1.解析html文件，创建DOM树

自上而下解析，遇到任何样式（link、style）和脚本（script）都会阻塞

1）css加载不会阻塞html文件的解析，但会阻塞dom的渲染

2）css加载会阻塞后面js语句的执行

3）js会阻塞html的解析和渲染

4）没有defer和async标签的script会立即加载并执行

5）有async标签的js，js的加载执行和html的解析和渲染并行

6）有defer标签的js，js的加载和html的解析和渲染并行，但会在html解析完成后执行,在触发DOMContentLoaded事件前执行

7）DOMContentLoaded和onload的区别：DOMContentLoaded在html解析完毕后执行，loload在页面完全加载完成后执行（包括样式和图片）

监听资源加载完成有四种方式

1. window.onload = function(){....}
2. window.addEventListener("load",function(){....});
3. document.body.onload = function(){....}
4. <body onload = "load()">

==========

meta相关总结

HTML5页面窗口自动调整到设备宽度，并禁止用户缩放页面

```
<meta name="viewport" content="width=device-width,initial-scale=1.0,minimum-scale=1.0,maximum-scale=1.0,user-scalable=no" />
```

忽略将页面中的数字识别为电话号码

```
<meta name="format-detection" content="telephone=no" />
```

忽略Android平台中对邮箱地址的识别

```
<meta name="format-detection" content="email=no" />
```

当网站添加到主屏幕快速启动方式，可隐藏地址栏，仅针对ios的safari

```
<meta name="apple-mobile-web-app-capable" content="yes" />
<!-- ios7.0版本以后，safari上已看不到效果 -->
```

将网站添加到主屏幕快速启动方式，仅针对ios的safari顶端状态条的样式

<meta name="apple-mobile-web-app-status-bar-style" content="black" />

==========

一：HTML页面渲染及优化详解

页面渲染主要经过过程，具体介绍如下

1:HTML/SVG/XHTML，解析这三种文件会产生一个 DOM Tree

2:CSS，解析 CSS 会产生 CSS 规则树

3:合并DOM Tree 和CSS规则树（CSSOM）生成渲染树（Render树）

4:浏览器根据Render树计算DOM节点位置，搭建页面结构

5:根据每个节点的属性（css）绘制HTML页面

文档对象模型（DOM树）的解析过程

字节 → 字符 → 令牌 → 节点 → 对象模型

1:**字节转换：**浏览器从磁盘或网络读取 HTML 的原始字节，并根据文件的指定编码（例如 UTF-8）将它们转换成各个字符。

2:**令牌化：**浏览器将字符串转换成 W3C HTML5 标准规定的各种令牌，例如，“<html>”、“<body>”，以及其他尖括号内的字符串。每个令牌都具有特殊含义和一组规则（HTML语义化）。

3:**词法分析：** 发出的令牌转换成定义其属性和规则的“对象”。

4:**DOM 树：**根据父子关联关系，构建出对应节点数（DOM树）

二、CSS 对象模型 (CSSOM)

CSS树的生成和DOM树的生成基本是相同，如下

因为CSS关系也会有父子关系，就是css中常说的继承，一些样式如font-size等子元素会继承父级，所以会生成一套对应的CSS树和DOM树相对应。

三、Reflow（重排）and Repaint（重绘）

通过上面的页面渲染介绍，很容易理解，重排就是重新布局页面结构，计算节点位置，而重绘就是绘制页面，只是一些样式如背景，颜色的变化等，不需要重新计算位置布局，所以**重排一定会导致重绘，但是重绘不一定导致重排**

导致页面重排主要有以下几点
1:页面首次渲染
2:浏览器窗口大小发生改变
3:元素尺寸（宽高等）或位置（布局属性如padding、margin）发生改变。
4:元素内容变化（文字数量或图片大小等等）。
5:元素字体大小变化。
6:添加或者删除可见的DOM元素。
7:激活CSS伪类（例如：:hover）。
8:设置style属性。
9:查询某些属性或调用某些方法。

四、重排优化
分离读写

```
div.style.left = '10px';
console.log(div.offsetLeft);
div.style.top = '10px';
console.log(div.offsetTop);
div.style.width = '20px';
console.log(div.offsetWidth);
div.style.height = '20px';
console.log(div.offsetHeight);
```

这种情况看会触发四次重排，offsetTop，scrollTop，clientTop等属性的修改会触发重排，当浏览器获取DOM样式的时候立刻会执行一次重排，因为需要计算浏览器位置坐标，不会再继续观察下文是否还有DOM的操作

```
div.style.left = '10px';
div.style.top = '10px';
div.style.width = '20px';
div.style.height = '20px';
console.log(div.offsetLeft);
console.log(div.offsetTop);
console.log(div.offsetWidth);
console.log(div.offsetHeight);
```

这种情况看似会触发四次重排，实际只会触发一次重排，现代浏览器基本都有渲染机制，浏览器会批量将样式修改一次性执行，批量修改完后再批量获取DOM位置，实际只触发一次.

缓存DOM信息

```
div.style.left = div.offsetLeft + 1 + 'px';
div.style.top = div.offsetTop + 1 + 'px';
```

等价于

```
let curLeft = div.offsetLeft;
div.style.left = curLeft + 1 + 'px';
let curTop = div.offsetTop;
div.style.top = curTop + 1 + 'px';
```

通过上文，容易理解这种情况会触发两次重排，可以使用缓存（实际是分离读写）来优化。

```
let curLeft = div.offsetLeft;
let  curTop = div.offsetTop;
div.style.left = curLeft + 1 + 'px';
div.style.top = curTop + 1 + 'px';
```

脱离文档

```
var ul = document.getElementById('demo');
for(var i = 0; i < 5; i++){
    var li = document.createElement('li');
    var text = document.createTextNode(i);
    li.appendChild(text);
    ul.appendChild(li);
}
```

这种情况会触发五次重排

```
var ul = document.getElementById('demo');
ul.style.display = 'none'; <--
for(var i = 0; i < 1e5; i++){
    var li = document.createElement('li');
    var text = document.createTextNode(i);
    li.appendChild(text);
    ul.appendChild(li);
```

```
}
ul.style.display = 'block'; <--

var ul = document.getElementById('demo');
var frg = document.createDocumentFragment(); <--
for(var i = 0; i < 1e5; i++){
    var li = document.createElement('li');
    var text = document.createTextNode(i);
    li.appendChild(text);
    frg.appendChild(li); <--
}
ul.appendChild(frg); <--

var ul = document.getElementById('demo');
var clone = ul.cloneNode(true); <--
for(var i = 0; i < 1e5; i++){
    var li = document.createElement('li');
    var text = document.createTextNode(i);
    li.appendChild(text);
    clone.appendChild(li); <--
}
ul.parentNode.replaceChild(clone,ul); <--
```

1:display:'none, 隐藏DOM节点
2:创建创建一个新的空白的文档片段（DocumentFragment），它不是主DOM树的一部分，因为文档片段存在于**内存中**，并不在DOM树中，所以将子元素插入到文档片段时不会引起页面重排（对元素位置和几何上的计算）。因此，使用文档片段通常会带来更好的性能。
3:cloneNode() 方法 拷贝所有属性和值。如果传递给它的参数是 true，它还将递归复制当前节点的所有子孙节点.
总结
脱离文档 => 改变演示 => 回归文档

<
——>

display:none、visibility:hidden和overflow:hidden的区别，顺便带一下
display:none:
隐藏元素，不占网页中的任何空间，让这个元素彻底消失（看不见也摸不着）,由于会影响到网页的空间，所以会引起一次重排和重绘。
visibility:hidden:
他是把那个层隐藏了，也就是你看不到它的内容但是它内容所占据的空间还是存在的。（看不见但摸得到），该操作不会对页面有影响，所以只会引起一次重绘。
overflow:hidden:
让超出的元素隐藏（不占据网页空间），就是在设置该属性的时候他会根据你设置的宽高把多余的那部分剪掉，会引起一次重排和重绘。
<
——
————————————————————————————————————>

动画脱离文档流
避免设置大量的style属性，因为通过设置style属性改变结点样式的话，每一次设置都会触发一
次reflow，所以最好是使用class属性实现元素的动画，设置position属性，最好是设为absoulte
或fixed，脱离文档流，这样不会影响其他元素的布局。

事件委托
事件委托原理：利用事件冒泡的特性，子元素都会冒泡到父元素上，当子元素如（li）元素都绑定事件时，可以只需绑定父元素（ul）来达到相同的效果，这样不仅减少了对dom的操作，减少

```
window.onload = function(){
    var oUl = document.getElementById("ul1");
    var aLi = oUl.getElementsByTagName('li');
    for(var i=0;i<aLi.length;i++){
        aLi[i].onclick = function(){
            alert(123);
        }
    }
}
```

委托

```
window.onload = function(){
    var oUl = document.getElementById("ul1");
    oUl.onclick = function(ev){
        var ev = ev || window.event;
        var target = ev.target || ev.srcElement;
        if(target.nodeName.toLowerCase() == 'li'){
                    alert(123);
                alert(target.innerHTML);
        }
    }
}
```

尽量不使用table布局

不要使用table布局，因为table中某个元素一旦触发了reflow，那么整个table的元素都会触发reflow。那么在不得已使用table的场合，可以设置table-layout:auto;或者是table-layout:fixed这样可以让table一行一行的渲染，这种做法也是为了限制reflow的影响范围.重排或重绘，而且不用分配大量变量来保存dom，减少了内存。

通过类名改样式

不要一个个修改属性，应通过一个class来修改

```
document.getElementsByClassName('outer')
[0].style.width="50px";
document.getElementsByClassName('outer')
[0].style.top="60px"
```

改为:

```
document.getElementsByClassName('outer')[0].className
+= ' news'
```