

CSS 动画

从入门到工作：CSS 全解

版权声明

本内容版权属杭州饥人谷教育科技有限公司（简称饥人谷）所有。

任何媒体、网站或个人未经本网协议授权不得转载、链接、转贴，或以其他方式复制、发布和发表。

已获得饥人谷授权的媒体、网站或个人在使用时须注明「资料来源：饥人谷」。

对于违反者，饥人谷将依法追究 responsibility。

联系方式

如果你想要购买本课程

请微信联系 [xiedaimala02](#) 或 [xiedaimala03](#)

如果你发现有人盗用本课程

请微信联系 [xiedaimala02](#) 或 [xiedaimala03](#)

动画的原理

人脑的 bug

动画

• 定义

- ✓ 由许多静止的画面（帧）
- ✓ 以一定的速度（如每秒30张）连续播放时
- ✓ 肉眼因视觉残象产生错觉
- ✓ 而误以为是活动的画面

• 概念

- ✓ 帧：每个静止的画面都叫做帧
- ✓ 播放速度：每秒 24 帧（影视）或者每秒 30 帧（游戏）

一个最简单的例子

- 将 div 从左往右移动

- ✓ [jsbin](#)

- 原理

- ✓ 每过一段时间（用 setInterval 做到）
- ✓ 将 div 移动一小段距离
- ✓ 直到移动到目标地点

- 注意性能

- ✓ 绿色表示重新绘制（repaint）了
- ✓ CSS 渲染过程依次包含布局、绘制、合成
- ✓ 其中布局和绘制有可能被省略

前端高手不用 left 做动画

- 用 transform (变形)

- ✓ [jsbin](#)

- 原理

- ✓ transform: translateX(0 => 300px)
- ✓ 直接修改会被合成，需要等一会修改
- ✓ transition 过渡属性可以自动脑补中间帧

- 注意性能

- ✓ 并没有 repaint (重新绘制)
- ✓ 比改 left 性能好

浏览器渲染原理

既然讲到这里，我们就深入一下吧

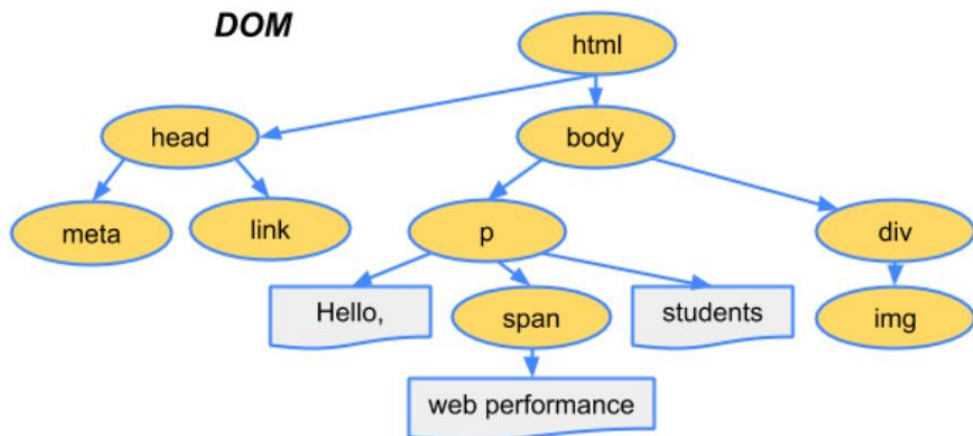
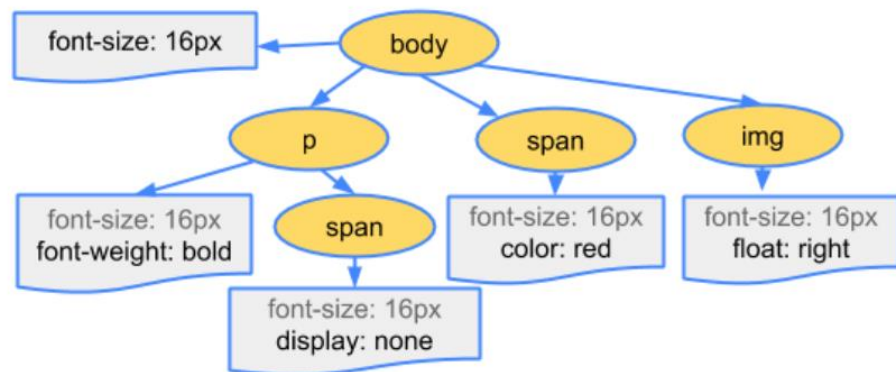
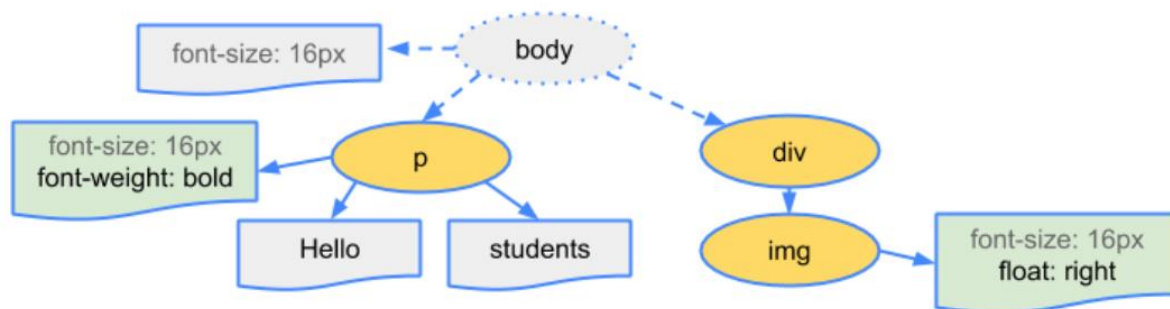
参考文章

- Google 团队写的文章（右上角中文）
 - ✓ [渲染树构建、布局及绘制](#)
 - ✓ [渲染性能](#)
 - ✓ [使用 transform 来实现动画](#)
- 查看 CSS 各属性触发什么
 - ✓ [CSSTriggers.com](#)

浏览器渲染过程

- 步骤

- ✓ 根据 HTML 构建 HTML 树 (DOM)
- ✓ 根据 CSS 构建 CSS 树 (CSSOM)
- ✓ 将两棵树合并成一颗渲染树 (render tree)
- ✓ Layout 布局 (文档流、盒模型、计算大小和位置)
- ✓ Paint 绘制 (把边框颜色、文字颜色、阴影等画出来)
- ✓ Compose 合成 (根据层叠关系展示画面)

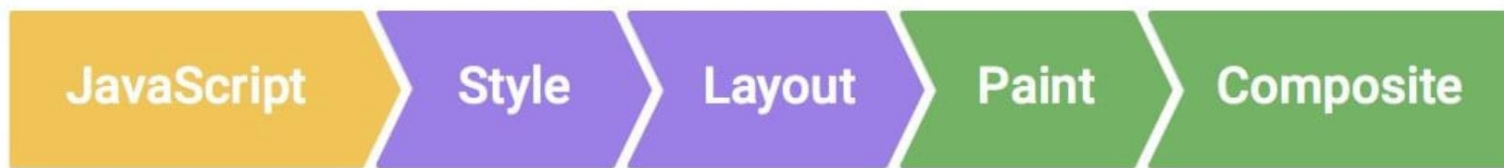
DOM**CSSOM****Render Tree**

如何更新样式

- 一般我们用 JS 来更新样式
 - ✓ 比如 `div.style.background = 'red'`
 - ✓ 比如 `div.style.display = 'none'`
 - ✓ 比如 `div.classList.add('red')`
 - ✓ 比如 `div.remove()` 直接删掉节点
- 那么这些方法有什么不同吗
 - ✓ 有三种不同的渲染方式
 - ✓ 详细看下一张 PPT

三种更新方式

1. JS / CSS > 样式 > 布局 > 绘制 > 合成



2. JS / CSS > 样式 > 绘制 > 合成



3. JS / CSS > 样式 > 合成



三种更新方式

- 第一种，全走

- ✓ div.remove() 会触发当前消失，其他元素 relayout

- 第二种，跳过 layout

- ✓ 改变背景颜色，直接 repaint + composite

- 第三种，跳过 layout 和 paint

- ✓ 改变 transform，只需 composite
- ✓ 注意必须全屏查看效果，在 iframe 里看有问题

我 TM 怎么知道 每个属性触发什么流程

CSS 变态之处来了：自己一个一个试吧

还好，程序员喜欢分享

<https://csstriggers.com/>

这个网站已经把所有属性全试过了

CSS 动画优化

考的概率5%

- 没什么技术含量

- ✓ 答案都在 [Google 写的文章](#)里，谁看完谁牛 X

- JS 优化

- ✓ 使用 requestAnimationFrame 代替 setTimeout 或 setInterval

- CSS 优化

- ✓ 使用 will-change 或 translate

- 没错

- ✓ 完全就是死记硬背

transform

完整介绍

你不知道去看 MDN 吗

- 他们充钱了
- 哦，充钱了啊，那我还是讲解一下吧

transform

- 四个常用功能

其它不用看，看了面试也不会问

- ✓ 位移 translate
- ✓ 缩放 scale
- ✓ 旋转 rotate
- ✓ 倾斜 skew

- 经验

- ✓ 一般都需要配合 transition 过渡
- ✓ inline 元素不支持 transform，需要先变成 block

transform 之 translate

• 常用写法

`translateX(50%);` 向右走自己长度的50%

- ✓ `translateX(<length-percentage>)`
- ✓ `translateY(<length-percentage>)`
- ✓ `translate(<length-percentage> , <length-percentage>?)`
- ✓ `translateZ(<length>)` 且父容器 perspective
- ✓ `translate3d(x, y, z)`
- ✓ JSBin 演示

• 经验

- ✓ 要学会看懂 MDN 的语法示例
- ✓ `translate(-50%,-50%)` 可做绝对定位元素的居中

```
<div class="father">  
  <div class="son"></div>  
</div>
```

```
.father{  
  border:1px solid black;  
  height:500px;  
  position:relative;  
}  
.son{  
  border:1px solid red;  
  height:200px;  
  width:100px;  
  position:absolute;  
  left:50%;  
  top:50%;  
  transform:translate(-50%,-50%);  
}
```

transform 之 scale

- 常用写法

- ✓ `scaleX(<number>)`
- ✓ `scaleY(<number>)`
- ✓ `scale(<number> , <number>?)`
- ✓ [JSBin 演示](#)

- 经验

- ✓ 用得较少，因为容易出现模糊

transform 之 rotate

• 常用写法

- ✓ rotate([<angle> | <zero>]) transform:rotate(360deg);
- ✓ rotateZ([<angle> | <zero>])
- ✓ rotateX([<angle> | <zero>])
- ✓ rotateY([<angle> | <zero>])
- ✓ rotate3d 太复杂，无法用语言表述
- ✓ JSBin 演示

• 经验 比较少用到

- ✓ 一般用于 360 度旋转制作 loading
- ✓ 用到时再搜索 rotate MDN 看文档

transform 之 skew

- 常用写法

- ✓ skewX([<angle> | <zero>])
- ✓ skewY([<angle> | <zero>])
- ✓ skew([<angle> | <zero>], [<angle> | <zero>]?)
- ✓ [JSBin 演示](#)

- 经验

- ✓ 用得较少
- ✓ 用到时再搜 skew MDN 文档

transform 多重效果

- 组合使用

- ✓ transform: scale(0.5) translate(-100%, -100%);
- ✓ transform: none; 取消所有

实践

- 跳动的心，先给大家

- ✓ JSBin

<http://js.jirengu.com/lajod/1/edit?html,css,output>

- 心得

- ✓ CSS 需要你有想象力，而不是逻辑
- ✓ CSS 给出的属性都很简单，但是可以组合得很复杂

transition

过渡

你不知道去看 MDN 吗

- 他们充钱了
- 哦，充钱了啊，那我还是讲解一下吧

transition 过渡

- 作用

- ✓ 补充中间帧

- 语法

过多久再动（很少用）

- ✓ transition: 属性名 时长 过渡方式 延迟
- ✓ transition: left 200ms linear `transition:width 1s ease 3s;`
- ✓ 可以用逗号分隔两个不同属性
- ✓ transition: left 200ms, top 400ms
- ✓ 可以用 all 代表所有属性
- ✓ transition: all 200ms
- ✓ 过渡方式有: linear | ease | ease-in | ease-out | ease-in-out | cubic-bezier | step-start | step-end | steps, 具体含义要靠数学知识

注意

- 并不是所有属性都能过渡

- ✓ display: none => block 没法过渡
- ✓ 一般改成 visibility: hidden => visible (虽然不见，但还是占位置 (不要问为什么))
- ✓ display 和 visibility 的区别自己搜一下
- ✓ background 颜色可以过渡吗? 可以
- ✓ opacity 透明度可以过渡吗? 可以

过渡必须要有起始

一般只有一次动画，或者两次

比如 hover 和非 hover 状态的过渡

如果除了其实，还有中间点

怎么办

两种办法

- 使用两次 transform

- ✓ `.a === transform ===> .b`
- ✓ `.b === transform ===> .c`
- ✓ 如何知道到了中间点呢?
- ✓ 用 `setTimeout` 或者监听 `transitionend` 事件
- ✓ JSBin 示例

- 使用 animation

- ✓ 声明关键帧
- ✓ 添加动画
- ✓ JSBin 示例

提问

- 如何让动画停在最后一帧
 - ✓ 搜索 css animation stop at end
 - ✓ 网友给出的答案是加个 forwards
 - ✓ JSBin 演示

@keyframes 完整语法

- 标准写法

- ✓ 搜索 [keyframes MDN](#) 讲得很清楚
- ✓ 一种写法是 from to
- ✓ 另一种写法是百分数

```
@keyframes slidein {  
  from {  
    transform: translateX(0%);  
  }  
  
  to {  
    transform: translateX(100%);  
  }  
}
```

```
1 | @keyframes identifier {  
2 |   0% { top: 0; left: 0; }  
3 |   30% { top: 50px; }  
4 |   68%, 72% { left: 50px; }  
5 |   100% { top: 100px; left: 100%; }  
6 | }
```

animation

• 缩写语法

animation: 时长 | 过渡方式 | 延迟 | 次数 | 方向 | 填充模式
| 是否暂停 | 动画名;

- ✓ 时长: 1s 或者 1000ms
- ✓ 过渡方式: 跟 transition 取值一样, 如 linear
- ✓ 次数: 3 或者 2.4 或者 infinite
- ✓ 方向: reverse | alternate | alternate-reverse
- ✓ 填充模式: none | forwards | backwards | both
- ✓ 是否暂停: paused | running
- ✓ 以上所有属性都有对应的单独属性

实践

- 把红心重新做一遍

- ✓ [JSBin 示例](http://js.jirengu.com/titec/1/edit?html,css,output)

<http://js.jirengu.com/titec/1/edit?html,css,output>

再见

作业：用 transition 和 animation 分别画跳动的心