第一章---加载与执行

脚本位置

浏览器在解析到<body>标签之前，不会渲染页面的任何部分。把脚本放到页面顶部会导致明显的延迟，通常表现为显示空白页面，用户无法浏览内容，也无法和页面进行交互。

尽管IE8，FX3.5，Safari4和Chrome2+都允许并行下载JavaScript文件。这样<script>标签在下载外部资源时不会阻塞其他<script>标签。遗憾的是，JavaScript下载过程中任然会阻塞其他资源的下载，比如图片。

由于脚本会阻塞其他资源的下载，因此推荐将所有<script>标签尽可能放到<body>标签的底部，以尽量减少对整个页面下载的影响。

组织脚本

由于每个<script>标签初始下载时都会阻塞页面渲染，所以减少页面包含<script>标签数量有助于改善这一情况。这不仅仅针对外链脚本，内嵌脚本的数量也要同样的限制。浏览器在解析HTML页面的过程中每遇到一个<script>标签，都会因执行脚本而导致一定的延迟，因此最小化延迟时间将会明显地改善页面的性能。

因此下载单个100kb的文件比下载4个25kb的文件更快。

无阻塞下载JavaScript的方法

延迟的脚本

HTML4为<script>标签定义了一个拓展属性：defer。Defer属性指明本元素所含的脚本不会修改DOM，因此代码能够安全地延迟执行。对应的JavaScript文件将在页面解析到这个<script>标签时开始下载，但不执行，直到DOM下载完成(onload事件被触发前)，因而它不会阻塞其他进程，此类文件可以与页面其他资源并行下载。

经过我的测试发现只有在外链脚本时defer才起到作用，内嵌脚本没有起到作用，我用的浏览器为Chrome。

动态创建脚本

动态创建script脚本后，文件会在被添加到页面时立即下载。它的重点在于：文件的下载和执行过程不会阻塞页面其他进程，在下载文件时里面的代码会立即执行。

这种情况下，当下载的代码包含其他脚本要调用的接口或者方法时，就会有问题，所以我们需要跟踪并确保脚本下载完毕且准备就绪。FX，Opera，Chrome和Safari 3+的版本会在script元素接收完成时触发load事件，因此，你可以通过监听此事件来判断是否下载完毕。

具体代码如下：

function loadScript(url, callback) {

var script = document.createElement("script");

script.type = "text/javascript";

script.src = url;

script.onload = function () {

callback();

};

document.getElementsByTagName("head")[0].appendChild(script);

}