# 前端面试指南 — css 篇面试题

css面试题基本在前端面试中，主要在笔试，笔试过程中，如果看没什么人，忘记了，可以拿手机出现偷偷的查，一般给笔试后在会议室没人在，所以可以偷偷查，那如果没有笔试，只有面试，那就看你背的程度了，css一定要准备好，如果笔试都过不了，那就会被直接pass掉

# 考点 1：初始化样式

## 问题： 什么是 CSS 初始化？并说说为什么要初始化 CSS 样式？

CSS 初始化是指：开发者对浏览器的默认样式进行重置。

1、浏览器差异

因为浏览器的兼容问题，不同的样式会有默认初始样式，margin 和padding，下划线等等，浏览器不同， 数值还不一样，如果直接写样式，会出现差异，布局出现错乱，所以要初始化样式，达到统一的布局。

2、提高编码质量

初始化CSS 后，可以让开发者省去很多写单独兼容的代码，减少代码体积，节约网页下载时间；还会使得我们开发网页内容时更加方便简洁，开发者就不用考虑太多基础样式的兼容问题了。

**考点 2：margin重合问题**

## 问题：有遇到过margin重合问题吗

相邻两个盒子垂直方向上的margin会发生重叠，只会取比较大的margin

1. 设置padding代替margin
2. 设置float
3. 设置overflow
4. 设置position：absolute 绝对定位
5. 设置display： inline-block 。

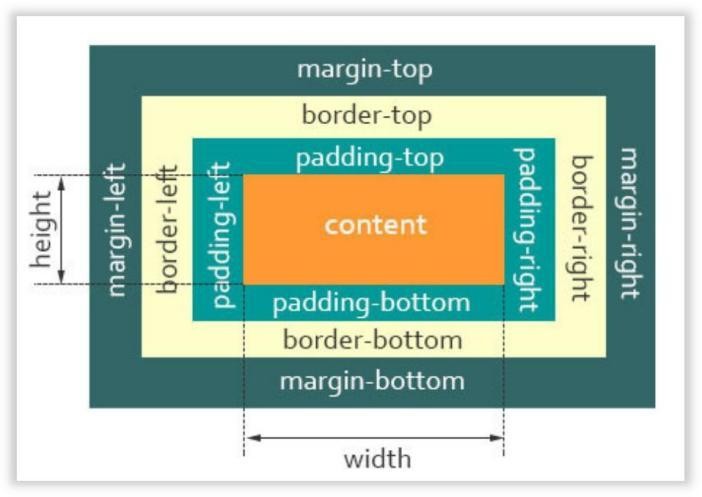
# 考点 3：关于盒模型

## 问题：请说出你对盒模型的理解。

理解：我们可以把页面上所有的html 元素都可以看作是盒子，也就是说整个 html 页面就是由无数个盒子通过特定的布局结合在一起的。

每个盒子由 4 部分构成：外边距margin、内边距 padding、内容 content、边框 border。如下图：

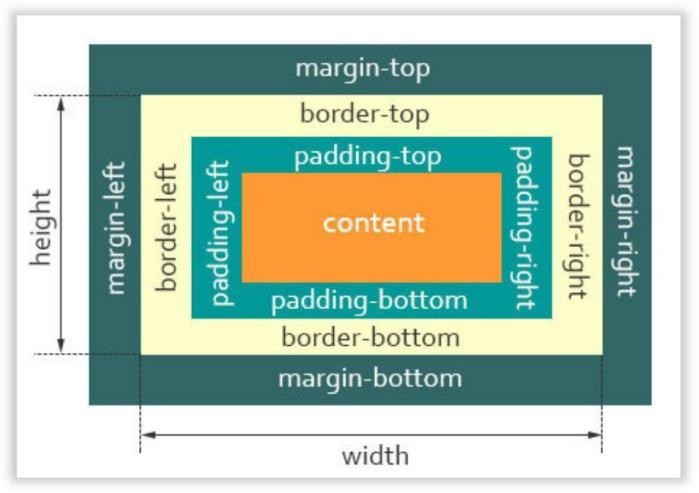
而目前市面上存在 2 中盒模型：标准盒模型 和IE 盒子模型，它俩对计算宽度和高度的不同。先说标准盒模型，也就是 W3C 规定的盒子模型。



在标准模式下：

盒子总宽度 = width + padding + border + margin。盒子总高度 = height + padding + border + margin。

也就是（划重点啦!!!!）我们设置的 width/height 只是内容 content（上图橙色的部分）的宽/高度，不包含 padding 和 border 值 。

反过来，我们再看看 IE 盒子模型，又称怪异盒模型。

也就是（划重点啦!!!!）我们设置的 width/height 只是内容 content（上图橙色的部分）的宽/高度，不包含 padding 和 border 值 。

反过来，我们再看看 IE 盒子模型，又称怪异盒模型。

在IE 盒子模型下：

盒子总宽度 = width+ margin = (内容区宽度 + padding + border) + margin。

也就是我们设置的 width/height 包含了 padding 和 border 值（如上图橙色+浅绿色+黄色三部分）。总结：标准盒子模型和 IE 盒子模型的差别就在于宽度和高度包含的范围不同。

注意啦！加分项来了：拓展到 CSS3 的box-sizing 新特性

CSS3 新增了box-sizing 属性，它可以让开发者指定盒子模型种类。

值为content-box：padding 和border 不算在我们设置的width/height 里面。也就是说，指定盒子模型为标准盒模型。

值为 border-box：padding 和border 算在了我们设置的 width/height 里面。也就是说，也就是指定盒子模型为 IE 盒子模型。

# 考点 4：隐藏元素

## 问题：请说说隐藏一个元素的几种方法，以及它们之间的区别。（display none visibility hidden区别？）

第 1 种：设置元素的 display 为none 是最常用的隐藏元素的方法。

.hide {

display:none;

}

将元素设置为 display:none 后，元素在页面上将彻底消失，元素本来占有的空间就会被其他元素占有，也就是说它会导致浏览器的重排和重绘。

第 2 种：设置元素的 visibility 为hidden。

.hidden{

visibility:hidden

}

和display:none 的区别在于，元素在页面消失后，其占据的空间依旧会保留着，所以它只会导致浏览器重绘而不会重排，适用于那些元素隐藏后不希望页面布局会发生变化的场景。

第 3 种：设置元素的 opacity 为 0。

.transparent

{opacity:0;

}

这种方法和 visibility:hidden 的一个共同点是元素隐藏后依旧占据着空间，但我设置透明度为 0 后，元素只是隐身了，它依旧存在页面中。

注意：加分项来啦！！！拓展到三种隐藏元素的对于点击事件绑定的区别 如果对这 3 中不同隐藏元素的 dom 节点绑定其点击事件，会有下面的结论：

* display:none：元素彻底消失，很显然不会触发其点击事件
* visibility:hidden：设置元素的 visibility 后无法触发点击事件，说明这种方法元素也是消失了，只是依然占据着页面空间。
* opacity:0：可以触发点击事件，原因也很简单，设置元素透明度为 0 后，元素只是相对于人眼不存在而已，对浏览器来说，它还是存在的，所以可以触发点击事件。

# 考点 5：清除浮动

## 问题：请你说出你用过清除浮动的几种办法，以及它们的优缺点。

第 1 种：在最后一个浮动标签后，新加一个标签，给其设置 clear：both；

.father{

width: 400px;

}

.big{

width: 200px; height: 200px; float: left;

}

.small{

width: 120px; height: 120px; float: left;

}

.clear{

clear:both;

}

<div class="father">

<div class="big">big</div>

<div class="small">small</div>

<div class="clear">clear 标签法</div>

</div>

使用这种办法，如果我们清除了浮动，父元素自动检测子盒子最高的高度，然后与其同高。优点：通俗易懂，方便。

缺点：添加无意义标签，语义化差，所以不建议使用。

第 2 种：给父元素添加 overflow:hidden。

.father{

width: 400px; overflow: hidden;

}

.big{

width: 200px; height: 200px; float: left;

}

优点：代码简洁。

缺点：如果内容增多的时候容易造成不会自动换行导致内容被隐藏掉，无法显示要溢出的元素，因为设置了overflow：hidden，看具体情况来决定是否使用。

第 3 种：使用 after 伪元素清除浮动。

.clearfix:after{/\*伪元素是行内元素 正常浏览器清除浮动方法\*/ content: "";

display: block; height: 0; clear:both; visibility: hidden;

}

.clearfix{

\*zoom: 1; /\*ie6 清除浮动的方式 \*号只有IE6-IE7 执行，其他浏览器不执行\*/

}

<div class="father clearfix">

<div class="big">big</div>

<div class="small">small</div>

</div>

优点：符合闭合浮动思想，结构语义化正确.

缺点：ie6-7 不支持伪元素 :after，使用zoom:1 触发hasLayout。整体相对来说，推荐使用 after 伪元素来清除浮动。

# 考点 6：CSS3 新特性

## 问题：请说出你使用过哪些 CSS3 新特性？

border-radius：用于实现圆角。box-shadow：用于实现阴影。

border-image：用于实现边框图片。text-shadow：用于实现文字阴影。

linear-gradient：用于实现背景线性渐变。

transform：用于实现元素变形，包括旋转 rotate、扭曲 skew、缩放 scale 和移动 translate 以及矩阵变形matrix。

transition：用于在一定的时间区间内，把元素从一种状态平滑地过渡到另一种状态。animation：结合@keyframes 创建实现动画。

## 问题：请说出 CSS3 新增伪类有哪些？分别代表什么含义？

p:first-of-type 选择属于其父元素的首个元素。p:last-of-type 选择属于其父元素的最后元素。p:only-of-type 选择属于其父元素唯一的元素。p:only-child 选择属于其父元素的唯一子元素。p:nth-child(2) 选择属于其父元素的第二个子元素。

:enabled :disabled 表单控件的禁用状态。

:checked 单选框或复选框被选中。

# 考点 7：样式优先级

## 问题：CSS 样式的优先级是怎么样的？

!important > 内联样式 > ID 选择器 > 伪类 > 属性选择器 > 类选择器 > 标签选择器 > 通配符（\*） 。如非特殊情况，慎用!important。因为使用!important 会扰乱原本层叠和权重产生正常的作用顺序，使后期维护带来麻烦。

# 考点 8：单位对比

## 问题：请说出你常用的几个 CSS 单位，比如：px，em，rem ,vw等。

px：最常用的，它是相对于显示器屏幕分辨率而言的。

优缺点：比较稳定和精确，但在浏览器中放大或缩放浏览页面时会出现页面混乱的情况。

em：相对单位，基准点为父节点字体的大小，如果自身定义了 font-size 按自身来计算（浏览器默认字体是 16px），整个页面内 1em 不是一个固定的值。

优缺点：em 的值并不是固定的，它会继承父级元素的字体大小。

rem：相对单位，基于 root 元素，即根据html 元素的大小来计算，不受容器本身字体大小的影响，全部根据html 的字体大小重新计算。设定根元素<html> 的font-size 属性，默认为 16px，那么 1rem = 16px。 优缺点：这个单位可谓集相对大小和绝对大小的优点于一身，通过它既可以做到只修改根元素就成比例地调整所有字体大小，又可以避免字体大小逐层复合的连锁反应。

Vh：1vw等于屏幕可视区高度的百分之一。

Vw：1vw等于屏幕可视区宽度的百分之一。

# 考点 9：技巧题

## 问题：用纯 CSS 创建一个三角形的原理是什么？

技巧：采用的是相邻边框链接处的均分原理，将元素的宽高设为 0，只设置 border , 将任意三条边隐藏掉

（颜色设为 transparent ），剩下的就是一个三角形。

<div class="triangle"></div>

.triangle{

border-color: transparent transparent green transparent; border-style: solid;

border-width: 0px 300px 300px 300px; height: 0px;

width: 0px;

}

## 问题：怎么让 Chrome 支持小于 12px 的文字？

技巧：针对 chrome 浏览器，加webkit 前缀，用transform:scale()这个属性进行放缩。

span{

font-size: 12px; display: inline-block;

-webkit-transform:scale(0.8);

}

<span>10px 的字体效果</span>

## 问题：CSS 如何去除inline-block 元素间的间距？

我们使用 CSS 把非inline-block 的元素改为inline-block 的时候，元素之间就会产生默认的间距。

技巧：使用font-size:0。这个方法基本上可以解决大部分浏览器下inline-block 元素之间的间，不过Chrome 浏览器默认有最小字体大小限制，因为，考虑到兼容性，我们还需要添加：-webkit-text-size-adjust。

.space {

font-size: 0;

-webkit-text-size-adjust:none;

}

## 问题：CSS 如何实现单行文本溢出显示省略号？

技巧：用 text-overflow:ellipsis 属性，还需要加宽度 width 属来兼容部分浏览器。

p {

width: 100px; overflow: hidden; white-space: nowrap; text-overflow: ellipsis;

}

<p>这里再长一点就要变成省略号了</p>

## 问题：CSS 如何实现多行文本溢出显示省略号？（注意是多行）

技巧：用 box-orient 和line-clamp 属性，还需要加 width 和overflow。

p {

display: -webkit-box;

-webkit-box-orient: vertical;

-webkit-line-clamp: 3; overflow: hidden; width: 200px;

}

<p>这里再长一点就要变成多行文字+省略号了</p>

## 问题：如何修改 chrome 记住密码后自动填充表单的黄色背景 ？

技巧：用-webkit-autofill 属性。

input:-webkit-autofill, textarea:-webkit-autofill, select:-webkit-autofill

{background-color: rgb(250, 255, 189); /\* #FAFFBD; \*/ background-image: none;

color: rgb(0, 0, 0);

}

## 问题：让页面里的字体变清晰，变细用 CSS 怎么做？

技巧：用 -webkit-font-smoothing 属性。

-webkit-font-smoothing: antialiased;

注 意 : -webkit-font-smoothing 在 window 系 统 下 没 有 起 作 用 ， 但 是 在 IOS 设 备 上 起 作 用

-webkit-font-smoothing：antialiased 是最佳的，灰度平滑。

## 问题：display:inline-block后出现缝隙的解决方案

1. 设置父元素font-size：0；
2. flex布局
3. Float:left

# 考点 10：编程题

## 问题：编程题：如何用 CSS 实现多列等高布局，即需要每个列的高度能与最高的那个列等齐？

方法一：使用 padding 和 margin 实现。

/\* 公共样式 \*/ ul {

list-style: none; padding: 0;

margin: 0;

}

ul li {

background: #0b88ec;

}

ul li p {

padding: 10px; margin: 0; color: #fff;

}

/\* padding margin 实现样式 \*/

.test-1 ul {

overflow: hidden;

}

.test-1 li {

float: left; width: 33.33%;

padding-bottom: 9999px; margin: 0 0 -9999px 0;

}

<!-- padding margin 实现 -->

<div class="test-1">

<ul>

<li>

<p>大师傅大师傅</p>

</li>

<li>

<p>

啊手动阀第三方士大夫啥的</p>

</li>

<li>

<p>

十分士大夫士大夫第三方士大夫士大夫

</p>

</li>

</ul>

</div>

方法二：使用 display:table 实现。

/\* 公共样式 \*/ ul {

list-style: none; padding: 0;

margin: 0;

}

ul li {

background: #0b88ec;

}

ul li p {

padding: 10px; margin: 0; color: #fff;

}

/\* table 实现样式 \*/

.test-2 ul {

display: table;

}

.test-2 li {

display: table-cell; width: 33.33%;

}

<!-- table 实现 -->

<div class="test-2">

<ul>

<li>

<p>

撒大苏打萨达萨达是</p>

</li>

<li>

<p>

撒旦撒旦撒旦撒旦阿斯顿萨达是。

</p>

</li>

<li>

<p>

阿萨的撒旦撒旦阿斯顿阿松大实打实的。

</p>

</li>

</ul>

</div>

方法三：使用 flex 实现。

/\* 公共样式 \*/ ul {

list-style: none; padding: 0;

margin: 0;

}

ul li {

background: #0b88ec;

}

ul li p {

padding: 10px; margin: 0; color: #fff;

}

/\* flex 实现样式 \*/

.test-3 ul {

display: box;

display: -webkit-flex; display: flex;

}

.test-3 li {

-webkit-box-flex: 1;

-webkit-flex: 1;

flex: 1;

width: 0;

}

<!-- flex 实现 -->

<div class="test-3">

<ul>

<li>

<p>

阿斯顿萨达实打实的撒旦撒旦。

</p>

</li>

<li>

<p>

阿松大阿松大撒旦撒打算d。

</p>

</li>

<li>

<p>

撒旦撒啊倒萨的撒的撒旦阿松大阿萨

</p>

</li>

</ul>

</div>

## 问题：如何用 CSS 实现一个不定宽高的盒子能够水平垂直居中？

核心是使用 justify-content 和 align-items 实现。

.wrap{

width: 500px; height: 500px; margin: 0 auto; border:1px solid red;

/\*子元素水平居中\*/ justify-content: center;

/\*子元素垂直居中\*/ align-items: center; display: flex;

}

.box{

display: inline-block; background: pink; padding: 20px; border-radius: 8px;

}

<div class="wrap">

<div class="box">

不定宽高的水平垂直居中

</div>

</div>

# 其他可能遇到的问题

## 问题：移动端的1px在有些机型上看上去比较粗细，虽然写1px但看上去的效果可能是2px（移动端的1px适配问题）

伪元素+transform方法，原理是吧原先元素的border去掉，然后利用:before或者:after重做border，并transform的scale缩小一半，原先的元素相对定位，新做的border绝对定位。

.hairlines li{ position: relative; border:none; margin-top: 10px;

}

.hairlines li:after{ content: ''; position: absolute; left: 0;

bottom: 0; background: #cccccc;

width: 100%;height: 1px;-webkit-transform: scaleY(0.5);transform: scaleY(0.5);-webkit-transform- origin: 0 0;transform-origin: 0 0;

}

## 问题：请戳出ie6/7下特有的cssbug

（1）、li边距“无故” 增加

设置ul的显示形式为\*display:inline-block;即可，前面加\*是只 针对IE6/IE7有效

（2）、IE6 不支持min-height属性，但它却认为height就是最小高度

使用ie6不支持但其余浏览器支持的属性!important。

（3）、Overflow:

在IE6/7中，overflow无法正确的隐藏有相对定位position:relative;的子元素。解决方法就是给外包容器.wrap加上position:relative;。

（4）、border：none 在IE6不起作用： 写成border：0 就可以了，

（5）、100%高度

在IE6下，如果要给元素定义100%高度，必须要明确定义它的父级元素的高度，如果你需要给元素定义满屏的高度，就得先给html和body定义 height:100%;。

（6）、双边距 Bug

当元素浮动时，IE6会错误的的把浮动方式的margin值双倍计算，给float的元素添加一个display：inline

（7）、IE6 绝对定位的元素1px 间距bug

当绝对定位的父元素或宽度为奇数时，bottom和right会多出现1px，

解决方案，针对IE6进行hack处理