

浅谈DOM的操作以及性能优化问题-重绘重排

投稿: jingxian

下面小编就为大家带来一篇浅谈DOM的操作以及性能优化问题-重绘重排。小编觉得挺不错的，现在分享给大家，也给大家做个参考。一起跟随小编过来看看吧

写在前面:

大家都知道DOM的操作很昂贵。

然后贵在什么地方呢?

一、访问DOM元素

二、修改DOM引起的重绘重排

一、访问DOM

像书上的比喻: 把DOM和JavaScript (这里指ECMScript) 各自想象为一个岛屿, 它们之间用收费桥梁连接, ECMAScript每次访问DOM, 都要途径这座桥, 并交纳“过桥费”, 访问DOM的次数越多, 费用也就越高。因此, 推荐的做法是尽量减少过桥的次数, 努力待在ECMAScript岛上。我们不可能不用DOM的接口, 那么, 怎样才能提高程序的效率?

既然无法避免, 那就减少访问。(width、offsetTop、left... 能少就少, 可以缓存起来的, 就缓存)

```
// code1错误
console.time(1);
for(var i = 0; i < times; i++) {
    document.getElementById("div1").innerHTML += 'a';
}
console.timeEnd(1);

// code2正确
console.time(2);
var str = '';
for(var i = 0; i < times; i++) {
    str += 'a';
}
document.getElementById("div2").innerHTML = str;
console.timeEnd(2);
//////////
```

1: 1385.897ms

2: 4.650ms

html集合&遍历DOM

html集合类似数组, 但是跟数组还是不一样的。如: document.getElementsByTagName('a') 返回的html集合。这个集合是实时更新的, 而后面代码修改了DOM, 会反映在这个html集合里面。可尝试代码。

```
<body>
<ul id= 'fruit'>
<li> apple </li>
<li> orange </li>
<li> banana </li>
</ul>
</body>
<script type="text/javascript">
var lis = document.getElementsByTagName('li');
var peach = document.createElement('li');
peach.innerHTML = 'peach';
document.getElementById('fruit').appendChild(peach);

console.log(lis.length); // 4
</script>
```

正因为这个原因: html集合, 读取 length 属性比数组消耗大多了。

要解决这个问题并不难, 在遍历DOM集合的时候, 缓存length就好了。不要每次使用就获取, 主要体现在for循环中(你应该知道, for循环中, 每一次都会执行判读语句, 读取length)

```
console.time(0);
var lis0 = document.getElementsByTagName('li');
var str0 = '';
for(var i = 0; i < lis0.length; i++) {
    str0 += lis0[i].innerHTML;
}
console.timeEnd(0);

console.time(1);
var lis1 = document.getElementsByTagName('li');
var str1 = '';
for(var i = 0, len = lis1.length; i < len; i++) {
    str1 += lis1[i].innerHTML;
}
console.timeEnd(1);
```

二、重绘重排

1.什么是重绘重排?

浏览器下载完页面中的所有组件——HTML标记、JavaScript、CSS、图片之后会解析生成两个内部数据结构——DOM树和渲染树。

在文档初次加载时, 浏览器引擎通过解析html文档构建一棵DOM树, 之后根据DOM元素的几何属性构建一棵用于展示渲染的渲染树。渲染树中的节点被称为“帧”或“盒”符合CSS模型的定义, 可理解为(包括理解页面元素为一个具有大小, 填充, 边距, 边框和位置的盒子)。由于隐藏元素不需要显示, 渲染树中并不包含DOM树中隐藏的元素(知道这点有用)。当渲染树构建完成, 浏览器把每一个元素放到正确的位置上, 然后再根据每一个元素的其他样式, 绘制页面。

由于浏览器的流布局, 对渲染树的计算通常只需要遍历一次就可以完成。但table及其内部元素除外, 它可能需要多次计算才能确定好其在渲染树中节点的属性, 通常要花3倍于同等元素的时间。这也是为什么我们要避免使用table做布局的一个原因。

重绘: 是一个元素外观的改变所触发的浏览器行为, 例如改变visibility、outline、背景色等属性(上面说到的其他属性)。浏览器会根据元素的新属性重新绘制, 使元素呈现新的外观。重绘不会带来重新布局, 并不一定伴随重排。

重排: 当DOM的变化影响了元素的几何属性(宽或高), 浏览器需要重新计算元素的几何属性, 同样其他元素的几何属性和位置也会因此受到影响。浏览器会使渲染树中受到影响的部分失效, 并重新构造渲染树。这个过程称为重排。重排一定伴随着重绘。

2. 触发重排的操作:

2.1 修改DOM元素几何属性:

修改元素大小, 位置, 内容 (一般只有重绘, 但是内容可能导致元素大小变化)

2.2 DOM树结构发生变化

当DOM树的结构变化时, 例如节点的增减、移动等, 也会触发重排。浏览器引擎布局的过程, 类似于树的前序遍历, 是一个从上到下从左到右的过程。通常在这个过程中, 当前元素不会再影响其前面已经遍历过的元素。所以, 如果在body最前面插入一个元素, 会导致整个文档的重新渲染, 而在其后插入一个元素, 则不会影响到前面的元素。

2.4 改变浏览器大小

3.渲染树变化的排队和刷新

思考下面代码:

```
var ele = document.getElementById("myDiv");
ele.style.borderLeft = '1px';
ele.style.borderRight = '1px';
ele.style.borderTop = '2px';
// var_top = ele.offsetTop; //刷新队列
ele.style.padding = '5px';
```

三行代码, 三次修改元素的几何属性, 浏览器应该发生三次重排重绘。

但是浏览器并不会这么笨, 它也是有做优化的。它会把三次修改“保存”起来(大多数浏览器通过队列化修改并批量执行来优化重排过程, 也有设置时间片段的), 一次完成!

然而, 如果你在前三行代码中, 以下获取DOM布局信息。(为了返回最新的布局信息, 将立即执行渲染树变化队列的更新)

如上面被注释的第4行, 如果取消注释会导致 (2+3)、(5) 两次重排;

获取关于DOM布局信息的属性:

```
offsetTop, offsetLeft, offsetWidth, offsetHeight
scrollTop, scrollLeft, scrollWidth, scrollHeight
clientTop, clientLeft, clientWidth, clientHeight
getComputedStyle() (currentStyle in IE)
```

4 应对方法: 尽量减少重绘次数、减少重排次数、缩小重排的影响范围。

4.1 合并多次操作, 如上面的操作

ele.style.cssText = 'border-left: 1px; border-right: 2px; padding: 5px';

4.2 将需要多次重排的元素, position属性设为absolute或fixed, 这样此元素就脱离了文档流, 它的变化不会影响到其他元素。例如有动画效果的元素最好设置为绝对定位。

4.3 由于display属性为none的元素不在渲染树中, 对隐藏的元素操作不会引发其他元素的重排。如果要对一个元素进行复杂的操作时, 可以先隐藏它, 操作完成后再显示。这样只在隐藏和显示时触发2次重排。但是这可能导致浏览器的闪烁。

4.4 在内存中多次操作节点, 完成后再渲染到文档中去(可使用fragment元素)。例如要异步获取表格数据, 渲染到页面。可以先取得数据后在内存中构建整个表格的html片段, 再一次性渲染到文档中去, 而不是循环添加每一行。

```
var fragment = document.createDocumentFragment(); // 未使用的虚拟节点, appendChild(fragment) //append()的是里面

var li = document.createElement('li');
li.innerHTML = 'apple';
fragment.appendChild(li);

var li = document.createElement('li');
li.innerHTML = 'watermelon';
fragment.appendChild(li);

document.getElementById('fruit').appendChild(fragment);
```

以上就是小编为大家带来的浅谈DOM的操作以及性能优化问题-重绘重排全部内容了, 希望大家多多支持脚本之家~

您可能感兴趣的文章:

高性能JavaScript 重排与重绘(2)

少儿英语

文昌楼盘

马来西亚房价

男子没有精

哈伦少儿英语

助听器

学习软件编程

js购跑车

抑郁有哪些症

脚小怎么办

相关文章

使用百度地图api实现根据地址查询经纬度

2014-12-12

微信小程序动态显示项目倒计时效果

2017-06-06

js 鼠标事件的抓取

2010-04-04

js 获取范围内的随机数实例代码

2016-08-08

JS打开图片另存为对话框实现代码

2012-12-12

通过JavaScript进行UTF-8编码的实现方法

2016-06-06

微信端开发--登录小程序步骤

2017-01-01

JavaScript中的console.time()函数详细介绍

2014-12-12

javascript间隔定时器(定时定时器)学习 间隔调用和延时调用

2014-01-01

javascript显示上周、上个月日期的处理方法

2016-02-02

女人如何受孕

男子没有精

制作模型

全脸面的填充

男友下蛋痛

面部脂肪填充

勃而不坚

上节育环

男人健康问题

房产证抵押贷款

网友评论

0 条评论

登录

我来说两句...

搜索

硬件

安全

手游

网络

攻略

评测

应用

苹果

Win10

首页

编程

手机

电脑

教程

电脑版 - 返回首页

Copyright ©2017 脚本之家. All Rights Reserved

世界上最牛的公路, 断在半空中, 司机开上去像要起飞

北青网 刚刚

七岁小女孩竟然离奇怀孕, 真相揭秘让人后怕!

北青网 刚刚

自学编程怎么入门

如何学习编程

广告

一见钟情嫁大13岁老公, 夫妻恩爱了13年, 在《甄嬛传》事业再翻红

北青网 刚刚

朋友妻不可欺! 盘点娱乐圈挖好兄弟墙角的四大男星

李洪伙 刚刚

龙发装饰怎么样

龙发装饰公司

广告

这里的年轻女孩不甘心嫁外国人? 但全都梦想着嫁给中国小伙?

北青网 1分钟前

21岁女子遭遇车祸, 医生已宣布死亡, 母亲用一块冰块竟出现奇迹

北青网 1分钟前

海外留学生招聘

留学生招聘会

广告

奇怪村庄, 700年内不外娶不外嫁, 整个村子都姓吴

北青网 1分钟前

17岁嫁给57岁的爷爷, 3年居然成功怀孕? 林靖恩: 李坤城很小心

北青网 1分钟前

牙膏去黑头

牙膏去黑头真的没吗

广告

印度小伙竟拥有39个老婆和136个孩子, 每天最大开销就是吃饭

北青网 2分钟前

铁证如山! 李小璐出轨已坐实, 这一细节出卖了她

北青网 2分钟前

达内培训靠谱吗

参加达内培训就业好吗

广告