1. DOM操作成本高在哪
2. 什么是DOM

* Document Object Model 文档对象模型
* DOM是为HTML(and XML)提供的API。只提供编程接口，无法实际操作HTML中的内容。在浏览器端，前端们可以通过脚本语言(javascript)通过DOM操作HTML内容。
* Python也可以访问DOM。DOM不是提供给js的API，也不是js中的API。
* CSSOM：CSS Object Model 浏览器将CSS代码解析成树形的数据结构，与DOM是2个独立的数据结构。

1. 浏览器渲染过程
   1. 解析HTMl，构建DOM树
   2. 解析CSS，生成CSS规则树
   3. 合并DOM树和CSS规则树，生成render树
   4. 布局render树(Layout/reflow)，负责各元素尺寸、位置的计算
   5. 绘制render树(paint)，绘制页面像素信息
   6. 浏览器将各层的信息发送给GPU，GPU将各层合成(composite)，显示在屏幕上

* 前3点可能要多次执行，比如js脚本操作DOM、更改CSS样式时，浏览器要重新构建DOM、CSSOM树，要重新render，重新layout、paint

1. 何时触发reflow和repaint

reflow回流：元素的内容、结构、位置或尺寸发生了变化

repaint重绘：元素发生的改变只影响了节点的一些样式，如背景色、边框颜色，文字颜色等

* 回流的成本开销高于重绘，一个节点的回流往往导致子节点以及同级节点的回流
* 回流一定重绘，重绘不一定回流

引起回流：

页面第一次渲染(初始化)

DOM树变化(如增删节点)

Render树变化

浏览器窗口resize

获取元素的某些属性

引起重绘：

背景色、颜色、字体改变，字体大小发生变化时，会触发回流

减少reflow repaint次数

避免逐个修改节点样式，尽量一次性修改

使用DocumentFragment将需要多次修改的DOM元素缓存，最后一次性append到真实的DOM中

将需要多次修改的DOM元素设置display:none，操作完再显示

避免多次读取某些属性

将复杂的节点元素脱离文档流，降低回流成本

1. 应该将css放在头部，将js放在尾部

构建Render树需要DOM和CSSOM，所以HTML和CSS都会阻塞渲染，所以要让CSS尽早加载，以缩短首次渲染时间

js引擎线程和渲染线程是互斥的

1. 首屏优化的方法

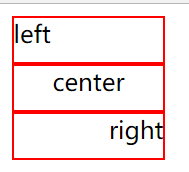
减少资源请求数量

使CSS样式表尽早加载，减少@important的引用，应该用link

异步js (async)

1. 文本方向
2. text-align 文本水平对齐方式

text-align:left | center | right | inherit



1. dir 文本方向

dir:ltr | rtl | auto

有符号才起作用

1. 文本溢出

单行文本溢出：

div{width: 200px;overflow: hidden;text-overflow: ellipsis;white-space: nowrap;}

多行文本溢出

div{width: 200px;display: -webkit-box;-webkit-box-orient: vertical;-webkit-line-clamp:3;overflow: hidden;}