重绘和回流以及如何优化

参考

1.概念

2.引起重绘和回流的事件

3.对优化的思考

1.概念：

```
js 代码：

var s = document.body.style;
s.padding = "2px"; // 回流+重绘
s.border = "1px solid red"; // 再一次 回流+重绘
s.color = "blue"; // 再一次重绘
s.backgroundColor = "#ccc"; // 再一次 重绘
s.fontSize = "14px"; // 再一次 回流+重绘
// 添加node，再一次 回流+重绘
document.body.appendChild(document.createTextNode('abc!'));
```

说到这里大家都知道回流比重绘的代价要更高，回流的花销跟render tree有多少节点需要重新构建有关系，假设你直接操作body，比如在body最前面插入1个元素，会导致整个render tree回流，这样代价当然会比较高，但如果是指body后面插入1个元素，则不会影响前面元素的回流。

2.引起重绘和回流的事件

2.1 引起重绘：

颜色、背景色、文字颜色改变

2.2 引起回流：

- 边框，宽、高、填充改变
- 添加或者删除可见的DOM元素
- padding
- DOM操作
- 位置改变
- 调整窗口大小
- 页面渲染初始化（DOM载入后的第一次回流，将会遍历所有frame。）每个页面至少回流一次，即页面首次加载

> *回流必将引起重绘，而重绘不一定会引起回流*

3.避免回流和重绘方法

- 减少逐项更改样式，最好一次性更改style，或者将样式定义为class并一次性更新
- 避免循环操作dom，创建一个documentFragment或div，在它上面应用所有DOM操作，最后再把它添加到window.document
- 避免多次读取offset等属性。无法避免则将它们缓存到变量
- 将复杂的元素绝对定位或固定定位，使得它脱离文档流，否则回流代价会很高

第二种方法-例子（取自行前墙）

```
function addCitysToSelect(selector, provinceSelector, needCity) {
    var selectedProvince = provinceSelector.options[provinceSelector.selectedIndex];
    var citys = gaodeMap.getCitys(selectedProvince.value);
    selector[0].options.length = 1;
    if (citys instanceof Array){
        if (!needCity){
            selector[0].selectedIndex = 0;
        }
        else{
            selector[0].selectedIndex = 1;
        }
        var fragment = document.createDocumentFragment();
        citys.forEach(function (city) {
            var option = document.createElement('option');
            option.value = city.citycode;
            option.innerText = city.name;
            fragment.appendChild(option);
        });
        selector.append(fragment);
    }
    else{
        selector[0].selectedIndex = 0;
        if (!needCity){
            messageControler.cityChange('');
        }
    }
}
```