40 | CSS渲染：CSS是如何绘制颜色的？ | 极客时间

开篇词 | 从今天起，重新理解前端
01 | 明确你的前端学习路线与方法
02 | 列一份前端知识架构图
03 | HTML语义：div和span不是够用了吗？
04 | HTML语义：如何运用语义类标签来呈现Wiki网页？
05 | JavaScript类型：关于类型，有哪些你不知道的细节？
06 | JavaScript对象：面向对象还是基于对象？
07 | JavaScript对象：我们真的需要模拟类吗？
08 | JavaScript对象：你知道全部的对象分类吗？
新年彩蛋 | 2019，有哪些前端技术值得关注？
09 | CSS语法：除了属性和选择器，你还需要知道这些带@的规则
10 | 浏览器：一个浏览器是如何工作的？（阶段一）
11 | 浏览器：一个浏览器是如何工作的？（阶段二）
12 | 浏览器：一个浏览器是如何工作的（阶段三）
13 | 浏览器：一个浏览器是如何工作的？（阶段四）
14 | 浏览器：一个浏览器是如何工作的？（阶段五）
15 | HTML元信息类标签：你知道head里一共能写哪几种标签吗？
16 | JavaScript执行（一）：Promise里的代码为什么比setTimeout先执行？
17 | JavaScript执行（二）：闭包和执行上下文到底是怎么回事？
18 | JavaScript执行（三）：你知道现在有多少种函数吗？
19 | JavaScript执行（四）：try里面放return，finally还会执行吗？
20 | CSS 选择器：如何选中svg里的a元素？
21 | CSS选择器：伪元素是怎么回事儿？
22 | 浏览器DOM：你知道HTML的节点有哪几种吗？
23 | HTML链接：除了a标签，还有哪些标签叫链接？
24 | CSS排版：从毕升开始，我们就开始用正常流了
25 | 浏览器CSSOM：如何获取一个元素的准确位置

26 | JavaScript词法：为什么12.toString会报错?
27 |（小实验）理解编译原理：一个四则运算的解释器
28 | JavaScript语法（预备篇）：到底要不要写分号呢?
用户故事 | 那些你与“重学前端”的不解之缘
29 | JavaScript语法（一）：在script标签写export为什么会抛错?
期中答疑 | name(){}与name: function() {}，两种写法有什么区别吗?
30 | JavaScript语法（二）：你知道哪些JavaScript语句?
31 | JavaScript语法（三）：什么是表达式语句?
32 | JavaScript语法（四）：新加入的**运算符，哪里有些不一样呢?
33 | HTML替换型元素：为什么link一个CSS要用href，而引入js要用src呢?
34 | HTML小实验：用代码分析HTML标准
35 | CSS Flex排版：为什么垂直居中这么难?
36 | 浏览器事件：为什么会有捕获过程和冒泡过程?
37 | 浏览器API（小实验）：动手整理全部API
38 | CSS动画与交互：为什么动画要用贝塞尔曲线这么奇怪的东西?
答疑加餐 | 学了这么多前端的“小众”知识，到底对我有什么帮助?
39 | HTML语言：DTD到底是什么?
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。

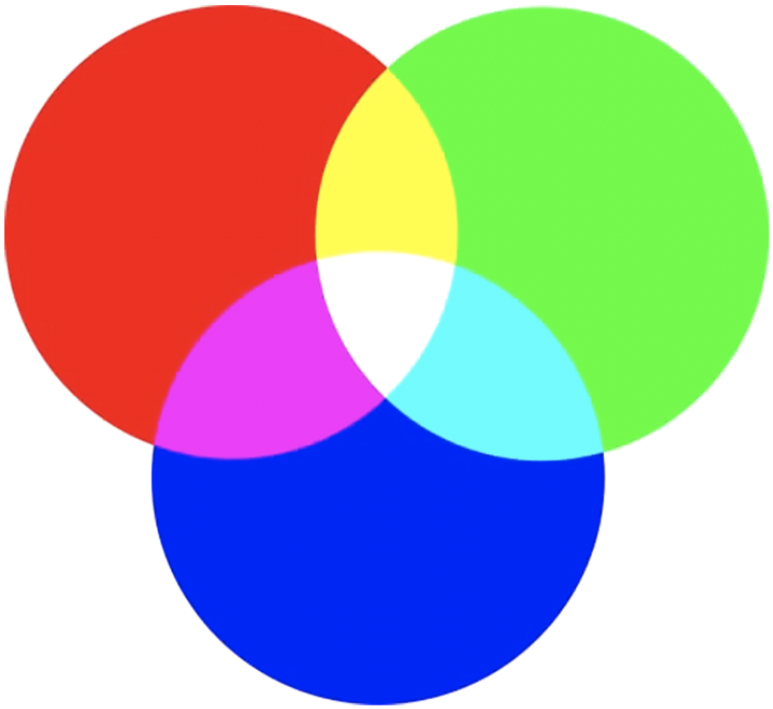
颜色的原理

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。

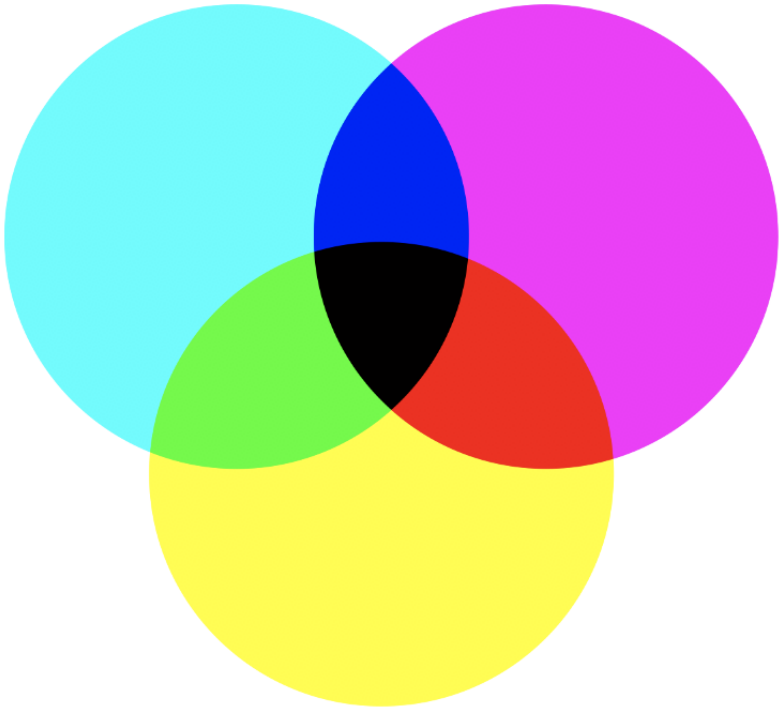
现代计算机中多用 0 - 255 的数字表示每一种颜色，这正好占据了一个字节，每一个颜色就占据三个字节。

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。

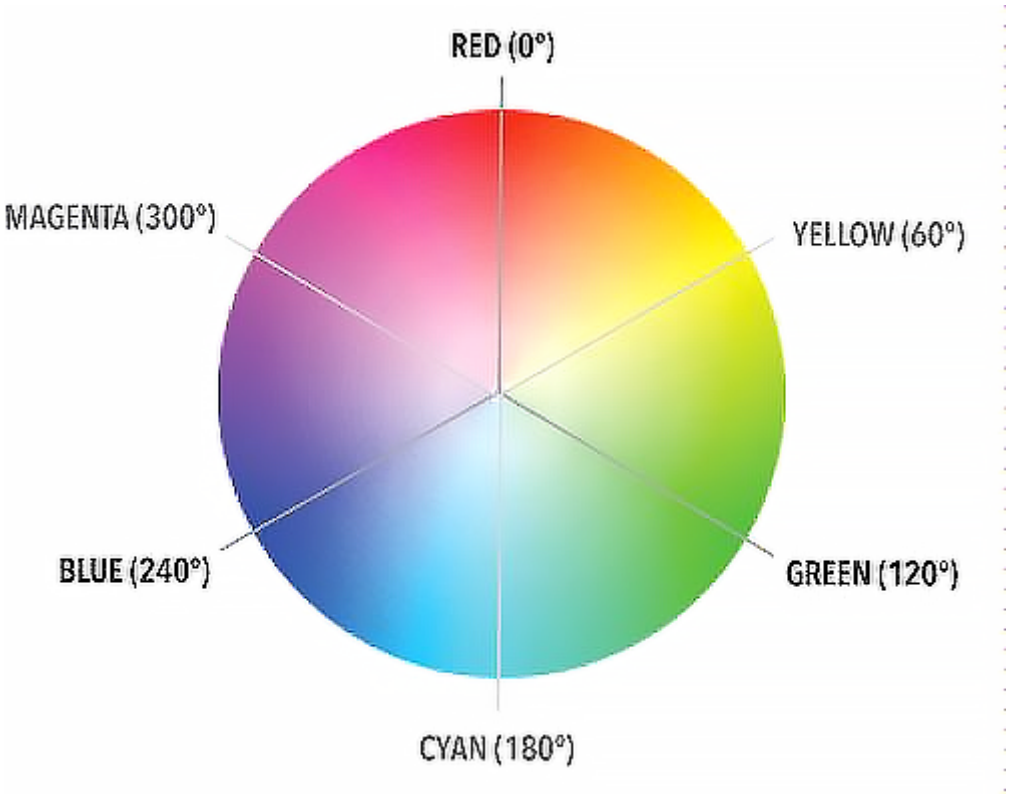
对 CMYK 颜色表示法来说，同一种颜色会有多种表示方案，但是我们参考印刷行业的习惯，会尽量优先使用黑色。

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：

```
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：

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

复制代码

放射性渐变需要一个中心点和若干个颜色：

```
radial-gradient(shape size at position, start-color, ..., last-color);
```
□复制代码

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：

```
<style>
.button {
    display: inline-block;
    outline: none;
    cursor: pointer;
    text-align: center;
    text-decoration: none;
    font: 14px/100% Arial, Helvetica, sans-serif;
    padding: .5em 2em .55em;
    text-shadow: 0 1px 1px rgba(0,0,0,.3);
    border-radius: .5em;
    box-shadow: 0 1px 2px rgba(0,0,0,.2);
    color: white;
    border: solid 1px ;
}
```

```
</style>

<div class="button orange">123</div>
```

□复制代码

```
var btn = document.querySelector(".button");

var h = 25;

setInterval(function(){

  h ++;

  h = h % 360;

  btn.style.borderColor=`hsl(${h}, 95%, 45%)`

  btn.style.background=`linear-gradient(to bottom,  hsl(${h},95%,54.1%),  hsl(${h},95%,84.1%))`

},100);
```

□复制代码

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 js 和 canvas，把这只雄性折衷鹦鹉变成跟雌性一样可爱的红色吗？

bd2star

精选留言(7)

-

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>
  <img id="img" class="bird" src="./bird.jpg">
  <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>
```

6