1 浏览器模型概述 :

JavaScript 是浏览器的内置脚本语言。浏览器的核心是两部分：渲染引擎和 JavaScript 解释器（又称 JavaScript 引擎）。

1.1

网页中嵌入 JavaScript 代码，主要有三种方法。

<script>元素直接嵌入代码。<script>标签加载外部脚本。URL 协议

1.2 script 元素 // 原则 - js 对dom 的操作, 一定要等待dom渲染完毕

浏览器一边下载 HTML 网页，一边开始解析。也就是说，不等到下载完，就开始解析。

解析过程中，浏览器发现<script>元素，就暂停解析，把网页渲染的控制权转交给 JavaScript 引擎。如果<script>元素引用了外部脚本，就下载该脚本再执行，否则就直接执行代码。

JavaScript 引擎执行完毕，控制权交还渲染引擎，恢复往下解析 HTML 网页。

1.3 使用defer 属性

为了解决脚本文件下载阻塞网页渲染的问题，一个方法是对<script>元素

加入defer属性。它的作用是延迟脚本的执行，等到 DOM 加载生成后，再执行脚本。

1.4 async 属性解决“阻塞效应”的另一个方法是对<script>元素加入async属性。

规定一旦脚本可用，则会异步执行。需要注意的是，一旦采用这个属性，就无法保证脚本的执行顺序。哪个脚本先下载结束，就先执行那个脚本。另外，使用async属性的脚本文件里面的代码，不应该使用document.write方法。

1.5 脚本的动态加载

2.0 浏览器的组成

渲染引擎和 JavaScript 解释器（又称 JavaScript 引擎）。

2.1 渲染引擎

渲染引擎的主要作用是，将网页代码渲染为用户可看到的效果。

渲染引擎处理网页，通常分成四个阶段。

解析代码：HTML 代码解析为 DOM，CSS 代码解析为 CSSOM（CSS Object Model）。

对象合成：将 DOM 和 CSSOM 合成一棵渲染树（render tree）。

布局：计算出渲染树的布局（layout）。

绘制：将渲染树绘制到屏幕。

以上四步并非严格按顺序执行，往往第一步还没完成，第二步和第三步就已经开始了。所以，会看到这种情况：网页的 HTML 代码还没下载完，但浏览器已经显示出内容了。

2.2 JavaScript 引擎

JavaScript 引擎的主要作用是，读取网页中的 JavaScript 代码，对其处理后运行。

JavaScript 是一种解释型语言，也就是说，它不需要编译，由解释器实时运行。

这样的好处是运行和修改都比较方便，刷新页面就可以重新解释；

缺点是每次运行都要调用解释器，系统开销较大，运行速度慢于编译型语言。

URL参数传递

document.getElementById("windowOpen").onclick = function(){

\_son = window.open("20190716-2.html", "\_self" );

}

Window-open

document.getElementById("windowOpen").onclick = function(){

\_son = window.open("20190716-2.html", "\_self" );

}

History：

document.getElementById("goBack").onclick = function(e){

// window.history.go(-1); // 返回上一页

window.history.back(); // 返回上一页

}

location ：

document.getElementById( "bt1").onclick = function(){

alert(1);

window.location = "http://fangwentao.com";

}

onload ：

window.onload = function(){

\_son = window.open("20190716-3.html");

}

Onunload:

window.onunload = function(){

\_son.close();

}

Reload：

document.getElementById("reload").onclick = function(){

location.reload();

}