1.减少DOM操作，DOM操作消耗大量性能,如在循环内保存html字符串，在循环外一次写入

2.innerHTML比DOM（如：createElement()）方法要快一些，而且操作简单

3.节点克隆

使用element.cloneNode()克隆已有的元素比创建新的元素更快

4.html集合（如getELementByTagName()获取的集合）

1）html集合始终与文档保持连接，内次访问都需要重复查询过程，因此效率低

解决方法：把html集合拷贝到普通数组，遍历数组比遍历html集合快得多

2）减少html集合的访问

如循环中用局部变量保存length值

5.使用children（html标签）代替childNode（nodeList类型），因为children不包含空格

6.选择器优化

使用querySelector比原生的如getELementById,getELementByTagName,getELementByTag的组合要快

6.重绘和重排

重绘和重排都会产生高昂的代价，应该避免多次产生重绘重排

重排的发生

1）页面初始化，第一次渲染

2）页面元素的大小，位置，几何形状的变化

3）html元素的某些属性的读写

如:

offsetTop,offsetHeight

clientTop,clientWidth

scrollTop,scrollWidth

4）内容的改变

5）浏览器窗口变化

6）dom元素的添加和删除

优化：

1.通过修改元素的class偏高修改元素的css属性

2.批量修改DOM：

1）使元素脱离文档流

如absolute，fixed定位，隐藏元素，使用documentFragment

=》应用改变

=》回到文档流中

2）使用documentFragment批量添加html元素

3.缓存布局信息，保存到局部变量，避免多次访问，如：

offsetTop,offsetHeight

clientTop,clientWidth

scrollTop,scrollWidth

4.在动画循环中，使用current而避免使用offsetLeft，应像这样：

var current = ele.offsetLeft;

for(...) {

if(current<200) {//无需使用offsetLeft

current++;

ele.style.left = current+'px';

}

}

5.脱离动画流

1）使用绝对定位脱离文档流

2）执行动画

3）恢复文档流

6.使用事件委托，冒泡原理，把多个类似功能的元素的事件注册到同一个祖先元素上，减少时间注册次数，减少内存消耗

7.使用css3动画，使用显存处理，效率更高。