# CSS

## 动画

* + 1. 对于复杂动画，一般将其调整为绝对定位，防止挤压其他元素产生回流
    2. CSS3硬件加速：
       1. transform、opacity、filters、Will-change。使用上述属性时，会通过GPU加速，不产生回流
       2. 注意：不要为太多元素开启加速，否则会有性能问题；硬件加速会使抗锯齿失效

# JavaScript

## 基本运算

* + 1. 转换浮点数至整型：用~~a代替parseInt（a）

## 变量

* + 1. 尽量使用let和const：使用块作用域可以更早地让垃圾回收程序介入，尽早回收内存
    2. 对象池：当一个会创建对象的函数被频繁调用时，会导致垃圾回收更频繁，可以使用对象池来减少垃圾回收次数
       1. 对象池需要使用某种结构来维护所有对象【一般用数组】，有获取对象、释放对象两个方法
       2. 获取对象的方法会从池中取出对象【可先重新赋值】并返回【若池为空，创建对象并返回】
       3. 释放对象的方法会将对象放入池中；变量释放完对象后需要置null来解除引用
    3. 静态分配【不常用】：数组到达指定长度时，引擎会删除当前数组，创建更大一级的数组。初始化时创建一个大小够用的数组可以减少回收

## DOM操作

* + 1. 多次获取布局信息：获取布局信息时会强制清空队列进行回流，应该将布局信息存储为变量
    2. 修改DOM：使元素脱离文档流后操作，再加回文档流，会引起两次回流（浏览器会用队列存储），默认Chrome：100ms，IE11：8040ms
       1. 方法一：隐藏元素，操作后，重新显示（display：none），Chrome：112ms，IE11：6630ms
       2. 方法二：使用文档片段（document fragment），构建子树，然后将它加入到文档中，Chrome：145ms，IE11：24050ms
       3. 方法三：拷贝原始元素（cloneNode），修改后替换原始元素（parentNode.replaceChild），Chrome：109ms，IE11：20210ms
    3. 多次修改样式：使用el.style.cssText += “xxx” 或 el.className += “xxx” 代替 多条style语句

## V8

* + 1. 隐藏类和删除操作：
       1. 创建的实例会与隐藏类关联起来，共享隐藏类可以提高性能。当实例动态创建或删除属性时，会生成新的隐藏类。
       2. 尽量通过在构造函数中声明属性、置null来删除属性