

CSS渲染：CSS是如何绘制颜色的？

winter 2019-04-27



你好，我是 winter，今天我们来学习一下 CSS 的渲染相关的属性。

我们在布局篇讲到，CSS 的一些属性决定了盒的位置，那么今天我讲到的属性就决定了盒如何被渲染。

按照惯例，还是先从简单得讲起，首先我们来讲讲颜色。

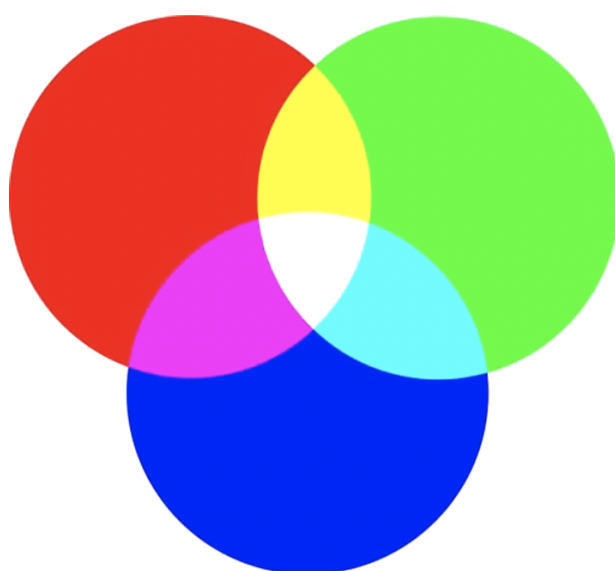
颜色的原理

首先我们来讲讲颜色，最常见的颜色相关的属性就是 `color` 和 `background-color`。

这两个属性没什么好讲的，它们分别表示文字颜色和背景颜色，我们这里重点讲讲颜色值。

RGB 颜色

我们在计算机中，最常见的颜色表示法是 RGB 颜色，它符合光谱三原色理论：红、绿、蓝三种颜色的光可以构成所有的颜色。



为什么是这三种颜色呢？这跟人类的视神经系统相关，人类的视觉神经分别有对红、绿、蓝三种颜色敏感的类型。

顺便提一下，人类对红色的感觉最为敏感，所以危险信号提示一般会选择红色；而红绿色盲的人，就是红和绿两种神经缺失一种。其它的动物视觉跟人可能不太一样，比如皮皮虾拥有 16 种视锥细胞，所以我猜它们看到的世界一定特别精彩。

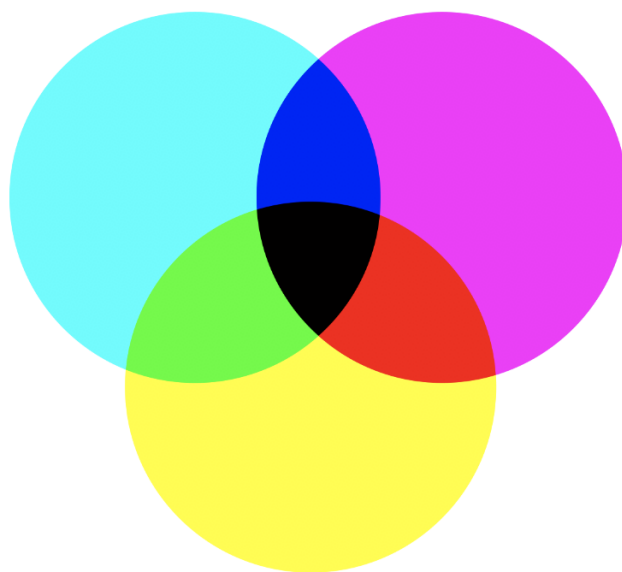
现代计算机中多用 0 – 255 的数字表示每一种颜色，这正好占据了一个字节，每一个颜色就占据三个字节。

这个数字远远超过了人体的分辨能力，因此，上世纪 90 年代刚推出这样的颜色系统的时候，它被称作真彩色。早年间还有更节约空间，但是精度更低的 16 色、256 色、8 位色和 16 位色表示法。

红绿蓝三种颜色的光混合起来就是白光，没有光就是黑暗，所以在 RGB 表示法中，三色数值最大表示白色，三色数值为 0 表示黑色。

CMYK 颜色

如果你上过小学美术课，应该听过“红黄蓝”三原色的说法，这好像跟我们说的不太一样。实际上是这样的，颜料显示颜色的原理是它吸收了所有别的颜色的光，只反射一种颜色，所以颜料三原色其实是红、绿、蓝的补色，也就是：品红、黄、青。因为它们跟红、黄、蓝相近，所以有了这样的说法。



在印刷行业，使用的就是这样的三原色（品红、黄、青）来调配油墨，这种颜色的表示法叫做 CMYK，它用一个四元组来表示颜色。

你一定会好奇，为什么它比三原色多了一种，其实答案并不复杂，在印刷行业中，黑色颜料价格最低，而品红、黄、青颜料价格较贵，如果要用三原色调配黑色，经济上是不划算的，所以印刷时会单独指定黑色。

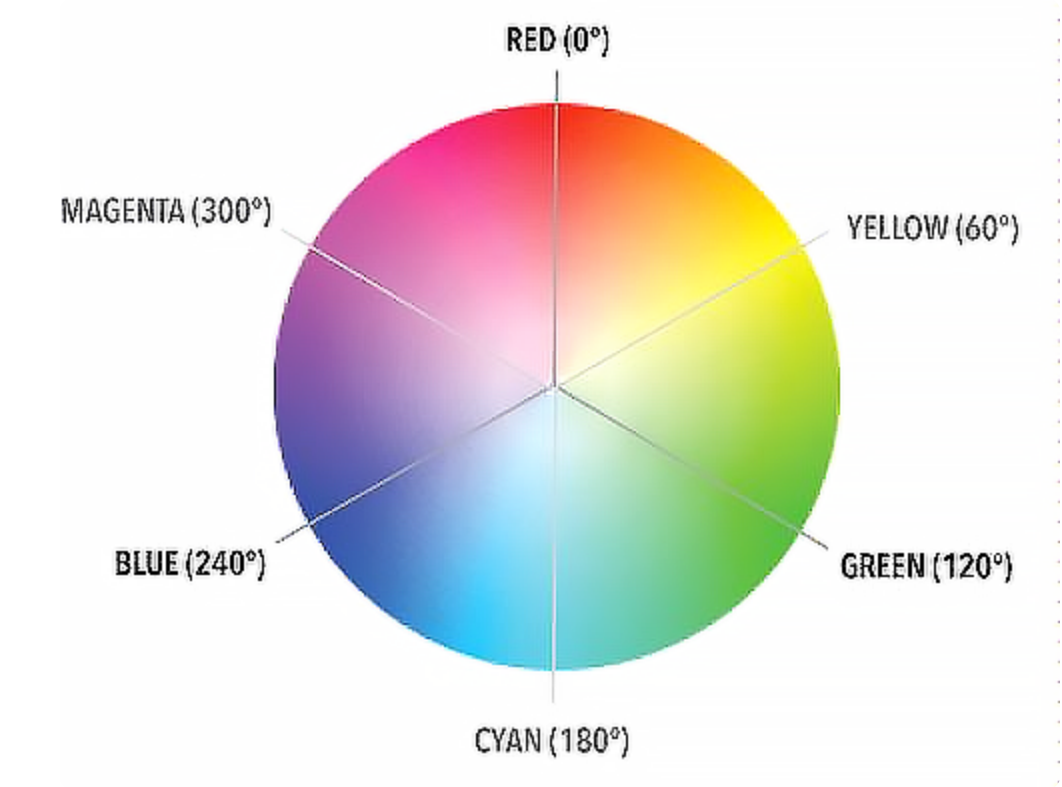
对 CMYK 颜色表示法来说，同一种颜色会有多种表示方案，但是我们参考印刷行业的习惯，会尽量优先使用黑色。

HSL 颜色

好了，讲了这么多，其实还没有涉及今天的主角：HSL 颜色。接下来我们就讲一讲。

我们刚才讲的颜色是从人类的视觉原理建模，应该说是十分科学了。但是，人类对颜色的认识却并非来自自己的神经系统，当我们把阳光散射，可以得到七色光：红橙黄绿蓝靛紫，实际上，阳光接近白光，它包含了各种颜色的光，它散射之后，应该是个基本连续的。这说明对人的感知来说，颜色远远大于红、绿、蓝。

因此，HSL 这样的颜色模型被设计出来了，它用一个值来表示人类认知中的颜色，我们用专业的术语叫做色相（H）。加上颜色的纯度（S）和明度（L），就构成了一种颜色的表示。



在这里，我需要特别推荐 HSL 颜色，因为它是一种语义化的颜色。当我们对一张图片改变色相时，人们感知到的是“图片的颜色变了”。这里先容我卖个关子，具体的例子待我们讲完了渐变再看。

其它颜色

接下来我们讲一讲 RGBA，RGBA 是代表 Red（红色）、Green（绿色）、Blue（蓝色）和 Alpha 的色彩空间。RGBA 颜色被用来表示带透明度的颜色，实际上，Alpha 通道类似

一种颜色值的保留字。在 CSS 中，Alpha 通道被用于透明度，所以我们的颜色表示被称作 RGBA，而不是 RGBO（Opacity）。

为了方便使用，CSS 还规定了名称型的颜色，它内置了大量（140 种）的颜色名称。不过这里我要挑出两个颜色来讲一讲：金（gold）和银（silver）。

如果你使用过这两个颜色，你会发现，金（gold）和银（silver）的视觉表现跟我们想象中的金色和银色相差甚远。与其被叫做金色和银色，它们看起来更像是难看的暗黄色和浅灰色。


为什么会这样呢？在人类天然的色彩认知中，实际上混杂了很多其它因素，金色和银色不仅仅是一种颜色，它还意味着一定的镜面反光程度，在同样的光照条件下，金属会呈现出更亮的色彩，这并非是用一个色值可以描述的，这就引出了我们接下来要讲的渐变。

渐变

在 CSS 中，background-image 这样的属性，可以设为渐变。CSS 中支持两种渐变，一种是线性渐变，一种是放射性渐变，我们先了解一下它们的基本用法：

线性渐变的写法是：

```
1 linear-gradient(direction, color-stop1, color-stop2, ...);
```

 复制代码

这里的 direction 可以是方向，也可以是具体的角度。例如：

to bottom

to top

to left

to right

to bottom left

to bottom right

to top left

to top right

120deg

3.14rad

以上这些都是合理的方向取值。

color-stop 是一个颜色和一个区段，例如：

rgba(255,0,0,0)


orange

yellow 10%

green 20%

lime 28px

我们组合一下，产生一个“真正的金色”的背景：

 复制代码


```
1 <style>
2 #grad1 {
3     height: 200px;
4     background: linear-gradient(45deg, gold 10%, yellow 50%, gold 90%);
5 }
6 </style>
7 <div id="grad1"></div>
```

放射性渐变需要一个中心点和若干个颜色：

 复制代码

```
1 radial-gradient(shape size at position, start-color, ..., last-color);
```

当我们应用的每一种颜色都是 HSL 颜色时，就产生了一些非常有趣的效果，比如，我们可以通过变量来调整一个按钮的风格：

 复制代码

```
1 <style>
```

```
2  .button {
3      display: inline-block;
4      outline: none;
5      cursor: pointer;
6      text-align: center;
7      text-decoration: none;
8      font: 14px/100% Arial, Helvetica, sans-serif;
9      padding: .5em 2em .55em;
10     text-shadow: 0 1px 1px rgba(0,0,0,.3);
11     border-radius: .5em;
12     box-shadow: 0 1px 2px rgba(0,0,0,.2);
13     color: white;
14     border: solid 1px ;
15 }
16
17 </style>
18 <div class="button orange">123</div>
19
```

 复制代码

```
1  var btn = document.querySelector(".button");
2  var h = 25;
3  setInterval(function(){
4      h ++;
5      h = h % 360;
6      btn.style.borderColor=`hsl(${h}, 95%, 45%)`
7      btn.style.background=`linear-gradient(to bottom, hsl(${h},95%,54.1%), hsl(
8  },100));
```

形状

CSS 中的很多属性还会产生形状，比如我们常见的属性：

border

box-shadow

border-radius

这些产生形状的属性非常有趣，我们也能看到很多利用它们来产生的 CSS 黑魔法。然而，这里我有一个相反的建议，我们仅仅把它们用于基本的用途，把 border 用于边框、把阴影用于阴影，把圆角用于圆角，所有其它的场景，都有一个更好的替代品：datauri+svg。

总结

今天我们介绍了 CSS 中渲染相关的属性：颜色和形状。

我们重点介绍了 CSS 的颜色系统，从颜色基本原理讲解了 RGB 颜色、CMYK 颜色和 HSV 颜色，我们还讲解了 Alpha 通道。

接下来我们又讲了颜色的一个重要应用：渐变，我们可以把渐变看作是一个更复杂的颜色，它非常实用，能够用渐变绘制很多的图像。

最后我们讲解了形状相关的属性，以及 SVG 应用的一个小技巧。

思考题



折衷鹦鹉是一种可爱的鸟类，但是雄性折衷鹦鹉居然是跟雌性颜色不一样！你能用 JavaScript 和 canvas，把这只雄性折衷鹦鹉变成跟雌性一样可爱的红色吗？

© 版权归极客邦科技所有，未经许可不得传播售卖。 页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

阿成

怎么说呢，要想完美的转换... 好难... 仅靠单像素颜色来识别出鹦鹉的轮廓还是不太可行... 也许把周围像素的颜色考虑进去是个办法... 不过这图挺大的...

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <style type="text/css">
    .bird {
      width: 400px;
      height: calc(1440 * 400 / 1920 * 1px);
    }
    canvas.bird {
      background: #ccc;
    }
  </style>
</head>
<body>
  
  <canvas id="canvas" width="1920" height="1440" class="bird"></canvas>

  <script type="text/javascript">
    let canvas = document.getElementById('canvas')
    let ctx = canvas.getContext('2d')
    let img = document.getElementById('img')
    img.addEventListener('load', () => {
      ctx.drawImage(img, 0, 0)

      let imageData = ctx.getImageData(0, 0, canvas.width, canvas.height)
      let data = imageData.data

      for (let i = 0; i < data.length; i += 4) {
        if (isBird(data, i, canvas.width, canvas.height)) {
          ;[data[i], data[i + 1]] = [data[i + 1] * 1.2, data[i]]
        }
      }

      ctx.putImageData(imageData, 0, 0)
    })

    function isBird (data, i, width, height) {
      let r = data[i]
```

```
let g = data[i + 1]
let b = data[i + 2]

let [h, s, l] = rgb2hsl(r, g, b)
return h < 200 && h > 80 && s > 0.23 && l < 0.84
}
```

```
function rgb2hsl (r, g, b) {
  let r1 = r / 255
  let g1 = g / 255
  let b1 = b / 255

  let min = Math.min(r1, g1, b1)
  let max = Math.max(r1, g1, b1)

  let l = (min + max) / 2
  let s
  let h

  if (l < 0.5) {
    s = (max - min) / (max + min)
  } else {
    s = (max - min) / (2 - max - min)
  }

  if (max === r1) {
    h = (r1 - b1) / (max - min)
  } else if (max === g1) {
    h = 2 + (b1 - r1) / (max - min)
  } else if (max === b1) {
    h = 4 + (r1 - g1) / (max - min)
  }

  h *= 60

  while (h < 0) {
    h += 360
  }

  return [h, s, l]
}
```

</script>

</body>

</html>



sugar

我来给个答案吧，乍一看 感觉需要用到很多cv领域的技术，模式识别判定轮廓，然后根据色值不同进行greenToRed转译。后来想了一下，这明明是前端的课程嘛，按cv的解决方案，难道还要把opencv编译到wasm里？转念一想，其实css滤镜就能做这事儿，试了试 几行css代码能做的事 在opencv要引一大堆库 改一大堆参数了

2019-11-11



5



Stinson

CMYK，为什么有K，一方面是成本，另一方面是因为自然界的CMY不能合成纯黑的颜色，所以需要纯黑

2019-07-21



3



KL宇

```
<!DOCTYPE html>
<html>
  <head>
    <title>鹦鹉变成红色</title>
    <script>
      function loadImg() {
        let img = new Image
        img.src = "yingwu.jpg"

        img.onload = function () {
          drawCanvas(img)
        }
      }

      function drawCanvas(img) {
        let canvas = document.getElementById('canvas')
        canvas.width = img.width
        canvas.height = img.height
        let context = canvas.getContext('2d')

        context.drawImage(img, 0, 0);
```

```
//context.clearRect(200,432, 1110, 670);  
let sectionImg = context.getImageData(200, 432, 1110, 75 0);  
let imgData = sectionImg.data;  
  
for(let i =1; i < imgData.length;i += 4) {  
  if (imgData[i-1] < imgData[i]) {  
    let temp = imgData[i-1]  
    imgData[i - 1] = imgData[i]  
    imgData[i] = temp  
  }  
  
}  
  
context.putImageData(sectionImg, 200, 432); // 复制代码  
  
}  
document.addEventListener('DOMContentLoaded', function(){  
  loadImg()  
})  
  
</script>  
</head>  
<body>  
  <canvas id="canvas"></canvas>  
</body>  
</html>
```

2020-03-01



2



一路向北

老师在末尾提到了border、box-shadow、border-radius可以产生一些CSS黑魔法，而不是只定义边框、阴影和圆角，这里我很想知道，除了基本用途，他们可以产生什么样的黑魔法呢？有没有一些推荐的资料呢？

2020-01-03



2



1



Aaaaaaaayou

canvas可以得到每个像素的rgb分量，是不是把蓝色和红色的值换一下就可以了？

2019-04-28



1



猫总

原本实现控制RGB范围来手动抠图，不过在使用的時候发现并不直观，调整起来很随缘，回看了一遍课程才发现重点是HSL调色，改进之后还是能比较精准（主要是直观）的把鹦鹉给单独替换颜色了

2019-07-12



无双

请问老师，我后台用的是Tomcat服务器，前端用ajax请求静态资源时会间隔会报412，也就是一次成功进入后台，一次报412，这该怎么解决呢？

2019-04-29



Geek_0bb537

winter老师给我讲一下那个presentational attributes 看不懂

2019-04-28



Izayoizuki

HSL感觉还是绘画游戏原画之类用得更多，编程领域反而挺少，无论h5游戏还是客户端游戏理解一般都是rgb/rgba

2019-04-28



Mupernb

```
for(var i=0;i<imgData.data.length;i++){
    [imgData.data[4*i+0],imgData.data[4*i+1]]=imgData.data[4*i+1],imgData.data[4*i+0]]
}
```

2019-04-28

