前端性能优化的14个规则

规则01：尽量减少HTTP请求

前端优化的黄金准则指导着前端页面的优化策略：只有10%-20%的最终用户响应时间花在接受请求的HTML文档上，剩下的80%-90%时间花在为HTML文档所引用的所有组件（图片、脚本、样式表等）进行的HTTP请求上。因此，改善响应时间的最简单途径就是减少组件的数量，并由此减少HTTP请求的数量。当然很多人就会说，既然这样，那我们就减少页面组件的数量不就OK了吗？那你试试，你会掀起一场性能优化和产品设计之间的大PK。所以，我们要减少HTTP请求是要平衡性能和设计的。如果找到这个平衡点呢？书中从以下几个方面做了介绍，我逐一说明：

① 图片地图

初看“图片地图”四个字，对非专业的前端人员来说一头雾水，我的第一印象就是这样的，咱们以京东的移动站点为例，右侧用户和购物车的图标，正常实现我会选择如下方式：

<a href=”用户跳转页面URL”>

```
    <div class="定义用户icon显示的样式表"></div>
```


<a href=”购物车跳转页面URL”>

```
    <div class=" 定义用户icon显示的样式表"></div>
```


这种方式无可厚非的，但是两张图片就有两个HTTP请求，这明显是增加了页面中的HTTP请求。

那么我们可以把这两个HTTP请求变成一个吗？答案当然是可以的，这就是图片地图：允许在一张图片上关联多个URL，而目标URL的选择取决于用户单击了图片上的哪个位置。这样上面京东两个图标合并成一张图片，这样图片的HTTP请求就减少了一个。

示例代码如下：

```
<img src="合并后的图片">

<map name="map1">

        <area shape="rect" coords="0,0,40,40" href="用户跳转页面URL">

        <area shape="rect" coords="50,0,90,40" href="购物车跳转页面URL">
```

```
</map>
```

不过图片地图只支持矩形形状，其他形状不支持。

② 请CSS喝“雪碧”（CSS Sprites）

CSS Sprites一句话：将多个图片合并到一张单独的图片，这样就大大减少了页面中图片的HTTP请求。

③ 内联图片和脚本

使用data:URL（Base64编码）模式直接将图片包含在Web页面中而无需进行HTTP请求。

但是此种方法存在明显缺陷：

- 不受IE的欢迎；
- 图片太大不宜采用这种方式，因为Base64编码之后会增加图片大小，这样页面整体的下载量会变大；
- 内联图片在页面跳转的时候不会被缓存。（大图片可以使用浏览器的本地缓存，在首次访问的时候保存到浏览器缓存中，典型的是HTML5的manifest缓存机制以及LocalStorage等）。

④ 样式表的合并

将页面样式定义、脚本、页面本身代码严格区分开，但是样式表、脚本也不是分割越细越好，因为没多引用一个样式表就增加一次HTPP请求，能合并的样式表尽量合并。一个网站有一个公用样式表定义，每个页面只要有一个样式表就OK啦。

通过以上四个努力之后，你会发现你的网页响应时间最多能减少一半，这不是作者说大话，也不是我狂吹，我亲手用我的移动网站首页做了一个尝试，本地测试之后响应时间能减少40%左右。所以减少页面HTTP请求数量，是一个很重要的原则。遵循此原则可以同时改善首次访问和后续访问的响应时间，而每一个网站的首次响应时间会决定用户之后还来不来的重要原因。

规则02：使用内容发布网络（CDN的使用）

什么叫内容发布网络（CDN）？它是一组分布在多个不同地理位置的Web服务器，用于更加有效地向用户发布内容。主要用于发布页面静态资源：图片、css文件、js文件等。如此，能轻易地提高响应速度。

关于CDN的具体详细原理以及优缺点，各位可以自行询问度娘或者google。

规则03：添加Expires头

浏览器使用缓存来减少HTTP请求的数据，并减小HTTP响应的大小，使页面加载更快。Web服务器使用Expires头来告诉浏览器它可以使用一个组件的当前副本，直到指定的deadline为止。HTTP规范中称此头为：在这一时间之后响应被认为失效。

个人对这块表示不想使用，其实就是一句话，把一些css、js、图片在首次访问的时候全部缓存到浏览器本地，从我做移动网站的过程中发现，其实没有这么复杂，完全可以使用HTML5提供的本地缓存机制就OK了。关于HTML5本地缓存机制，各位可以查阅相关资料。后续我也会对HTML5的缓存机制进行介绍的。

规则04：压缩组件（使用Gzip方式）

书中关于压缩从gzip压缩方式到如何压缩讲了很多，我想直接跳过，对于做PC网站或者移动网站来说，急需要压缩的是css文件和js文件，至于如何压缩，网上有很多在线工具，去挑选一个自己用的顺手看的顺眼的就好，当然也有人选择对HTML进行压缩，这样也可以。但是实际工作中我没有这么做。之所谓没有这么做，是因为我觉得很麻烦。不要鄙视我，毕竟我不是一个真正意义上的前端工程师，哈哈！

规则05：将CSS样式表放在顶部

如果将css样式定义放在页面中或者页面底部，会出现短暂白屏或者某一区域短暂白板的情况，这和浏览器的运营机制有关的，不管页面如何加载，页面都是逐步呈现的。所以在每做一个页面的时候，用Link标签把每一个样式表定义放在head中。

规则06：将javascript脚本放在底部

浏览器在加载css文件时，页面逐步呈现会被阻止，直到所有css文件加载完毕，所以要把css文件的引用放到head中去，这样在加载css文件时不会组织页面的呈现。但是对于js文件，在使用的时候，它下面所有也页面内容的呈现都会被阻塞，将脚本放在页面越靠下的地方，就意味着越多的内容能够逐步呈现。

规则07：避免使用CSS表达式

CSS表达式是动态玩CSS的一种很强大的方式，但是强大的同时也存在很高的危险性。因为css表达式的频繁求值会导致css表达式性能低下。如果真想玩css表达式，可以选用只求值一次的表达式或者使用事件处理来改变css的值。

规则08：使用外部javascript和CSS

内联js和css其实比外部文件有更快的响应速度，那为什么还要用外部呢？因为使用外部的js和css可以让浏览器缓存他们，这样不仅HTML文档大小减少，而且不会增加HTTP请求数量。

另外，使用外部js和css可以提高组件的可复用性。

规则09：减少DNS查询

DNS查询有时间开销，通常一个浏览器查找一个给定主机名的IP地址需要20-120ms。

缓存DNS：缓存DNS查询可以很好地提高网页性能，一旦缓存了DNS查询，之后对于相同主机名的请求就无需进行再次的DNS查找，至少短时间内不需要。

所以在使用页面中URL、图片、js文件、css文件等时，不要使用过多不同的主机名。

规则10：精简javascript

如何精简？最初始的精简方式就是移除不必要的字符减小js文件大小，改善加载时间。包括所有的注释、不必要的空白字符。

高级一点的精简方式就是：混淆。它不但会移除不必要的字符，还会改写代码，比如函数和变量的名字会被改成很短的字符串，这样使js代码更简练更难阅读。

但是我一般很少使用混淆，一个现在互联网时代，代码没有必要整的那么神秘，大可以大家一起share，天下代码一起抄，只要抄出自己的特色就ok了。而且一旦使用混淆，对于js代码的维护和调试都很复杂，因为有时候混淆之后的js代码完全看不懂。

其实实际开发过程中，从文件大小和代码可复用性来说，不仅仅是js代码需要精简，css代码一样也很需要精简。

规则11：避免重定向

重定向的英文是Redirect，用于将用户从一个URL重新跳转到另一个URL。最常见的Redirect就是301和302两种。

关于重定向的性能影响这里就不说了，自行查阅相关资料吧。

在我们实际开发中避免重定向最简单也最容易被忽视的一个问题就是，设置URL的时候，最后的“/”，有些人有时候会忽略，其实你少了“/”，这时候的URL就被重定向了，所以在给页面链接加URL的时候切记最后的“/”不可丢。

规则12：删除重复脚本

重复的js代码除了有不必要的HTTP请求之外，还会浪费执行js的时间。
将你使用的js代码模块化，可以很好地避免这个问题，至于js模块化如何实现，现在有很多可以使用的开源框架，我用的比较多的是我们公司玉伯的Sea.js。

规则13：配置ETag

Etag（Entity Tag），实体标签，是Web服务器和浏览器用户确认缓存组件的有效性的一种机制。
写的很复杂，对我这种非专业的前端开发人员来说，有点过了，关于这个原则有兴趣的自己看吧。

规则14：使Ajax可缓存

针对页面中主动的Ajax请求返回的数据要缓存到本地，当然这个是针对短期内不会变化的数据。如果不确定数据变化周期的话，可以增加一个修改标识的判断，我正常处理过程中会给一些Ajax请求返回的数据增加一个MD5值的判断，每次请求会判断当前MD5是否变化，如果变化了取最新的数据，如果不变化，则不变。
做前端页面，尤其是移动网站的页面，我闷所记住的准则就是：尽量减少页面大小，尽量降低页面响应时间。在页面性能和交互设计之中找平衡点。