进行如下操作（满足一个条件即可）：
•父元素设置为块状格式化上下文元素；即开启BFC
•父元素设置border-top值；
•父元素设置padding-top值；
•父元素和第一个子元素之间添加内联元素进行分隔。

对于margin-bottom合并，可以进行如下操作（满足一个条件即可）：
•父元素设置为块状格式化上下文元素；
•父元素设置border-bottom值；
•父元素设置padding-bottom值；
•父元素和最后一个子元素之间添加内联元素进行分隔；
•父元素设置height、min-height或max-height。

（3）空块级元素的margin合并。

解决办法：
•设置垂直方向的border；
•设置垂直方向的padding；
•里面添加内联元素（直接Space键空格是没用的）；
•设置height或者min-height。

回答:

margin重叠指的是在垂直方向上，两个相邻元素的margin发生重叠的情况。

一般来说可以分为四种情形：

第一种是相邻兄弟元素的marin-bottom和margin-top的值发生重叠。这种情况下我们可以通过设置其中一个元素为BFC
来解决。

第二种是父元素的margin-top和子元素的margin-top发生重叠。它们发生重叠是因为它们是相邻的，所以我们可以通过这
一点来解决这个问题。我们可以为父元素设置border-top、padding-top值来分隔它们，当然我们也可以将父元素设置为BFC
来解决。

第三种是高度为auto的父元素的margin-bottom和子元素的margin-bottom发生重叠。它们发生重叠一个是因为它们相
邻，一个是因为父元素的高度不固定。因此我们可以为父元素设置border-bottom、padding-bottom来分隔它们，也可以为
父元素设置一个高度，max-height和min-height也能解决这个问题。当然将父元素设置为BFC是最简单的方法。

第四种情况，是没有内容的元素，自身的margin-top和margin-bottom发生的重叠。我们可以通过为其设置border、pa
dding或者高度来解决这个问题。

27.对 BFC 规范（块级格式化上下文：block formatting context）的理解?

相关知识点：

块级格式化上下文（Block Formatting Context，BFC）是Web页面的可视化CSS渲染的一部分，是布局过程中生成块级盒
子的区域，也是浮动元素与其他元素的交互限定区域。

通俗来讲

•BFC是一个独立的布局环境，可以理解为一个容器，在这个容器中按照一定规则进行物品摆放，并且不会影响其它环境中的物品。
•如果一个元素符合触发BFC的条件，则BFC中的元素布局不受外部影响。

创建BFC

（1）根元素或包含根元素的元素
（2）浮动元素float＝left|right或inherit（≠none）
（3）绝对定位元素position＝absolute或fixed
（4）display＝inline-block|flex|inline-flex|table-cell或table-caption
（5）overflow＝hidden|auto或scroll(≠visible)

回答：

BFC指的是块级格式化上下文，一个元素开启了BFC之后，那么它内部元素产生的布局不会影响到外部元素，外部元素的布局也
不会影响到BFC中的内部元素。一个BFC就像是一个隔离区域，和其他区域互不影响。

一般来说根元素是一个BFC区域，浮动和绝对定位的元素也会形成BFC，display属性的值为inline-block、flex这些
属性时也会创建BFC。还有就是元素的overflow的值不为visible时都会创建BFC。

详细资料可以参考：
《深入理解 BFC 和 MarginCollapse》
《前端面试题-BFC（块格式化上下文）》

28.IFC 是什么?

IFC指的是行级格式化上下文，它有这样的一些布局规则：

（1）行级上下文内部的盒子会在水平方向，一个接一个地放置。
（2）当一行不够的时候会自动切换到下一行。
（3）行级上下文的高度由内部最高的内联盒子的高度决定。

详细资料可以参考：
《[译]:BFC 与 IFC》
《BFC 和 IFC 的理解（布局）》

29.请解释一下为什么需要清除浮动？清除浮动的方式

浮动元素可以左右移动，直到遇到另一个浮动元素或者遇到它外边缘的包含框。浮动框不属于文档流中的普通流，当元素浮动之后，
不会影响块级元素的布局，只会影响内联元素布局。此时文档流中的普通流就会表现得该浮动框不存在一样的布局模式。当包含框
的高度小于浮动框的时候，此时就会出现“高度塌陷”。

清除浮动是为了清除使用浮动元素产生的影响。浮动的元素，高度会塌陷，而高度的塌陷使我们页面后面的布局不能正常显示。

清除浮动的方式

（1）使用clear属性清除浮动。参考30。

（2）使用BFC块级格式化上下文来清除浮动。参考26。

因为BFC元素不会影响外部元素的特点，所以BFC元素也可以用来清除浮动的影响，因为如果不清除，子元素浮动则父元
素高度塌陷，必然会影响后面元素布局和定位，这显然有违BFC元素的子元素不会影响外部元素的设定。

30.使用 clear 属性清除浮动的原理?

使用clear属性清除浮动，其语法如下：

```
clear:none|left|right|both
```

如果单看字面意思，clear:left应该是“清除左浮动”，clear:right应该是“清除右浮动”的意思，实际上，这种解释是有问
题的，因为浮动一直还在，并没有清除。

官方对clear属性的解释是：“元素盒子的边不能和前面的浮动元素相邻。”，我们对元素设置clear属性是为了避免浮动元素
对该元素的影响，而不是清除掉浮动。

还需要注意的一点是clear属性指的是元素盒子的边不能和前面的浮动元素相邻，注意这里“前面的”3个字，也就是clear属
性对“后面的”浮动元素是不闻不问的。考虑到float属性要么是left，要么是right，不可能同时存在，同时由于clear
属性对“后面的”浮动元素不闻不问，因此，当clear:left有效的时候，clear:right必定无效，也就是此时clear:left
等同于设置clear:both；同样地，clear:right如果有效也是等同于设置clear:both。由此可见，clear:left和cle

ar:right这两个声明就没有任何使用的价值，至少在CSS世界中是如此，直接使用clear:both吧。

一般使用伪元素的方式清除浮动

```
.clear::before,  // 解决外边距重叠的问题
.clear::after{   // 解决高度塌陷的问题
content:'';
display:table;//也可以是'block'，或者是'list-item'
clear:both;
}
```

clear属性只有块级元素才有效的，而::after等伪元素默认都是内联水平，这就是借助伪元素清除浮动影响时需要设置disp
lay属性值的原因。

31.zoom:1 的清除浮动原理?

清除浮动，触发hasLayout；
zoom属性是IE浏览器的专有属性，它可以设置或检索对象的缩放比例。解决ie下比较奇葩的bug。譬如外边距（margin）
的重叠，浮动清除，触发ie的haslayout属性等。

来龙去脉大概如下：
当设置了zoom的值之后，所设置的元素就会扩大或者缩小，高度宽度就会重新计算了，这里一旦改变zoom值时其实也会发
生重新渲染，运用这个原理，也就解决了ie下子元素浮动时候父元素不随着自动扩大的问题。

zoom属性是IE浏览器的专有属性，火狐和老版本的webkit核心的浏览器都不支持这个属性。然而，zoom现在已经被逐步标
准化，出现在CSS3.0规范草案中。

目前非ie由于不支持这个属性，它们又是通过什么属性来实现元素的缩放呢？可以通过css3里面的动画属性scale进行缩放。

32.移动端的布局用过媒体查询吗?

假设你现在正用一台显示设备来阅读这篇文章，同时你也想把它投影到屏幕上，或者打印出来，而显示设备、屏幕投影和打印等这些
媒介都有自己的特点，CSS就是为文档提供在不同媒介上展示的适配方法

当媒体查询为真时，相关的样式表或样式规则会按照正常的级联规则被应用。当媒体查询返回假，标签上带有媒体查询的样式表仍将被
下载（只不过不会被应用）。

包含了一个媒体类型和至少一个使用宽度、高度和颜色等媒体属性来限制样式表范围的表达式。CSS3加入的媒体查询使得无需修改
内容便可以使样式应用于某些特定的设备范围。

详细资料可以参考：
《CSS3@media 查询》
《响应式布局和自适应布局详解》

33.使用 CSS 预处理器吗？喜欢哪个？

SASS（SASS、LESS没有本质区别，只因为团队前端都是用的SASS）

34.CSS 优化、提高性能的方法有哪些？

加载性能：

（1）css压缩：将写好的css进行打包压缩，可以减少很多的体积。
（2）css单一样式：当需要下边距和左边距的时候，很多时候选择:margin:top 0 bottom 0;但margin-bottom:bot
tom;margin-left:left;执行的效率更高。
（3）减少使用@import,而建议使用link，因为后者在页面加载时一起加载，前者是等待页面加载完成之后再进行加载。

选择器性能：

（1）关键选择器（key selector）。选择器的最后面的部分为关键选择器（即用来匹配目标元素的部分）。CSS选择器是从右到
左进行匹配的。当使用后代选择器的时候，浏览器会遍历所有子元素来确定是否是指定的元素等等；

（2）如果规则拥有ID选择器作为其关键选择器，则不要为规则增加标签。过滤掉无关的规则（这样样式系统就不会浪费时间去匹
配它们了）。

（3）避免使用通配规则，如*{}计算次数惊人！只对需要用到的元素进行选择。

（4）尽量少的去对标签进行选择，而是用class。

（5）尽量少的去使用后代选择器，降低选择器的权重值。后代选择器的开销是最高的，尽量将选择器的深度降到最低，最高不要超过
三层，更多的使用类来关联每一个标签元素。

（6）了解哪些属性是可以通过继承而来的，然后避免对这些属性重复指定规则。

渲染性能：

（1）慎重使用高性能属性：浮动、定位。

（2）尽量减少页面重排、重绘。

（3）去除空规则：{}。空规则的产生原因一般来说是为了预留样式。去除这些空规则无疑能减少css文档体积。

（4）属性值为0时，不加单位。

（5）属性值为浮动小数0.**，可以省略小数点之前的0。

（6）标准化各种浏览器前缀：带浏览器前缀的在前。标准属性在后。

（7）不使用@import前缀，它会影响css的加载速度。

（8）选择器优化嵌套，尽量避免层级过深。

（9）css雪碧图，同一页面相近部分的小图标，方便使用，减少页面的请求次数，但是同时图片本身会变大，使用时，优劣考虑清

楚，再使用。

（10）正确使用display的属性，由于display的作用，某些样式组合会无效，徒增样式体积的同时也影响解析性能。

（11）不滥用web字体。对于中文网站来说WebFonts可能很陌生，国外却很流行。web fonts通常体积庞大，而且一些浏
览器在下载web fonts时会阻塞页面渲染损伤性能。

可维护性、健壮性：

（1）将具有相同属性的样式抽离出来，整合并通过class在页面中进行使用，提高css的可维护性。
（2）样式与内容分离：将css代码定义到外部css中。

详细资料可以参考：
《CSS 优化、提高性能的方法有哪些？》
《CSS 优化，提高性能的方法》

35.浏览器是怎样解析 CSS 选择器的?

样式系统从关键选择器开始匹配，然后左移查找规则选择器的祖先元素。只要选择器的子树一直在工作，样式系统就会持续左移，直
到和规则匹配，或者是因为不匹配而放弃该规则。

试想一下，如果采用从左至右的方式读取CSS规则，那么大多数规则读到最后（最右）才会发现是不匹配的，这样做会费时耗能，
最后有很多都是无用的；而如果采取从右向左的方式，那么只要发现最右边选择器不匹配，就可以直接舍弃了，避免了许多无效匹配。

详细资料可以参考：
《探究 CSS 解析原理》

36.在网页中应该使用奇数还是偶数的字体？为什么呢?

（1）偶数字号相对更容易和web设计的其他部分构成比例关系。比如：当我用了14px的正文字号，我可能会在一些地方用14
×0.5=7px的margin，在另一些地方用14×1.5=21px的标题字号。
（2）浏览器缘故，低版本的浏览器ie6会把奇数字体强制转化为偶数，即13px渲染为14px。
（3）系统差别，早期的Windows里，中易宋体点阵只有12和14、15、16px，唯独缺少13px。

详细资料可以参考：
《谈谈网页中使用奇数字体和偶数字体》
《现在网页设计中的为什么少有人用 11px、13px、15px 等奇数的字体？》

37.margin 和 padding 分别适合什么场景使用?

margin是用来隔开元素与元素的间距；padding是用来隔开元素与内容的间隔。
margin用于布局分开元素使元素与元素互不相干。
padding用于元素与内容之间的间隔，让内容（文字）与（包裹）元素之间有一段距离。

何时应当使用margin：
•需要在border外侧添加空白时。
•空白处不需要背景（色）时。
•上下相连的两个盒子之间的空白，需要相互抵消时。如15px+20px的margin，将得到20px的空白。

何时应当时用padding：
•需要在border内测添加空白时。
•空白处需要背景（色）时。
•上下相连的两个盒子之间的空白，希望等于两者之和时。如15px+20px的padding，将得到35px的空白。

38.抽离样式模块怎么写，说出思路，有无实践经验？[阿里航旅的面试题]

我的理解是把常用的css样式单独做成css文件……通用的和业务相关的分离出来，通用的做成样式模块儿共享，业务相关的，放
进业务相关的库里面做成对应功能的模块儿。

详细资料可以参考：
《CSS 规范-分类方法》

39.简单说一下 css3 的 all 属性。

all属性实际上是所有CSS属性的缩写，表示，所有的CSS属性都怎样怎样，但是，不包括unicode-bidi和direction
这两个CSS属性。支持三个CSS通用属性值，initial,inherit,unset。

initial是初始值的意思，也就是该元素除了unicode-bidi和direction以外的CSS属性都使用属性的默认初始
值。

inherit是继承的意思，也就是该元素除了unicode-bidi和direction以外的CSS属性都继承父元素的属性值。

unset是取消设置的意思，也就是当前元素浏览器或用户设置的CSS忽略，然后如果是具有继承特性的CSS，如color，则
使用继承值；如果是没有继承特性的CSS属性，如background-color，则使用初始值。

详细资料可以参考：
《简单了解 CSS3 的 all 属性》

40.为什么不建议使用统配符初始化 css 样式。

采用*{padding:0;margin:0;}这样的写法好处是写起来很简单，但是是通配符，需要把所有的标签都遍历一遍，当网站较大时，
样式比较多，这样写就大大的加强了网站运行的负载，会使网站加载的时候需要很长一段时间，因此一般大型的网站都有分层次的一
套初始化样式。

出于性能的考虑，并不是所有标签都会有padding和margin，因此对常见的具有默认padding和margin的元素初始化即
可，并不需使用通配符*来初始化。

41.absolute 的 containingblock（包含块）计算方式跟正常流有什么不同?

（1）内联元素也可以作为“包含块”所在的元素；

（2）“包含块”所在的元素不是父块级元素，而是最近的position不为static的祖先元素或根元素；

（3）边界是padding box而不是content box。

42.对于 hasLayout 的理解?

hasLayout是IE特有的一个属性。很多的IE下的css bug都与其息息相关。在IE中，一个元素要么自己对自身的内容进
行计算大小和组织，要么依赖于父元素来计算尺寸和组织内容。当一个元素的hasLayout属性值为true时，它负责对自己和可
能的子孙元素进行尺寸计算和定位。虽然这意味着这个元素需要花更多的代价来维护自身和里面的内容，而不是依赖于祖先元素来完
成这些工作。

详细资料可以参考：
《CSS 基础篇--CSS 中 IE 浏览器的 hasLayout，IE 低版本的 bug 根源》
《CSS 魔法堂：hasLayout 原来是这样的！》

43.元素竖向的百分比设定是相对于容器的高度吗?

如果是height的话，是相对于包含块的高度。

如果是padding或者margin竖直方向的属性则是相对于包含块的宽度。

44.全屏滚动的原理是什么？用到了 CSS 的哪些属性？（待深入实践）

原理：有点类似于轮播，整体的元素一直排列下去，假设有5个需要展示的全屏页面，那么高度是500%，只是展示100%，容器及容
器内的页面取当前可视区高度，同时容器的父级元素overflow属性值设为hidden，通过更改容器可视区的位置来实现全
屏滚动效果。主要是响应鼠标事件，页面通过CSS的动画效果，进行移动。

```
overflow：hidden；transition：all 1000ms ease；
```

详细资料可以参考：
《js 实现网页全屏切换（平滑过渡），鼠标滚动切换》
《用 ES6 写全屏滚动插件》

45.什么是响应式设计？响应式设计的基本原理是什么？如何兼容低版本的 IE？（待深入了解）

响应式网站设计是一个网站能够兼容多个终端，而不是为每一个终端做一个特定的版本。基本原理是通过媒体查询检测不同的设备屏
幕尺寸做处理。页面头部必须有meta声明的viewport。

详细资料可以参考：
《响应式布局原理》
《响应式布局的实现方法和原理》

46.视差滚动效果，如何给每页做不同的动画？（回到顶部，向下滑动要再次出现，和只出现一次分别怎么做？）

> 视差滚动是指多层背景以不同的速度移动，形成立体的运动效果，带来非常出色的视觉体验。

详细资料可以参考：
[《如何实现视差滚动效果的网页？》]

47.如何修改 chrome 记住密码后自动填充表单的黄色背景?

> chrome表单自动填充后，input文本框的背景会变成黄色的，通过审查元素可以看到这是由于chrome会默认给自动填充的in
> put表单加上input:-webkit-autofill私有属性，然后对其赋予以下样式：
>
> ```
> {
> background-color:rgb(250,255,189)!important;
> background-image:none!important;
> color:rgb(0,0,0)!important;
> }
> ```
>
> 对chrome默认定义的background-color，background-image，color使用important是不能提高其优先级的，但是
> 其他属性可使用。
>
> 使用足够大的纯色内阴影来覆盖input输入框的黄色背景，处理如下
>
> ```
> input:-webkit-autofill,textarea:-webkit-autofill,select:-webkit-autofill{
> -webkit-box-shadow:000px 1000px white inset;
> border:1px solid #CCC !important;
> }
> ```

详细资料可以参考：
[《去掉 chrome 记住密码后的默认填充样式》]
[《修改谷歌浏览器 chrome 记住密码后自动填充表单的黄色背景》]

48.怎么让 Chrome 支持小于 12px 的文字?

在谷歌下css设置字体大小为12px及以下时，显示都是一样大小，都是默认12px。

解决办法：

（1）可以使用Webkit的内核的-webkit-text-size-adjust的私有CSS属性来解决，只要加了-webkit-text-size
-adjust:none;字体大小就不受限制了。但是chrome更新到27版本之后就不可以用了。所以高版本chrome谷歌浏览器
已经不再支持-webkit-text-size-adjust样式，所以要使用时候慎用。

（2）还可以使用css3的transform缩放属性-webkit-transform:scale(0.5);注意-webkit-transform:scale(0.
75);收缩的是整个元素的大小，这时候，如果是内联元素，必须要将内联元素转换成块元素，可以使用display：block/
inline-block/...；

（3）使用图片：如果是内容固定不变情况下，使用将小于12px文字内容切出做图片，这样不影响兼容也不影响美观。

详细资料可以参考：
《谷歌浏览器不支持 CSS 设置小于 12px 的文字怎么办？》

49.让页面里的字体变清晰，变细用 CSS 怎么做?

webkit内核的私有属性：-webkit-font-smoothing，用于字体抗锯齿，使用后字体看起来会更清晰舒服。

在MacOS测试环境下面设置-webkit-font-smoothing:antialiased;但是这个属性仅仅是面向MacOS，其他操作系统设
置后无效。

详细资料可以参考：
《让字体变的更清晰 CSS 中-webkit-font-smoothing》

50.font-style 属性中 italic 和 oblique 的区别?

italic和oblique这两个关键字都表示“斜体”的意思。

它们的区别在于，italic是使用当前字体的斜体字体，而oblique只是单纯地让文字倾斜。如果当前字体没有对应的斜体字体，
则退而求其次，解析为oblique，也就是单纯形状倾斜。

51.设备像素、css 像素、设备独立像素、dpr、ppi 之间的区别?

设备像素指的是物理像素，一般手机的分辨率指的就是设备像素，一个设备的设备像素是不可变的。

css像素和设备独立像素是等价的，不管在何种分辨率的设备上，css像素的大小应该是一致的，css像素是一个相对单位，它是相
对于设备像素的，一个css像素的大小取决于页面缩放程度和dpr的大小。

dpr指的是设备像素和设备独立像素的比值，一般的pc屏幕，dpr=1。在iphone4时，苹果推出了retina屏幕，它的dpr
为2。屏幕的缩放会改变dpr的值。

ppi指的是每英寸的物理像素的密度，ppi越大，屏幕的分辨率越大。

详细资料可以参考：
《什么是物理像素、虚拟像素、逻辑像素、设备像素，什么又是 PPI,DPI,DPR 和 DIP》
《前端工程师需要明白的「像素」》
《CSS 像素、物理像素、逻辑像素、设备像素比、PPI、Viewport》
《前端开发中像素的概念》

52.layout viewport、visual viewport 和 ideal viewport 的区别?

相关知识点：

如果把移动设备上浏览器的可视区域设为viewport的话，某些网站就会因为viewport太窄而显示错乱，所以这些浏览器就决定
默认情况下把viewport设为一个较宽的值，比如980px，这样的话即使是那些为桌面设计的网站也能在移动浏览器上正常显示了。
ppk把这个浏览器默认的viewport叫做layout viewport。

layout viewport的宽度是大于浏览器可视区域的宽度的，所以我们还需要一个viewport来代表浏览器可视区域的大小，ppk把
这个viewport叫做visual viewport。

ideal viewport是最适合移动设备的viewport，ideal viewport的宽度等于移动设备的屏幕宽度，只要在css中把某一元
素的宽度设为ideal viewport的宽度（单位用px），那么这个元素的宽度就是设备屏幕的宽度了，也就是宽度为100%的效果。i
deal viewport的意义在于，无论在何种分辨率的屏幕下，那些针对ideal viewport而设计的网站，不需要用户手动缩放，也
不需要出现横向滚动条，都可以完美的呈现给用户。

回答：

移动端一共需要理解三个viewport的概念的理解。

第一个视口是布局视口，在移动端显示网页时，由于移动端的屏幕尺寸比较小，如果网页使用移动端的屏幕尺寸进行布局的话，那么整
个页面的布局都会显示错乱。所以移动端浏览器提供了一个layout viewport布局视口的概念，使用这个视口来对页面进行布局展
示，一般layout viewport的大小为980px，因此页面布局不会有太大的变化，我们可以通过拖动和缩放来查看到这个页面。

第二个视口指的是视觉视口，visual viewport指的是移动设备上我们可见的区域的视口大小，一般为屏幕的分辨率的大小。visu
al viewport和layout viewport的关系，就像是我们通过窗户看外面的风景，视觉视口就是窗户，而外面的风景就是布局视口
中的网页内容。

第三个视口是理想视口，由于layout viewport一般比visual viewport要大，所以想要看到整个页面必须通过拖动和缩放才
能实现。所以又提出了ideal viewport的概念，ideal viewport下用户不用缩放和滚动条就能够查看到整个页面，并且页面在
不同分辨率下显示的内容大小相同。ideal viewport其实就是通过修改layout viewport的大小，让它等于设备的宽度，这个
宽度可以理解为是设备独立像素，因此根据ideal viewport设计的页面，在不同分辨率的屏幕下，显示应该相同。

详细资料可以参考：
《移动前端开发之 viewport 的深入理解》
《说说移动前端中 viewport（视口）》
《移动端适配知识你到底知多少》

53.position:fixed;在 android 下无效怎么处理?

因为移动端浏览器默认的viewport叫做layout viewport。在移动端显示时，因为layout viewport的宽度大于移动端屏幕
的宽度，所以页面会出现滚动条左右移动，fixed的元素是相对layout viewport来固定位置的，而不是移动端屏幕来固定位置的
，所以会出现感觉fixed无效的情况。

如果想实现fixed相对于屏幕的固定效果，我们需要改变的是viewport的大小为ideal viewport，可以如下设置：

```
<meta name="viewport" content="width=device-width,initial-scale=1.0,maximum-scale=1.0,minimum-sca
le=1.0,user-scalable=no"/>
```

54.如果需要手动写动画，你认为最小时间间隔是多久，为什么？（阿里）

多数显示器默认频率是60Hz，即1秒刷新60次，所以理论上最小间隔为1/60*1000ms＝16.7ms

55.如何去除 inline-block 元素间间距?

移除空格、使用margin负值、使用font-size:0、letter-spacing、word-spacing

详细资料可以参考：
《去除 inline-block 元素间间距的 N 种方法》

56.overflow:scroll 时不能平滑滚动的问题怎么处理?

以下代码可解决这种卡顿的问题：-webkit-overflow-scrolling:touch;是因为这行代码启用了硬件加速特性，所以滑动很流
畅。

详细资料可以参考：
《解决页面使用 overflow:scroll 在 iOS 上滑动卡顿的问题》

57.有一个高度自适应的 div，里面有两个 div，一个高度 100px，希望另一个填满剩下的高度。

（1）外层div使用position：relative；高度要求自适应的div使用
position:absolute;top:100px;bottom:0;
left:0;right:0;

（2）使用flex布局，设置主轴为竖轴，第二个div的flex-grow为1。

详细资料可以参考：
《有一个高度自适应的 div，里面有两个 div，一个高度 100px，希望另一个填满剩下的高度(三种方案)》

58.png、jpg、gif 这些图片格式解释一下，分别什么时候用。有没有了解过webp?

相关知识点：

（1）BMP，是无损的、既支持索引色也支持直接色的、点阵图。这种图片格式几乎没有对数据进行压缩，所以BMP格式的图片通常
具有较大的文件大小。

（2）GIF是无损的、采用索引色的、点阵图。采用LZW压缩算法进行编码。文件小，是GIF格式的优点，同时，GIF格式还具
有支持动画以及透明的优点。但，GIF格式仅支持8bit的索引色，所以GIF格式适用于对色彩要求不高同时需要文件体积
较小的场景。

（3）JPEG是有损的、采用直接色的、点阵图。JPEG的图片的优点，是采用了直接色，得益于更丰富的色彩，JPEG非常适合用来
存储照片，与GIF相比，JPEG不适合用来存储企业Logo、线框类的图。因为有损压缩会导致图片模糊，而直接色的选用，
又会导致图片文件较GIF更大。

（4）PNG-8是无损的、使用索引色的、点阵图。PNG是一种比较新的图片格式，PNG-8是非常好的GIF格式替代者，在可能的
情况下，应该尽可能的使用PNG-8而不是GIF，因为在相同的图片效果下，PNG-8具有更小的文件体积。除此之外，PNG-8
还支持透明度的调节，而GIF并不支持。现在，除非需要动画的支持，否则我们没有理由使用GIF而不是PNG-8。

（5）PNG-24是无损的、使用直接色的、点阵图。PNG-24的优点在于，它压缩了图片的数据，使得同样效果的图片，PNG-24格
式的文件大小要比BMP小得多。当然，PNG24的图片还是要比JPEG、GIF、PNG-8大得多。

（6）SVG是无损的、矢量图。SVG是矢量图。这意味着SVG图片由直线和曲线以及绘制它们的方法组成。当你放大一个SVG图
片的时候，你看到的还是线和曲线，而不会出现像素点。这意味着SVG图片在放大时，不会失真，所以它非常适合用来绘制企
业Logo、Icon等。

（7）WebP是谷歌开发的一种新图片格式，WebP是同时支持有损和无损压缩的、使用直接色的、点阵图。从名字就可以看出来它是
为Web而生的，什么叫为Web而生呢？就是说相同质量的图片，WebP具有更小的文件体积。现在网站上充满了大量的图片，
如果能够降低每一个图片的文件大小，那么将大大减少浏览器和服务器之间的数据传输量，进而降低访问延迟，提升访问体验。

- 在无损压缩的情况下，相同质量的WebP图片，文件大小要比PNG小26%；
- 在有损压缩的情况下，具有相同图片精度的WebP图片，文件大小要比JPEG小25%~34%；
- WebP图片格式支持图片透明度，一个无损压缩的WebP图片，如果要支持透明度只需要22%的格外文件大小。

但是目前只有Chrome浏览器和Opera浏览器支持WebP格式，兼容性不太好。

回答：

（1）jpeg(jpg)：支持的颜色比较丰富，不支持透明效果，不支持动图。
一般用来显示照片。

（2）gif：支持的颜色比较少，支持简单透明，支持动图。
适用颜色单一的图片、动图。

（3）png：支持的颜色丰富，支持复杂透明，不支持动图。
专为网页而生

（4）webp：这种格式是谷歌推出的专门用来表示网页中的图片的一种格式。
使用webp格式的最大的优点是，在相同质量的文件下，它拥有更小的文件体积。因此它非常适合于网络图片的传输，
因为图片体积的减少，意味着请求时间的减小，这样会提高用户的体验。
缺点是目前在兼容性上不太好。

选取图片规则：效果一样用小的，效果不一样用效果好的。

详细资料可以参考：
《图片格式那么多，哪种更适合你？》

59.浏览器如何判断是否支持 webp 格式图片

（1）宽高判断法。通过创建image对象，将其src属性设置为webp格式的图片，然后在onload事件中获取图片的宽高，如
果能够获取，则说明浏览器支持webp格式图片。如果不能获取或者触发了onerror函数，那么就说明浏览器不支持webp格
式的图片。

（2）canvas判断方法。我们可以动态的创建一个canvas对象，通过canvas的toDataURL将设置为webp格式，然后判断
返回值中是否含有image/webp字段，如果包含则说明支持WebP，反之则不支持。

详细资料可以参考：
《判断浏览器是否支持 WebP 图片》
《toDataURL()》

60.什么是 Cookie 隔离？（或者说：请求资源的时候不要让它带 cookie 怎么做）

网站向服务器请求的时候，会自动带上cookie，这样增加表头信息量，使请求变慢。

如果静态文件都放在主域名下，那静态文件请求的时候都带有的cookie的数据提交给server的，非常浪费流量，所以不如隔离开
，静态资源放CDN。

因为cookie有域的限制，因此不能跨域提交请求，故使用非主域名的时候，请求头中就不会带有cookie数据，这样可以降低请
求头的大小，降低请求时间，从而达到降低整体请求延时的目的。

同时这种方式不会将cookie传入WebServer，也减少了WebServer对cookie的处理分析环节，提高了webserver的
http请求的解析速度。

详细资料可以参考：
《CDN 是什么？使用 CDN 有什么优势？》

61.style 标签写在 body 后与 body 前有什么区别?

页面加载自上而下当然是先加载样式。写在body标签后由于浏览器以逐行方式对HTML文档进行解析，当解析到写在尾部的样式
表（外联或写在style标签）会导致浏览器停止之前的渲染，等待加载且解析样式表完成之后重新渲染，在windows的IE下可
能会出现FOUC现象（即样式失效导致的页面闪烁问题（结构解析完但样式还没有解析完））

62.什么是 CSS 预处理器/后处理器?

CSS预处理器定义了一种新的语言，其基本思想是，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。

预处理器例如：LESS、Sass、Stylus，用来预编译Sass或less ，增强了css代码的复用性，还有层级、mixin、
变量、循环、函数等，具有很方便的UI组件模块化开发能力，极大的提高工作效率。

CSS后处理器是对CSS进行处理，并最终生成CSS的预处理器，它属于广义上的CSS预处理器。我们很久以前就在用CSS后
处理器了，最典型的例子是CSS压缩工具（如clean-css），只不过以前没单独拿出来说过。还有最近比较火的Autoprefixer，
以CanIUse上的浏览器支持数据为基础，自动处理兼容性问题。

后处理器例如：PostCSS，通常被视为在完成的样式表中根据CSS规范处理CSS，让其更有效；目前最常做的是给CSS属性添加浏
览器私有前缀，实现跨浏览器兼容性的问题。

详细资料可以参考：
《CSS 预处理器和后处理器》

63.阐述一下 CSS Sprites

将一个页面涉及到的所有图片都包含到一张大图中去，然后利用CSS的background-image，background-repeat，background
-position的组合进行背景定位。利用CSS Sprites能很好地减少网页的http请求，从而很好的提高页面的性能；CSS Sprites
能减少图片的字节。

优点：

减少HTTP请求数，极大地提高页面加载速度
增加图片信息重复度，提高压缩比，减少图片大小
更换风格方便，只需在一张或几张图片上修改颜色或样式即可实现

缺点：

图片合并麻烦
维护麻烦，修改一个图片可能需要重新布局整个图片，样式

64.使用 rem 布局的优缺点?

优点：
在屏幕分辨率千差万别的时代，只要将rem与屏幕分辨率关联起来就可以实现页面的整体缩放，使得在设备上的展现都统一起来了。
而且现在浏览器基本都已经支持rem了，兼容性也非常的好。
0
缺点：
（1）在奇葩的dpr设备上表现效果不太好，比如一些华为的高端机型用rem布局会出现错乱。
（2）使用iframe引用也会出现问题。
（3）rem在多屏幕尺寸适配上与当前两大平台的设计哲学不一致。即大屏的出现到底是为了看得又大又清楚，还是为了看的更多的问
题。

详细资料可以参考：
《css3 的字体大小单位 rem 到底好在哪？》
《VW:是时候放弃 REM 布局了》
《为什么设计稿是 750px》
《使用 Flexible 实现手淘 H5 页面的终端适配》

65.几种常见的 CSS 布局

圣杯布局：

1. 比较特殊的三栏布局，同样也是**两边固定宽度，中间自适应**，唯一区别是dom结构必须是**先写中间列部分**，这样实现中间列可以优先加载。
2. 三个部分都设定为左浮动，否则左右两边内容上不去，就不可能与中间列同一行。然后设置center的宽度为100%(**实现中间列内容自适应)**，此时，left和right部分会跳到下一行。
3. 通过设置margin-left为负值让left和right部分回到与center部分同一行。
4. 通过设置父容器的padding-left和padding-right，让左右两边留出间隙。
5. 通过设置相对定位，让left和right部分移动到两边。

圣杯布局的缺点：

- center部分的最小宽度不能小于left部分的宽度，否则会left部分掉到下一行
- 如果其中一列内容高度拉长，其他两列的背景并不会自动填充。(借助等高布局正padding+负margin可解决，下文会介绍)

双飞翼布局：

1. 同样也是三栏布局，在圣杯布局基础上进一步优化，解决了圣杯布局错乱问题，实现了内容与布局的分离。而且任何一栏都可以是最高栏，不会出问题。
2. 实现步骤(前两步与圣杯布局一样)
    1. 三个部分都设定为左浮动，然后设置center的宽度为100%，此时，left和right部分会跳到下一行；
    2. 通过设置margin-left为负值让left和right部分回到与center部分同一行；
    3. center部分增加一个内层div，并设margin: 0 200px；

双飞翼布局的缺点：

```
多加一层 dom 树节点，增加渲染树生成的计算量。
```

两种布局实现方式对比:

- 两种布局方式都是把主列放在文档流最前面，使主列优先加载。
- 两种布局方式在实现上也有相同之处，都是让三列浮动，然后通过负外边距形成三列布局。
- 两种布局方式的不同之处在于如何处理中间主列的位置： **圣杯布局是利用父容器的左、右内边距+两个从列相对定位； 双飞翼布局是把主列嵌套在一个新的父级块中利用主列的左、右外边距进行布局调整。**

详细的资料可以参考：

《几种常见的 CSS 布局》

66.画一条 0.5px 的线

```
采用meta viewport的方式

采用border-image的方式

回答时答这个就行：
采用transform:scaleY()的方式 //注意chrome下实线变虚的问题，可以通过设置transform-origin: 50% 100%解决
```

详细资料可以参考：
《怎么画一条 0.5px 的边（更新）》

67.transition 和 animation 的区别

```
transition关注的是CSS property的变化，property值和时间的关系是一个三次贝塞尔曲线。一般结合transforms使用

animation作用于元素本身而不是样式属性，可以使用关键帧的概念，应该说可以实现更自由的动画效果。
```

详细资料可以参考：
《CSSanimation 与 CSStransition 有何区别？》
《CSS3Transition 和 Animation 区别及比较》
《CSS 动画简介》
《CSS 动画：animation、transition、transform、translate》

68.什么是首选最小宽度?

“首选最小宽度”，指的是元素最适合的最小宽度。

东亚文字（如中文）最小宽度为每个汉字的宽度。

西方文字最小宽度由特定的连续的英文字符单元决定。并不是所有的英文字符都会组成连续单元，一般会终止于空格（普通空格）、短
横线、问号以及其他非英文字符等。

如果想让英文字符和中文一样，每一个字符都用最小宽度单元，可以试试使用CSS中的word-break:break-all。

69.为什么 height:100%会无效?

对于普通文档流中的元素，百分比高度值要想起作用，其父级必须有一个可以生效的高度值。

原因是如果包含块的高度没有显式指定（即高度由内容决定），并且该元素不是绝对定位，则计算值为auto，因为解释成了auto，
所以无法参与计算。

使用绝对定位的元素会有计算值，即使祖先元素的height计算为auto也是如此。

70.min-width/max-width 和 min-height/max-height 属性间的覆盖规则?

（1）max-width会覆盖width，即使width是行类样式或者设置了!important。

（2）min-width会覆盖max-width，此规则发生在min-width和max-width冲突的时候。

71.内联盒模型基本概念

（1）内容区域（content area）。内容区域指一种围绕文字看不见的盒子，其大小仅受字符本身特性控制，本质上是一个字符盒子
（character box）；但是有些元素，如图片这样的替换元素，其内容显然不是文字，不存在字符盒子之类的，因此，对于这些
元素，内容区域可以看成元素自身。

（2）内联盒子（inline box）。“内联盒子”不会让内容成块显示，而是排成一行，这里的“内联盒子”实际指的就是元素的“外在盒
子”，用来决定元素是内联还是块级。该盒子又可以细分为“内联盒子”和“匿名内联盒子”两类。

（3）行框盒子（line box），每一行就是一个“行框盒子”（实线框标注），每个“行框盒子”又是由一个一个“内联盒子”组成的。

（4）包含块（containing box），由一行一行的“行框盒子”组成。

72.什么是幽灵空白节点?

“幽灵空白节点”是内联盒模型中非常重要的一个概念，具体指的是：在HTML5文档声明中，内联元素的所有解析和渲染表现就如同
每个行框盒子的前面有一个“空白节点”一样。这个“空白节点”永远透明，不占据任何宽度，看不见也无法通过脚本获取，就好像幽灵
一样，但又确确实实地存在，表现如同文本节点一样，因此，我称之为“幽灵空白节点”。

73.什么是替换元素?

通过修改某个属性值呈现的内容就可以被替换的元素就称为“替换元素”。因此，<img>、<object>、<video>、<iframe>或者表
单元素<textarea>和<input>和<select>都是典型的替换元素。

替换元素除了内容可替换这一特性以外，还有以下一些特性。

（1）内容的外观不受页面上的CSS的影响。用专业的话讲就是样式表现在CSS作用域之外。如何更改替换元素本身的外观需要
类似appearance属性，或者浏览器自身暴露的一些样式接口，

（2）有自己的尺寸。在Web中，很多替换元素在没有明确尺寸设定的情况下，其默认的尺寸（不包括边框）是300像素x150像
素，如<video>、<iframe>或者<canvas>等，也有少部分替换元素为0像素，如<img>图片，而表单元素的替换元素
的尺寸则和浏览器有关，没有明显的规律。

（3）在很多CSS属性上有自己的一套表现规则。比较具有代表性的就是vertical-align属性，对于替换元素和非替换元素，ve
rtical-align属性值的解释是不一样的。比方说vertical-align的默认值的baseline，很简单的属性值，基线之意，
被定义为字符x的下边缘，而替换元素的基线却被硬生生定义成了元素的下边缘。

（4）所有的替换元素都是内联水平元素，也就是替换元素和替换元素、替换元素和文字都是可以在一行显示的。但是，替换元素默认
的display值却是不一样的，有的是inline，有的是inline-block。

74.替换元素的计算规则?

替换元素的尺寸从内而外分为3类：固有尺寸、HTML尺寸和CSS尺寸。

（1）固有尺寸指的是替换内容原本的尺寸。例如，图片、视频作为一个独立文件存在的时候，都是有着自己的宽度和高度的。

（2）HTML尺寸只能通过HTML原生属性改变，这些HTML原生属性包括<img>的width和height属性、<input>的s
ize属性、<textarea>的cols和rows属性等。

（3）CSS尺寸特指可以通过CSS的width和height或者max-width/min-width和max-height/min-height设置的
尺寸，对应盒尺寸中的content box。

这3层结构的计算规则具体如下
（简单理解为：固有尺寸、HTML尺寸和CSS尺寸优先级依次递增。）
（1）如果没有CSS尺寸和HTML尺寸，则使用固有尺寸作为最终的宽高。

（2）如果没有CSS尺寸，则使用HTML尺寸作为最终的宽高。

（3）如果有CSS尺寸，则最终尺寸由CSS属性决定。

（4）如果“固有尺寸”含有固有的宽高比例，同时仅设置了宽度或仅设置了高度，则元素依然按照固有的宽高比例显示。

（5）如果上面的条件都不符合，则最终宽度表现为300像素，高度为150像素。

（6）内联替换元素和块级替换元素使用上面同一套尺寸计算规则。

75.content 与替换元素的关系?

content属性生成的对象称为“匿名替换元素”。

（1）我们使用content生成的文本是无法选中、无法复制的，好像设置了user select:none声明一般，但是普通元素的文本
却可以被轻松选中。同时，content生成的文本无法被屏幕阅读设备读取，也无法被搜索引擎抓取，因此，千万不要自以为是
地把重要的文本信息使用content属性生成，因为这对可访问性和SEO都很不友好。

（2）content生成的内容不能左右:empty伪类。

（3）content动态生成值无法获取。

76.margin:auto 的填充规则?

margin的'auto'可不是摆设，是具有强烈的计算意味的关键字，用来计算元素对应方向应该获得的剩余间距大小。但是触发mar
gin:auto计算有一个前提条件，就是width或height为auto时，元素是具有对应方向的自动填充特性的。

（1）如果一侧定值，一侧auto，则auto为剩余空间大小。
（2）如果两侧均是auto，则平分剩余空间。

77.margin 无效的情形

（1）display计算值inline的非替换元素的垂直margin是无效的。对于内联替换元素，垂直margin有效，并且没有ma
rgin合并的问题。

（2）表格中的<tr>和<td>元素或者设置display计算值是table-cell或table-row的元素的margin都是无效的。

（3）绝对定位元素非定位方位的margin值“无效”。

（4）定高容器的子元素的margin-bottom或者宽度定死的子元素的margin-right的定位“失效”。

78.border 的特殊性?

（1）border-width却不支持百分比。

（2）border-style的默认值是none，有一部分人可能会误以为是solid。这也是单纯设置border-width或border-col
or没有边框显示的原因。

（3）border-style:double的表现规则：双线宽度永远相等，中间间隔±1。

（4）border-color默认颜色就是color色值。

（5）默认background背景图片是相对于padding box定位的。

79.什么是基线和 x-height?

字母x的下边缘（线）就是我们的基线。

x-height指的就是小写字母x的高度，术语描述就是基线和等分线（meanline）（也称作中线，midline）之间的距离。在C
SS世界中，middle指的是基线往上1/2x-height高度。我们可以近似理解为字母x交叉点那个位置。

ex是CSS中的一个相对单位，指的是小写字母x的高度，没错，就是指x-height。ex的价值就在其副业上不受字体和字号影
响的内联元素的垂直居中对齐效果。内联元素默认是基线对齐的，而基线就是x的底部，而1ex就是一个x的高度。

80.line-height 的特殊性?

（1）对于非替换元素的纯内联元素，其可视高度完全由line-height决定。对于文本这样的纯内联元素，line-height就是高
度计算的基石，用专业说法就是指定了用来计算行框盒子高度的基础高度。

（2）内联元素的高度由固定高度和不固定高度组成，这个不固定的部分就是这里的“行距”。换句话说，line-height之所以起作
用，就是通过改变“行距”来实现的。在CSS中，“行距”分散在当前文字的上方和下方，也就是即使是第一行文字，其上方也是
有“行距”的，只不过这个“行距”的高度仅仅是完整“行距”高度的一半，因此，也被称为“半行距”。

（3）行距 = line-height - font-size。

（4）border以及line-height等传统CSS属性并没有小数像素的概念。如果标注的是文字上边距，则向下取整；如果是文字下
边距，则向上取整。

（5）对于纯文本元素，line-height直接决定了最终的高度。但是，如果同时有替换元素，则line-height只能决定最小高度。

（6）对于块级元素，line-height对其本身是没有任何作用的，我们平时改变line-height，块级元素的高度跟着变化实际上是
通过改变块级元素里面内联级别元素占据的高度实现的。

（7）line-height的默认值是normal，还支持数值、百分比值以及长度值。为数值类型时，其最终的计算值是和当前font-si
ze相乘后的值。为百分比值时，其最终的计算值是和当前font-size相乘后的值。为长度值时原意不变。

（8）如果使用数值作为line-height的属性值，那么所有的子元素继承的都是这个值；但是，如果使用百分比值或者长度值作为
属性值，那么所有的子元素继承的是最终的计算值。

（9）无论内联元素line-height如何设置，最终父级元素的高度都是由数值大的那个line-height决定的。

（10）只要有“内联盒子”在，就一定会有“行框盒子”，就是每一行内联元素外面包裹的一层看不见的盒子。然后，重点来了，在每个
“行框盒子”前面有一个宽度为0的具有该元素的字体和行高属性的看不见的“幽灵空白节点”。

81.vertical-align 的特殊性?

（1）vertical-align的默认值是baseline，即基线对齐，而基线的定义是字母x的下边缘。因此，内联元素默认都是沿着字
母x的下边缘对齐的。对于图片等替换元素，往往使用元素本身的下边缘作为基线。：一个inline-block元素，如果里面
没有内联元素，或者overflow不是visible，则该元素的基线就是其margin底边缘；否则其基线就是元素里面最后一行
内联元素的基线。

（2）vertical-align:top就是垂直上边缘对齐，如果是内联元素，则和这一行位置最高的内联元素的顶部对齐；如果display
计算值是table-cell的元素，我们不妨脑补成<td>元素，则和<tr>元素上边缘对齐。

（3）vertical-align:middle是中间对齐，对于内联元素，元素的垂直中心点和行框盒子基线往上1/2x-height处对齐。对
于table-cell元素，单元格填充盒子相对于外面的表格行居中对齐。

（4）vertical-align支持数值属性，根据数值的不同，相对于基线往上或往下偏移，如果是负值，往下偏移，如果是正值，往上
偏移。

（5）vertical-align属性的百分比值则是相对于line-height的计算值计算的。

（6）vertical-align起作用是有前提条件的，这个前提条件就是：只能应用于内联元素以及display值为table-cell的元
素。

（7）table-cell元素设置vertical-align垂直对齐的是子元素，但是其作用的并不是子元素，而是table-cell元素自身。

82.overflow 的特殊性?

（1）一个设置了overflow:hidden声明的元素，假设同时存在border属性和padding属性，则当子元素内容超出容器宽度
高度限制的时候，剪裁的边界是border box的内边缘，而非padding box的内边缘。

（2）HTML中有两个标签是默认可以产生滚动条的，一个是根元素<html>，另一个是文本域<textarea>。

（3）滚动条会占用容器的可用宽度或高度。

（4）元素设置了overflow:hidden声明，里面内容高度溢出的时候，滚动依然存在，仅仅滚动条不存在！

83.无依赖绝对定位是什么?

没有设置left/top/right/bottom属性值的绝对定位称为“无依赖绝对定位”。

无依赖绝对定位其定位的位置和没有设置position:absolute时候的位置相关。

84.absolute 与 overflow 的关系?

（1）如果overflow不是定位元素，同时绝对定位元素和overflow容器之间也没有定位元素，则overflow无法对absolute
元素进行剪裁。

（2）如果overflow的属性值不是hidden而是auto或者scroll，即使绝对定位元素高宽比overflow元素高宽还要大，也
都不会出现滚动条。

（3）overflow元素自身transform的时候，Chrome和Opera浏览器下的overflow剪裁是无效的。

85.clip 裁剪是什么?

所谓“可访问性隐藏”，指的是虽然内容肉眼看不见，但是其他辅助设备却能够进行识别和访问的隐藏。

clip剪裁被我称为“最佳可访问性隐藏”的另外一个原因就是，它具有更强的普遍适应性，任何元素、任何场景都可以无障碍使用。

86.relative 的特殊性?

（1）相对定位元素的left/top/right/bottom的百分比值是相对于包含块计算的，而不是自身。注意，虽然定位位移是相对自身，但是百分比值的计算值不是。

（2）top和bottom这两个垂直方向的百分比值计算跟height的百分比值是一样的，都是相对高度计算的。同时，如果包含块的高度是auto，那么计算值是0，偏移无效，也就是说，如果父元素没有设定高度或者不是“格式化高度”，那么relative类似top:20%的代码等同于top:0。

（3）当相对定位元素同时应用对立方向定位值的时候，也就是top/bottom和left/right同时使用的时候，只有一个方向的定位属性会起作用。而谁起作用则是与文档流的顺序有关的，默认的文档流是自上而下、从左往右，因此top/bottom同时使用的时候，bottom失效；left/right同时使用的时候，right失效。

87.什么是层叠上下文?

层叠上下文，英文称作stacking context，是HTML中的一个三维的概念。如果一个元素含有层叠上下文，我们可以理解为这个元
素在z轴上就“高人一等”。

层叠上下文元素有如下特性：

（1）层叠上下文的层叠水平要比普通元素高（原因后面会说明）。
（2）层叠上下文可以阻断元素的混合模式。
（3）层叠上下文可以嵌套，内部层叠上下文及其所有子元素均受制于外部的“层叠上下文”。
（4）每个层叠上下文和兄弟元素独立，也就是说，当进行层叠变化或渲染的时候，只需要考虑后代元素。
（5）每个层叠上下文是自成体系的，当元素发生层叠的时候，整个元素被认为是在父层叠上下文的层叠顺序中。

层叠上下文的创建：

（1）页面根元素天生具有层叠上下文，称为根层叠上下文。根层叠上下文指的是页面根元素，可以看成是<html>元素。因此，页面中所有的元素一定处于至少一个“层叠结界”中。

（2）对于position值为relative/absolute以及Firefox/IE浏览器（不包括Chrome浏览器）下含有position:fixed声明的定位元素，当其z-index值不是auto的时候，会创建层叠上下文。Chrome等WebKit内核浏览器下，position:fixed元素天然层叠上下文元素，无须z-index为数值。根据我的测试，目前IE和Firefox仍是老套路。

（3）其他一些CSS3属性，比如元素的opacity值不是1。

88.什么是层叠水平?

层叠水平，英文称作stacking level，决定了同一个层叠上下文中元素在z轴上的显示顺序。

显而易见，所有的元素都有层叠水平，包括层叠上下文元素，也包括普通元素。然而，对普通元素的层叠水平探讨只局限在当前层叠上
下文元素中。

89.元素的层叠顺序?

层叠顺序，英文称作 stacking order，表示元素发生层叠时有着特定的垂直显示顺序。

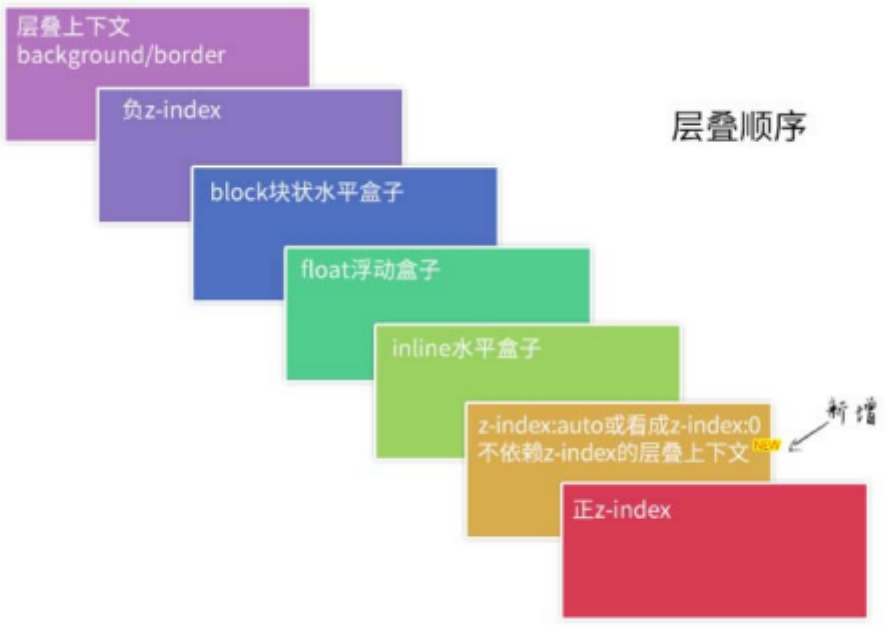

图 7-7　新的层叠顺序规则

90.层叠准则?

（1）谁大谁上：当具有明显的层叠水平标识的时候，如生效的z-index属性值，在同一个层叠上下文领域，层叠水平值大的那一个覆盖小的那一个。

（2）后来居上：当元素的层叠水平一致、层叠顺序相同的时候，在DOM流中处于后面的元素会覆盖前面的元素。

91.font-weight 的特殊性?

如果使用数值作为font-weight属性值，必须是100～900的整百数。因为这里的数值仅仅是外表长得像数值，实际上是一个具有特定含义的关键字，并且这里的数值关键字和字母关键字之间是有对应关系的。 一般使用font-weight: bold来加粗。

92.text-indent 的特殊性?

（1）text-indent仅对第一行内联盒子内容有效。

（2）非替换元素以外的display计算值为inline的内联元素设置text-indent值无效，如果计算值inline-block/inli
ne-table则会生效。

（3）<input>标签按钮text-indent值无效。

（4）<button>标签按钮text-indent值有效。

（5）text-indent的百分比值是相对于当前元素的“包含块”计算的，而不是当前元素。

93.letter-spacing 与字符间距?

letter-spacing可以用来控制字符之间的间距，这里说的“字符”包括英文字母、汉字以及空格等。

letter-spacing具有以下一些特性。

（1）继承性。
（2）默认值是normal而不是0。虽然说正常情况下，normal的计算值就是0，但两者还是有差别的，在有些场景下，letter-spacing会调整normal的计算值以实现更好的版面布局。
（3）支持负值，且值足够大的时候，会让字符形成重叠，甚至反向排列。
（4）和text-indent属性一样，无论值多大或多小，第一行一定会保留至少一个字符。
（5）支持小数值，即使0.1px也是支持的。
（6）暂不支持百分比值。

94.word-spacing 与单词间距?

letter-spacing作用于所有字符，但word-spacing仅作用于空格字符。换句话说，word-spacing的作用就是增加空格的间隙
宽度。

95.white-space 与换行和空格的控制?

white-space属性声明了如何处理元素内的空白字符，这类空白字符包括Space（空格）键、Enter（回车）键、Tab（制表符）
键产生的空白。因此，white-space可以决定图文内容是否在一行显示（回车空格是否生效），是否显示大段连续空白（空格是否
生效）等。

其属性值包括下面这些。
•normal：合并空白字符和换行符。
•pre：空白字符不合并，并且内容只在有换行符的地方换行。
•nowrap：该值和normal一样会合并空白字符，但不允许文本环绕。
•pre-wrap：空白字符不合并，并且内容只在有换行符的地方换行，同时允许文本环绕。
•pre-line：合并空白字符，但只在有换行符的地方换行，允许文本环绕。

96.隐藏元素的 background-image 到底加不加载?

相关知识点:

> 根据测试，一个元素如果display计算值为none，在IE浏览器下（IE8～IE11，更高版本不确定）依然会发送图片请求，Fire
> fox浏览器不会，至于Chrome和Safari浏览器则似乎更加智能一点：如果隐藏元素同时又设置了background-image，则图片
> 依然会去加载；如果是父元素的display计算值为none，则背景图不会请求，此时浏览器或许放心地认为这个背景图暂时是不会使
> 用的。
>
> 如果不是background-image，而是<img>元素，则设置display:none在所有浏览器下依旧都会请求图片资源。
>
> 还需要注意的是如果设置的样式没有对应的元素，则background-image也不会加载。hover情况下的background-image，在触
> 发时加载。

回答:

- (1) 元素的背景图片

-元素本身设置 display:none，会请求图片 -父级元素设置 display:none，不会请求图片 -样式没有元素使用，不会请求
-:hover 样式下，触发时请求

- (2) img 标签图片任何情况下都会请求图片

详细资料可以参考:
《CSS 控制前端图片 HTTP 请求的各种情况示例》

97.如何实现单行 / 多行文本溢出的省略（...）?

```
/*单行文本溢出*/
p {
  overflow: hidden;
  text-overflow: ellipsis;
  white-space: nowrap;
}

/*多行文本溢出*/
p {
  position: relative;
  line-height: 1.5em;
  /*高度为需要显示的行数*行高，比如这里我们显示两行，则为3*/
  height: 3em;
  overflow: hidden;
}

p:after {
  content: '...';
  position: absolute;
  bottom: 0;
  right: 0;
  background-color: #fff;
}
```

详细资料可以参考：
《【CSS/JS】如何实现单行 / 多行文本溢出的省略》
《CSS 多行文本溢出省略显示》

98.常见的元素隐藏方式?

- （1）使用 display:none;隐藏元素，渲染树不会包含该渲染对象，因此该元素不会在页面中占据位置，也不会响应绑定的监听事件。

- （2）使用 visibility:hidden;隐藏元素。元素在页面中仍占据空间，但是不会响应绑定的监听事件。

- （3）使用 opacity:0;将元素的不透明度设置为 0，以此来实现元素的隐藏。元素在页面中仍然占据空间，并且能够响应元素绑定的监听事件。

- （4）通过使用绝对定位将元素移除可视区域内，以此来实现元素的隐藏。

- （5）通过 z-index 负值，来使其他元素遮盖住该元素，以此来实现隐藏。

- （6）通过 clip/clip-path 元素裁剪的方法来实现元素的隐藏，这种方法下，元素仍在页面中占据位置，但是不会响应绑定的监听事件。

- （7）通过 transform:scale(0,0)来将元素缩放为 0，以此来实现元素的隐藏。这种方法下，元素仍在页面中占据位置，但是不会响应绑定的监听事件。

详细资料可以参考：
《CSS 隐藏元素的八种方法》

99.css 实现上下固定中间自适应布局?

```
利用绝对定位实现
body {
  padding: 0;
  margin: 0;
}

.header {
  position: absolute;
  top: 0;
  width: 100%;
  height: 100px;
  background: red;
}

.container {
  position: absolute;
  top: 100px;
  bottom: 100px;
  width: 100%;
  background: green;
}

.footer {
  position: absolute;
  bottom: 0;
  height: 100px;
  width: 100%;
  background: red;
}
```

```
利用flex布局实现
html,
body {
  height: 100%;
}

body {
  display: flex;
  padding: 0;
  margin: 0;
  flex-direction: column;
}

.header {
  height: 100px;
  background: red;
}

.container {
  flex-grow: 1;
  background: green;
}

.footer {
  height: 100px;
  background: red;
}
```

详细资料可以参考：
《css 实现上下固定中间自适应布局》

100.css 两栏布局的实现?

相关资料：

```
/*两栏布局一般指的是页面中一共两栏，左边固定，右边自适应的布局，一共有四种实现的方式。*/
/*以左边宽度固定为200px为例*/

/*（1）利用浮动，将左边元素宽度设置为200px，并且设置向左浮动。将右边元素的margin-left设置为200px，宽度设置为auto（默认为auto，撑满整个父元素）。*/
.outer {
  height: 100px;
}

.left {
  float: left;

  height: 100px;
  width: 200px;

  background: tomato;
}

.right {
  margin-left: 200px;

  width: auto;
```

```
  height: 100px;

  background: gold;
}

/*（2）第二种是利用flex布局，将左边元素宽度设置为200px。将右边的元素的放大比例设置为1。*/
.outer {
  display: flex;

  height: 100px;
}

.left {
  width: 200px;

  background: tomato;
}

.right {
  flex-grow: 1;

  background: gold;
}

/*（3）第三种是利用绝对定位布局的方式，将父级元素设置相对定位。左边元素设置为absolute定位，并且宽度设置为
200px。将右边元素的margin-left的值设置为200px。*/
.outer {
  position: relative;

  height: 100px;
}

.left {
  position: absolute;

  width: 200px;
  height: 100px;

  background: tomato;
}

.right {
  margin-left: 200px;
  height: 100px;

  background: gold;
}

/*（4）第四种还是利用绝对定位的方式，将父级元素设置为相对定位。左边元素宽度设置为200px，右边元素设置为绝对定位，左边定位为200px，其余方向定位为0。*/
.outer {
  position: relative;

  height: 100px;
}

.left {
```

```
  width: 200px;
  height: 100px;

  background: tomato;
}

.right {
  position: absolute;

  top: 0;
  right: 0;
  bottom: 0;
  left: 200px;

  background: gold;
}
```

《两栏布局 demo 展示》

回答：

两栏布局一般指的是页面中一共两栏，左边固定，右边自适应的布局，一共有四种实现的方式。

以左边宽度固定为 200px 为例

- （1）利用浮动，将左边元素宽度设置为 200px，并且设置向左浮动。将右边元素的 margin-left 设置为 200px，宽度设置为 auto（默认为 auto，撑满整个父元素）。

- （2）第二种是利用flex布局，将左边元素宽度设置为200px。将右边的元素的放大比例设置为1。

- （3）第三种是利用绝对定位布局的方式，将父级元素设置相对定位。左边元素设置为 absolute 定位，并且宽度设置为 200px。将右边元素的 margin-left 的值设置为 200px。

- （4）第四种还是利用绝对定位的方式，将父级元素设置为相对定位。左边元素宽度设置为 200px，右边元素设置为绝对定位，左边定位为 200px，其余方向定位为 0。

101.css 三栏布局的实现?

相关资料：

```
/*三栏布局一般指的是页面中一共有三栏，左右两栏宽度固定，中间自适应的布局，一共有五种实现方式。

这里以左边宽度固定为100px，右边宽度固定为200px为例。*/

/*（1）利用绝对定位的方式，左右两栏设置为绝对定位，中间设置对应方向大小的margin的值。*/
.outer {
  position: relative;
  height: 100px;
}

.left {
  position: absolute;

  width: 100px;
  height: 100px;
  background: tomato;
}

.right {
```

```
  position: absolute;
  top: 0;
  right: 0;

  width: 200px;
  height: 100px;
  background: gold;
}

.center {
  margin-left: 100px;
  margin-right: 200px;
  height: 100px;
  background: lightgreen;
}

/*（2）利用flex布局的方式，左右两栏的宽度分别设置为100px和200px，中间一栏增长系数设置为1*/
.outer {
  display: flex;
  height: 100px;
}

.left {
  width: 100px;
  background: tomato;
}

.right {
  width: 200px;
  background: gold;
}

.center {
  flex-grow: 1;
  background: lightgreen;
}

/*（3）利用浮动的方式，左右两栏设置固定大小，并设置对应方向的浮动。中间一栏设置左右两个方向的margin值，注意这种方式，中间一栏必须放到最后。*/
.outer {
  height: 100px;
}

.left {
  float: left;
  width: 100px;
  height: 100px;
  background: tomato;
}

.right {
  float: right;
  width: 200px;
  height: 100px;
  background: gold;
}

.center {
```

```
  height: 100px;
  margin-left: 100px;
  margin-right: 200px;
  background: lightgreen;
}

/*（4）圣杯布局，利用浮动和负边距来实现。父级元素设置左右的 padding，三列均设置向左浮动，中间一列放在最前面，宽度设置为父级元素的宽度，因此后面两列都被挤到了下一行，通过设置 margin 负值将其移动到上一行，再利用相对定位，定位到两边。*/
.outer{
  /*  为左右栏腾出空间  */
  padding-left: 100px;
  padding-right: 200px;
}
.left{
  float: left;
  width: 100px;
  height: 100px;
  background-color: red;
  margin-left: -100%;
  position: relative;
  left: -100px;
}
.center{
  float: left;
  width: 100%;
  height: 100px;
  background-color: #bfa;
}
.right{
  float: left;
  width: 200px;
  height: 100px;
  background-color: orange;
  margin-left: -200px;
  position: relative;
  left: 200px;
}

/*（5）双飞翼布局，双飞翼布局相对于圣杯布局来说，左右位置的保留是通过中间列的 margin 值来实现的，而不是通过父元
素的 padding 来实现的。本质上来说，也是通过浮动和外边距负值来实现的。*/

.outer {
  height: 100px;
}

.left {
  float: left;
  margin-left: -100%;

  width: 100px;
  height: 100px;
  background: tomato;
}

.right {
  float: left;
```

```
  margin-left: -200px;

  width: 200px;
  height: 100px;
  background: gold;
}

.wrapper {
  float: left;

  width: 100%;
  height: 100px;
  background: lightgreen;
}

.center {
  margin-left: 100px;
  margin-right: 200px;
  height: 100px;
}
```

《三栏布局 demo 展示》

回答:

三栏布局一般指的是页面中一共有三栏，左右两栏宽度固定，中间自适应的布局，一共有五种实现方式。

这里以左边宽度固定为100px，右边宽度固定为200px为例。

（1）利用绝对定位的方式，左右两栏设置为绝对定位，中间设置对应方向大小的margin的值。

（2）利用flex布局的方式，左右两栏的宽度分别设置为100px和200px，中间一栏增长系数设置为1

（3）利用浮动的方式，左右两栏设置固定大小，并设置对应方向的浮动。中间一栏设置左右两个方向的margin值，注意这种方式，中间一栏必须放到最后。

（4）圣杯布局，利用浮动和负边距来实现。父级元素设置左右的padding，三列均设置向左浮动，中间一列放在最前面，宽度设置为父级元素的宽度，因此后面两列都被挤到了下一行，通过设置margin负值将其移动到上一行，再利用相对定位，定位到两边。圣杯布局中间列的宽度不能小于左边列的宽度，否则左边列上不去，而双飞翼布局则不存在这个问题。

（5）双飞翼布局，双飞翼布局相对于圣杯布局来说，左右位置的保留是通过中间列的margin值来实现的，而不是通过父元素的padding来实现的。本质上来说，也是通过浮动和外边距负值来实现的。

102.实现一个宽高自适应的正方形

```
/*1.第一种方式是利用vw来实现*/
.square {
  width: 10vw;
  height: 10vw;
  background: tomato;
}

/*2.第二种方式是利用元素的margin/padding百分比是相对父元素width的性质来实现*/
.square {
  width: 20%;
```

```
  height: 0;
  padding-top: 20%;
  background: orange;
}

/*3.第三种方式是利用子元素的margin-top的值来实现的*/
.square {
  width: 30%;
  overflow: hidden;
  background: yellow;
}

.square::after {
  content: '';
  display: block;
  margin-top: 100%;
}
```

《自适应正方形 demo 展示》

103.实现一个三角形

```
/*三角形的实现原理是利用了元素边框连接处的等分原理。*/
.triangle {
  width: 0;
  height: 0;
  border-width: 100px;
  border-style: solid;
  border-color: transparent transparent tomato;
}
```

《三角形 demo 展示》

104.一个自适应矩形，水平垂直居中，且宽高比为 2:1

```
/*实现原理参考自适应正方形和水平居中方式*/
.box {
  position: absolute;
  top: 0;
  right: 0;
  left: 0;
  bottom: 0;
  margin: auto;

  width: 20%;
  height: 10vw;
  background: tomato;
}
```

105.你知道 CSS 中不同属性设置为百分比%时对应的计算基准?

```
公式：当前元素某CSS属性值 = 基准 * 对应的百分比
元素的 position 为 relative 和 absolute 时，top和bottom、left和right基准分别为包含块的height、width
元素的 position 为 fixed 时，top和bottom、left和right基准分别为初始包含块（也就是视口）的height、width，移动设备较为复杂，基准为 Layout viewport 的 height、width
```

元素的 height 和 width 设置为百分比时，基准分别为包含块的 height 和 width
元素的 margin 和 padding 设置为百分比时，基准为包含块的 width（易错）
元素的 border-width，不支持百分比
元素的 text-indent，基准为包含块的 width

元素的 border-radius，基准为分别为自身的height、width
元素的 background-size，基准为分别为自身的height、width
元素的 translateX、translateY，基准为分别为自身的height、width
元素的 line-height，基准为自身的 font-size

元素的 font-size，基准为父元素字体