套用上面的算法，依次求出 `A` `B` `C` `D` 的值：

- 因为没有内联样式 ，所以 A = 0
- ID选择器总共出现了1次， B = 1
- 类选择器出现了1次， 属性选择器出现了0次，伪类选择器出现0次，所以 C = (1 + 0 + 0) = 1
- 标签选择器出现了3次， 伪元素出现了0次，所以 D = (3 + 0) = 3

上面算出的 `A` 、 `B` 、 `C` 、 `D` 可以简记作： `(0, 1, 1, 3)`

知道了优先级是如何计算之后，就来看看比较规则：

- 从左往右依次进行比较 ，较大者优先级更高
- 如果相等，则继续往右移动一位进行比较
- 如果4位全部相等，则后面的会覆盖前面的

经过上面的优先级计算规则，我们知道内联样式的优先级最高，如果外部样式需要覆盖内联样式，就需要使用 `!important`

6.3. 继承属性

在 `css` 中，继承是指的是给父元素设置一些属性，后代元素会自动拥有这些属性

关于继承属性，可以分成：

- 字体系列属性

CSS | 复制代码

```
font:组合字体
font-family:规定元素的字体系列
font-weight:设置字体的粗细
font-size:设置字体的尺寸
font-style:定义字体的风格
font-variant:偏大或偏小的字体
```

- 文本系列属性

CSS | 复制代码

```
text-indent：文本缩进
text-align：文本水平对刘
line-height：行高
word-spacing：增加或减少单词间的空白
letter-spacing：增加或减少字符间的空白
text-transform：控制文本大小写
direction：规定文本的书写方向
color：文本颜色
```

- 元素可见性

CSS | 复制代码

```
visibility
```

- 表格布局属性

CSS | 复制代码

```
caption-side：定位表格标题位置
border-collapse：合并表格边框
border-spacing：设置相邻单元格的边框间的距离
empty-cells：单元格的边框的出现与消失
table-layout：表格的宽度由什么决定
```

- 列表属性

CSS | 复制代码

```
list-style-type：文字前面的小点点样式
list-style-position：小点点位置
list-style：以上的属性可通过这属性集合
```

- 引用

CSS | 复制代码

```
quotes：设置嵌套引用的引号类型
```

- 光标属性

```
1  cursor: 箭头可以变成需要的形状
```

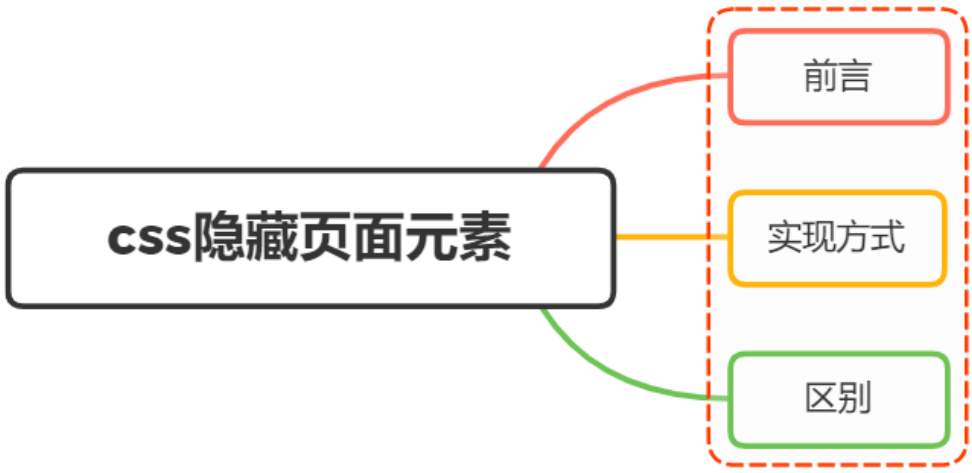
继承中比较特殊的几点：

- a 标签的字体颜色不能被继承
- h1-h6标签字体的大下也是不能被继承的

6.3.1. 无继承的属性

- display
- 文本属性：vertical-align、text-decoration
- 盒子模型的属性：宽度、高度、内外边距、边框等
- 背景属性：背景图片、颜色、位置等
- 定位属性：浮动、清除浮动、定位position等
- 生成内容属性：content、counter-reset、counter-increment
- 轮廓样式属性：outline-style、outline-width、outline-color、outline
- 页面样式属性：size、page-break-before、page-break-after

7. css中，有哪些方式可以隐藏页面元素？区别？



7.1. 前言

在平常的样式排版中，我们经常遇到将某个模块隐藏的场景

通过 `css` 隐藏元素的方法有很多种，它们看起来实现的效果是一致的

但实际上每一种方法都有一丝轻微的不同，这些不同决定了在一些特定场合下使用哪一种方法

7.2. 实现方式

通过 `css` 实现隐藏元素方法有如下：

- display:none
- visibility:hidden
- opacity:0
- 设置height、width模型属性为0
- position:absolute
- clip-path

7.2.1. display:none

设置元素的 `display` 为 `none` 是最常用的隐藏元素的方法

CSS | 复制代码

```css
.hide {
    display:none;
}
```

将元素设置为 `display:none` 后，元素在页面上将彻底消失

元素本身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

消失后，自身绑定的事件不会触发，也不会有过渡效果

特点：元素不可见，不占据空间，无法响应点击事件

7.2.2. visibility:hidden

设置元素的 `visibility` 为 `hidden` 也是一种常用的隐藏元素的方法

从页面上仅仅是隐藏该元素，DOM结果均会存在，只是当时在一个不可见的状态，不会触发重排，但是会触发重绘

```css
.hidden{
    visibility:hidden
}
```

给人的效果是隐藏了，所以他自身的事件不会触发

特点：元素不可见，占据页面空间，无法响应点击事件

7.2.3. opacity:0

`opacity` 属性表示元素的透明度，将元素的透明度设置为0后，在我们用户眼中，元素也是隐藏的

不会引发重排，一般情况下也会引发重绘

> 如果利用 animation 动画，对 opacity 做变化（animation会默认触发GPU加速），则只会触发 GPU 层面的 composite，不会触发重绘

```css
.transparent {
    opacity:0;
}
```

由于其仍然是存在于页面上的，所以他自身的的事件仍然是可以触发的，但被他遮挡的元素是不能触发其事件的

需要注意的是：其子元素不能设置opacity来达到显示的效果

特点：改变元素透明度，元素不可见，占据页面空间，可以响应点击事件

7.2.4. 设置height、width属性为0

将元素的 `margin` ， `border` ， `padding` ， `height` 和 `width` 等影响元素盒模型的属性设置成0，如果元素内有子元素或内容，还应该设置其 `overflow:hidden` 来隐藏其子元素

```css
.hiddenBox {
    margin:0;
    border:0;
    padding:0;
    height:0;
    width:0;
    overflow:hidden;
}
```

特点：元素不可见，不占据页面空间，无法响应点击事件

7.2.5. position:absolute

将元素移出可视区域

```css
.hide {
   position: absolute;
   top: -9999px;
   left: -9999px;
}
```

特点：元素不可见，不影响页面布局

7.2.6. clip-path

通过裁剪的形式

```css
.hide {
  clip-path: polygon(0px 0px,0px 0px,0px 0px,0px 0px);
}
```

特点：元素不可见，占据页面空间，无法响应点击事件

7.2.7. 小结

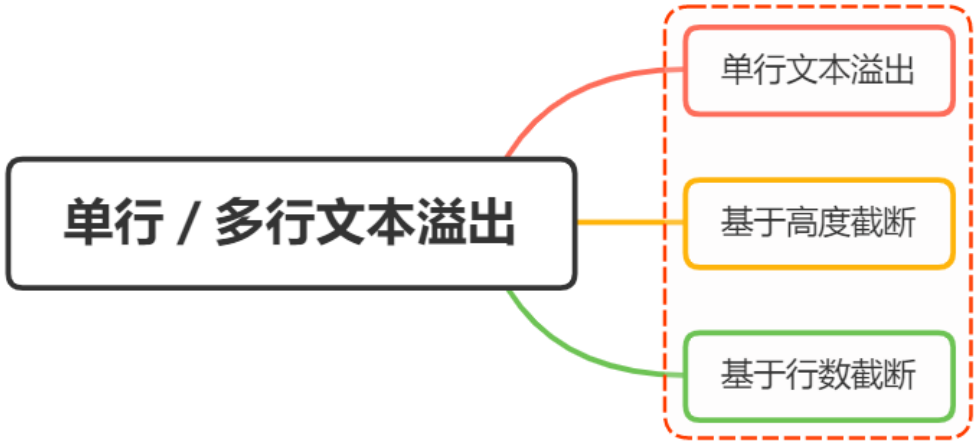
最常用的还是 `display:none` 和 `visibility:hidden` ，其他的方式只能认为是奇招，它们的真正用途并不是用于隐藏元素，所以并不推荐使用它们

7.3. 区别

关于 `display: none` 、 `visibility: hidden` 、 `opacity: 0` 的区别，如下表所示：

	display: none	visibility: hidden	opacity: 0
页面中	不存在	存在	存在
重排	会	不会	不会
重绘	会	会	不一定
自身绑定事件	不触发	不触发	可触发
transition	不支持	支持	支持
子元素可复原	不能	能	不能
被遮挡的元素可触发事件	能	能	不能

8. 如何实现单行 / 多行文本溢出的省略样式?



8.1. 前言

在日常开发展示页面，如果一段文本的数量过长，受制于元素宽度的因素，有可能不能完全显示，为了提高用户的使用体验，这个时候就需要我们把溢出的文本显示成省略号

对于文本的溢出，我们可以分成两种形式：

- 单行文本溢出

- 多行文本溢出

8.2. 实现方式

8.2.1. 单行文本溢出省略

理解也很简单，即文本在一行内显示，超出部分以省略号的形式展现

实现方式也很简单，涉及的 `css` 属性有：

- text-overflow：规定当文本溢出时，显示省略符号来代表被修剪的文本
- white-space：设置文字在一行显示，不能换行
- overflow：文字长度超出限定宽度，则隐藏超出的内容

`overflow` 设为 `hidden` ，普通情况用在块级元素的外层隐藏内部溢出元素，或者配合下面两个属性实现文本溢出省略

`white-space:nowrap` ，作用是设置文本不换行，是 `overflow:hidden` 和 `text-overflow:ellipsis` 生效的基础

`text-overflow` 属性值有如下：

- clip：当对象内文本溢出部分裁切掉
- ellipsis：当对象内文本溢出时显示省略标记（...）

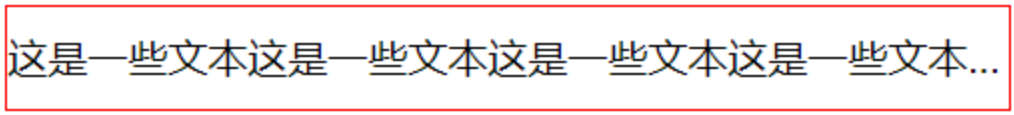
`text-overflow` 只有在设置了 `overflow:hidden` 和 `white-space:nowrap` 才能够生效的

举个例子

HTML | 复制代码

```
<style>
    p{
        overflow: hidden;
        line-height: 40px;
        width:400px;
        height:40px;
        border:1px solid red;
        text-overflow: ellipsis;
        white-space: nowrap;
    }
</style>
<p 这是一些文本这是一些文本这是一些文本这是一些文本这是一些文本这是一些文本这是一些文本这是一些文本这是一些文本这是一些文本这是一些文本</p >
```

效果如下：



可以看到，设置单行文本溢出较为简单，并且省略号显示的位置较好

8.2.2. 多行文本溢出省略

多行文本溢出的时候，我们可以分为两种情况：

- 基于高度截断
- 基于行数截断

8.2.2.1. 基于高度截断

8.2.2.2. 伪元素 + 定位

核心的 `css` 代码结构如下：

- position: relative：为伪元素绝对定位
- overflow: hidden：文本溢出限定的宽度就隐藏内容）
- position: absolute：给省略号绝对定位
- line-height: 20px：结合元素高度,高度固定的情况下,设定行高, 控制显示行数
- height: 40px：设定当前元素高度
- ::after {} ： 设置省略号样式

代码如下所示：

HTML | 复制代码

```html
<style>
    .demo {
        position: relative;
        line-height: 20px;
        height: 40px;
        overflow: hidden;
    }
    .demo::after {
        content: "...";
        position: absolute;
        bottom: 0;
        right: 0;
        padding: 0 20px 0 10px;
    }
</style>

<body>
    <div class='demo'>这是一段很长的文本</div>
</body>
```

实现原理很好理解，就是通过伪元素绝对定位到行尾并遮住文字，再通过 `overflow: hidden` 隐藏多余文字

这种实现具有以下优点：

- 兼容性好，对各大主流浏览器有好的支持
- 响应式截断，根据不同宽度做出调整

一般文本存在英文的时候，可以设置 `word-break: break-all` 使一个单词能够在换行时进行拆分

8.2.2.3. 基于行数截断

纯 `css` 实现也非常简单，核心的 `css` 代码如下：

- -webkit-line-clamp: 2：用来限制在一个块元素显示的文本的行数，为了实现该效果，它需要组合其他的WebKit属性）
- display: -webkit-box：和1结合使用，将对象作为弹性伸缩盒子模型显示
- -webkit-box-orient: vertical：和1结合使用 ，设置或检索伸缩盒对象的子元素的排列方式
- overflow: hidden：文本溢出限定的宽度就隐藏内容
- text-overflow: ellipsis：多行文本的情况下，用省略号“...”隐藏溢出范围的文本

HTML | 复制代码

```html
<style>
    p {
        width: 400px;
        border-radius: 1px solid red;
        -webkit-line-clamp: 2;
        display: -webkit-box;
        -webkit-box-orient: vertical;
        overflow: hidden;
        text-overflow: ellipsis;
    }
</style>
<p>
    这是一些文本这是一些文本这是一些文本这是一些文本这是一些文本
    这是一些文本这是一些文本这是一些文本这是一些文本这是一些文本
</p >
```

可以看到，上述使用了 `webkit` 的 `CSS` 属性扩展，所以兼容浏览器范围是 `PC` 端的 `webkit` 内核的浏览器，由于移动端大多数是使用 `webkit` ，所以移动端常用该形式

需要注意的是，如果文本为一段很长的英文或者数字，则需要添加 `word-wrap: break-word` 属性

还能通过使用 `javascript` 实现配合 `css` ，实现代码如下所示：

css结构如下：

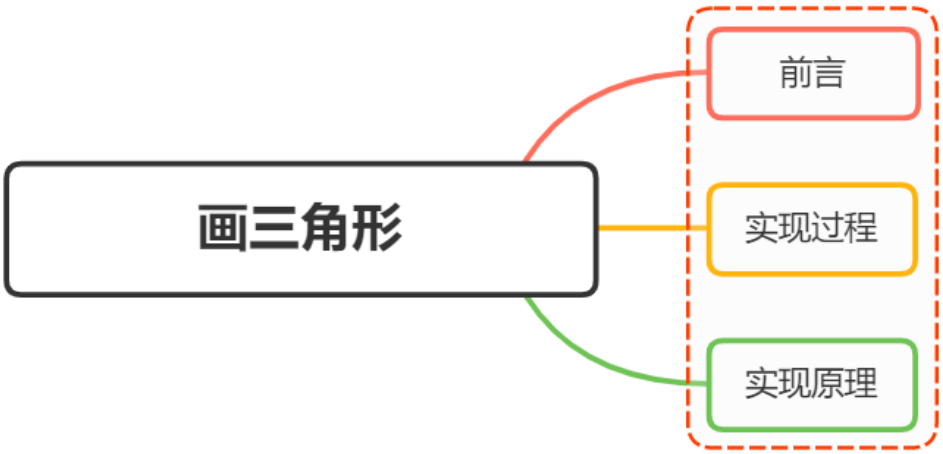
CSS | 复制代码

```css
p {
    position: relative;
    width: 400px;
    line-height: 20px;
    overflow: hidden;

}
.p-after:after{
    content: "...";
    position: absolute;
    bottom: 0;
    right: 0;
    padding-left: 40px;
    background: -webkit-linear-gradient(left, transparent, #fff 55%);
    background: -moz-linear-gradient(left, transparent, #fff 55%);
    background: -o-linear-gradient(left, transparent, #fff 55%);
    background: linear-gradient(to right, transparent, #fff 55%);
}
```

javascript代码如下：

```javascript
$(function(){
 //获取文本的行高，并获取文本的高度，假设我们规定的行数是五行，那么对超过行数的部分进行限制高度，并加上省略号
   $('p').each(function(i, obj){
        var lineHeight = parseInt($(this).css("line-height"));
        var height = parseInt($(this).height());
        if((height / lineHeight) >3 ){
            $(this).addClass("p-after")
            $(this).css("height","60px");
        }else{
            $(this).removeClass("p-after");
        }
    });
})
```

9. CSS如何画一个三角形？原理是什么？



9.1. 前言

在前端开发的时候，我们有时候会需要用到一个三角形的形状，比如地址选择或者播放器里面播放按钮

通常情况下，我们会使用图片或者 `svg` 去完成三角形效果图，但如果单纯使用 `css` 如何完成一个三角形呢?

实现过程似乎也并不困难，通过边框就可完成

9.2. 实现过程

在以前也讲过盒子模型，默认情况下是一个矩形，实现也很简单

HTML | 复制代码

```
<style>
    .border {
        width: 50px;
        height: 50px;
        border: 2px solid;
        border-color: #96ceb4 #ffeead #d9534f #ffad60;
    }
</style>
<div class="border"></div>
```

效果如下图所示：

将 `border` 设置 `50px` ，效果图如下所示：

白色区域则为 `width` 、 `height` ，这时候只需要你将白色区域部分宽高逐渐变小，最终变为0，则变成如下图所示：

这时候就已经能够看到4个不同颜色的三角形，如果需要下方三角形，只需要将上、左、右边框设置为0就可以得到下方的红色三角形

但这种方式，虽然视觉上是实现了三角形，但实际上，隐藏的部分任然占据部分高度，需要将上方的宽度去掉

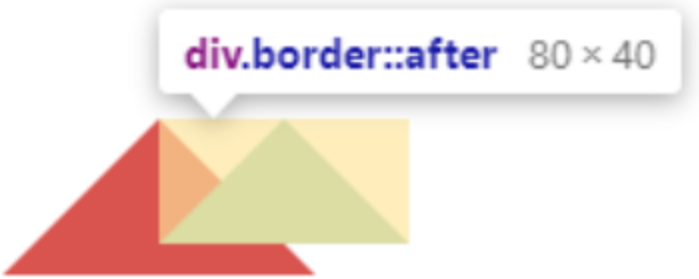
最终实现代码如下：

```css
.border {
    width: 0;
    height: 0;
    border-style:solid;
    border-width: 0 50px 50px;
    border-color: transparent transparent #d9534f;
}
```

如果想要实现一个只有边框是空心的三角形，由于这里不能再使用 `border` 属性，所以最直接的方法是利用伪类新建一个小一点的三角形定位上去

```css
.border {
    width: 0;
    height: 0;
    border-style:solid;
    border-width: 0 50px 50px;
    border-color: transparent transparent #d9534f;
    position: relative;
}
.border:after{
    content: '';
    border-style:solid;
    border-width: 0 40px 40px;
    border-color: transparent transparent #96ceb4;
    position: absolute;
    top: 0;
    left: 0;
}
```

效果图如下所示：



伪类元素定位参照对象的内容区域宽高都为0，则内容区域即可以理解成中心一点，所以伪元素相对中心这点定位

将元素定位进行微调以及改变颜色，就能够完成下方效果图：

最终代码如下：

CSS | 复制代码

```css
.border:after {
    content: '';
    border-style: solid;
    border-width: 0 40px 40px;
    border-color: transparent transparent #96ceb4;
    position: absolute;
    top: 6px;
    left: -40px;
}
```

9.3. 原理分析

可以看到，边框是实现三角形的部分，边框实际上并不是一个直线，如果我们将四条边设置不同的颜色，将边框逐渐放大，可以得到每条边框都是一个梯形

当分别取消边框的时候，发现下面几种情况：

- 取消一条边的时候，与这条边相邻的两条边的接触部分会变成直的
- 当仅有邻边时， 两个边会变成对分的三角
- 当保留边没有其他接触时，极限情况所有东西都会消失

通过上图的变化规则，利用旋转、隐藏，以及设置内容宽高等属性，就能够实现其他类型的三角形

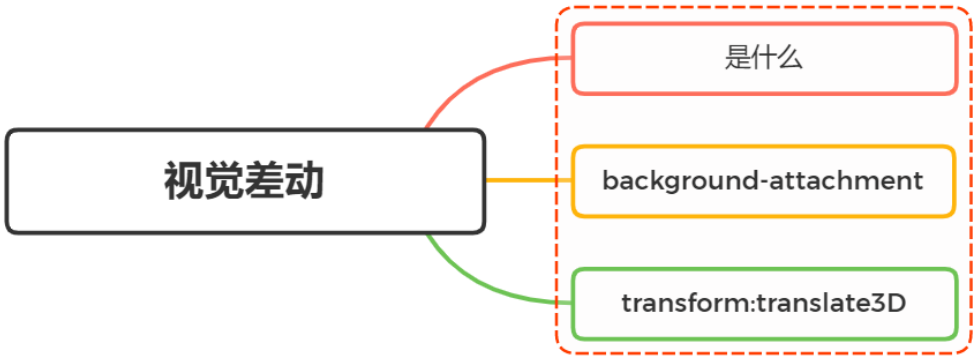
如设置直角三角形，如上图倒数第三行实现过程，我们就能知道整个实现原理

实现代码如下：

CSS | 复制代码

```css
.box {
    /* 内部大小 */
    width: 0px;
    height: 0px;
    /* 边框大小 只设置两条边*/
    border-top: #4285f4 solid;
    border-right: transparent solid;
    border-width: 85px;
    /* 其他设置 */
    margin: 50px;
}
```

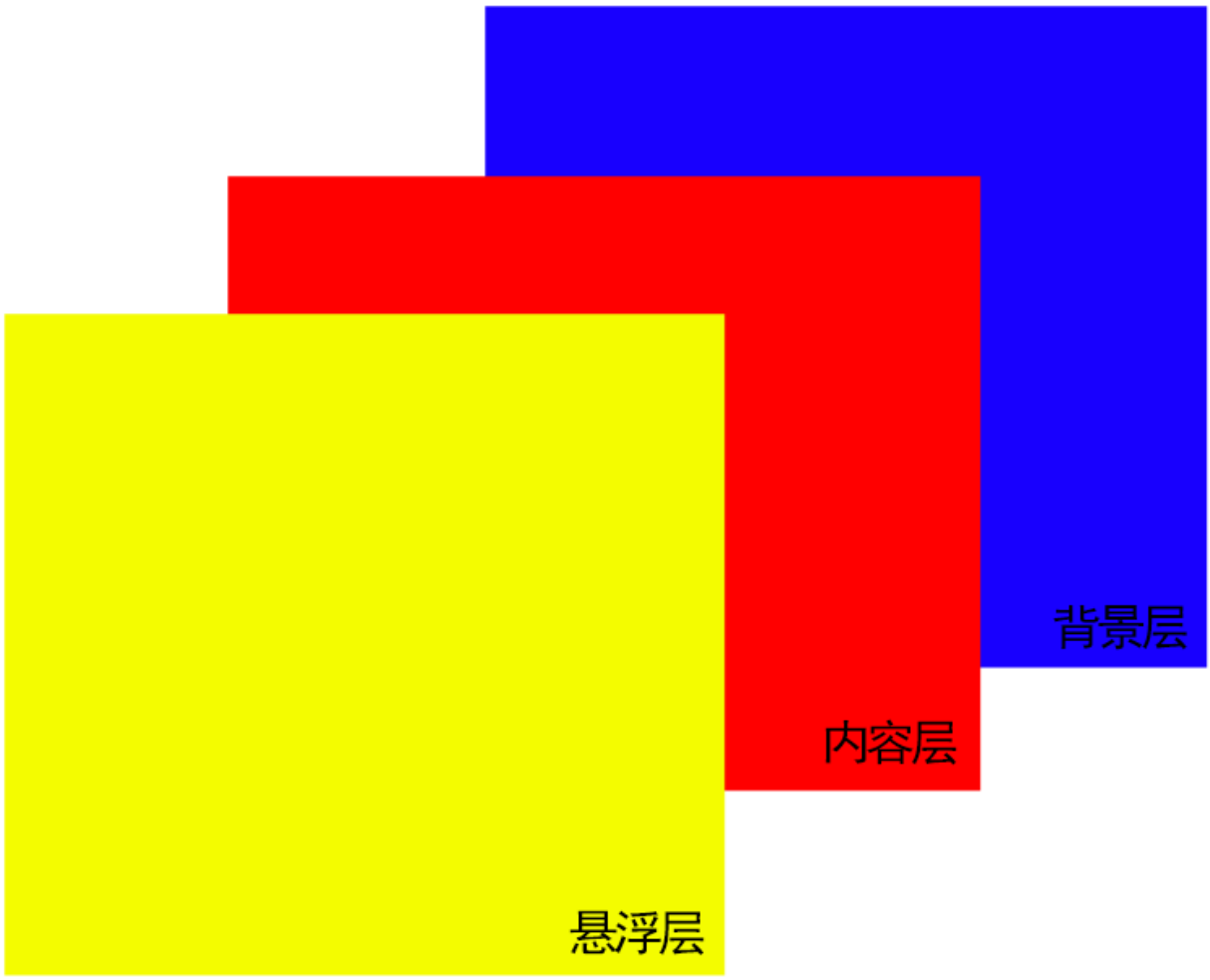
10. 如何使用css完成视差滚动效果?



10.1. 是什么

视差滚动（Parallax Scrolling）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

我们可以把网页解刨成：背景层、内容层、悬浮层



当滚动鼠标滑轮的时候，各个图层以不同的速度移动，形成视觉差的效果

10.2. 实现方式

使用 `css` 形式实现视觉差滚动效果的方式有：

- background-attachment
- transform:translate3D

10.2.1. background-attachment

作用是设置背景图像是否固定或者随着页面的其余部分滚动

值分别有如下：

- scroll：默认值，背景图像会随着页面其余部分的滚动而移动
- fixed：当页面的其余部分滚动时，背景图像不会移动
- inherit：继承父元素background-attachment属性的值

完成滚动视觉差就需要将 `background-attachment` 属性设置为 `fixed` ，让背景相对于视口固定。及时一个元素有滚动机制，背景也不会随着元素的内容而滚动

也就是说，背景一开始就已经被固定在初始的位置

核心的 `css` 代码如下：

CSS | 复制代码

```css
section {
    height: 100vh;
}

.g-img {
    background-image: url(...);
    background-attachment: fixed;
    background-size: cover;
    background-position: center center;
}
```

整体例子如下：

HTML | 复制代码

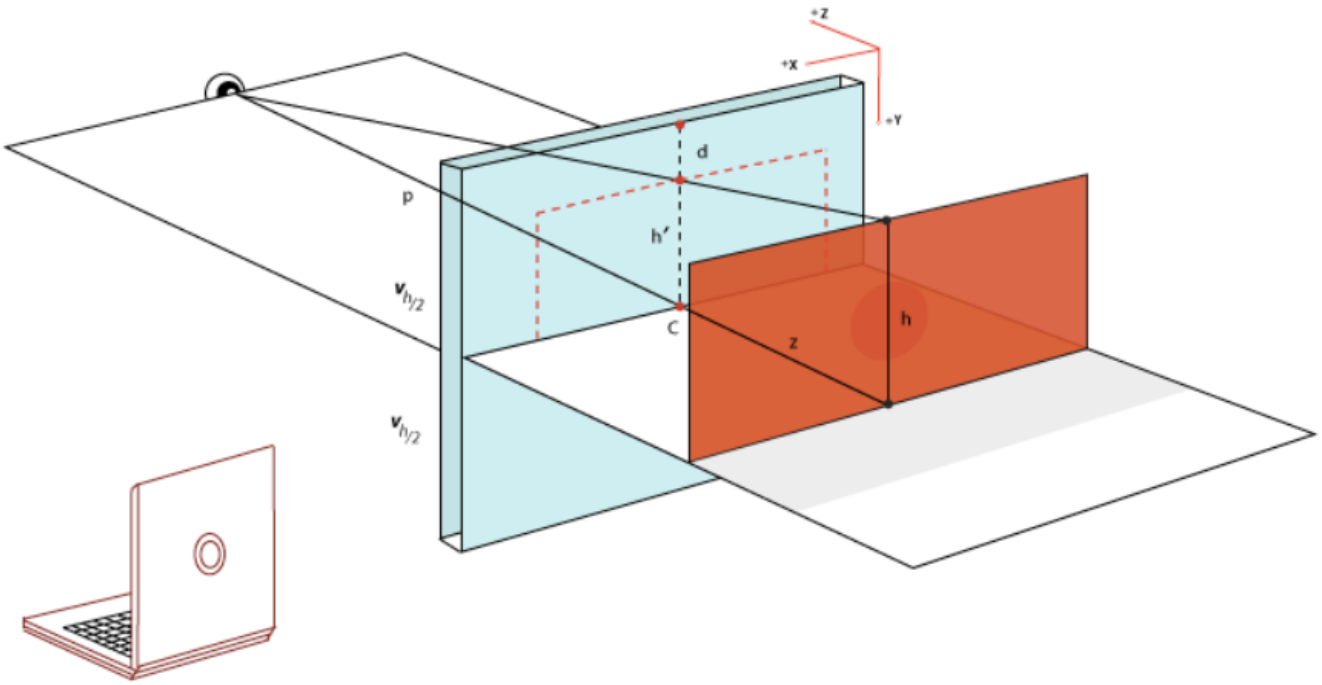
```
<style>
div {
            height: 100vh;
            background: rgba(0, 0, 0, .7);
            color: #fff;
            line-height: 100vh;
            text-align: center;
            font-size: 20vh;
        }

        .a-img1 {
            background-image: url(https://images.pexels.com/photos/1097491/pexels-photo-1097491.jpeg);
            background-attachment: fixed;
            background-size: cover;
            background-position: center center;
        }

        .a-img2 {
            background-image: url(https://images.pexels.com/photos/2437299/pexels-photo-2437299.jpeg);
            background-attachment: fixed;
            background-size: cover;
            background-position: center center;
        }

        .a-img3 {
            background-image: url(https://images.pexels.com/photos/1005417/pexels-photo-1005417.jpeg);
            background-attachment: fixed;
            background-size: cover;
            background-position: center center;
        }
</style>
 <div class="a-text">1</div>
    <div class="a-img1">2</div>
    <div class="a-text">3</div>
    <div class="a-img2">4</div>
    <div class="a-text">5</div>
    <div class="a-img3">6</div>
    <div class="a-text">7</div>
```

10.2.2. transform:translate3D

同样，让我们先来看一下两个概念 `transform` 和 `perspective`：

- transform: css3 属性，可以对元素进行变换(2d/3d)，包括平移 translate,旋转 rotate,缩放 scale,等等
- perspective: css3 属性，当元素涉及 3d 变换时，perspective 可以定义我们眼睛看到的 3d 立体效果，即空间感

`3D` 视角示意图如下所示：



举个例子：