**程序员都是懒孩子，想直接看自动优化的点：**[**传送门**](http://cabbit.me/web-image-optimization/#auto)

[HTTP Archieve](http://httparchive.org/interesting.php#bytesperpage)有个统计，图片内容已经占到了互联网内容总量的62%，也就是说超过一半的流量和时间都用来下载图片。从性能优化的角度看，图片也绝对是优化的热点和重点之一，Google PageSpeed或者Yahoo的14条性能优化规则无不把图片优化作为重要的优化手段，本文覆盖了Web图片优化的方方面面，从基本的图片格式选择、到尚未被广泛支持的响应式图片均有所提及。

Google Web Fundamentals的说法我很喜欢：

*图片优化既是一门艺术，也是一门科学，图片优化是一门艺术，是因为单个图片的压缩不存在最好的特定性方案，而图片优化之所以是一门科学，是因为许多开发得很出色的方法和算法可以明显减小图片的大小。要找到图片的最优设置，需要按照许多维度进行认真分析：格式能力、编码数据内容、像素尺寸等。*

**真的要用图片吗？**

要实现需要的效果，真的需要图片吗？这是首先要问自己的问题。浏览器和Web标准的发展速度极快，记得数年前我在用微软Silverlight 1.0写视频播放器的时候，中文还不能使用自定义字体显示，所以那时候写了很多糟糕的代码把需要的文字在服务器上生成图片并缓存起来。用户下载起来很慢，搜索引擎也完全无法检索这些文字。

但是现在不一样了，很多特效（渐变、阴影、圆角等等）都可以用纯粹的HTML、CSS、SVG等加以实现，实现这些效果少则寥寥数行代码，多则加载额外的库（一张普通的照片比[非常强大的效果库](http://daneden.github.io/animate.css/)也大了许多）。这些效果不但需要的空间很小，而且在多设备、多分辨率下都能很好的工作，在低级浏览器上也可以实现较好的功能降级。因此在存在备选技术的情况下，应该首先选择这些技术，只有在不得不使用图片的时候才加入真正的图片。

**备选技术**

* CSS效果、CSS动画。提供与分辨率无关的效果，在任何分辨率和缩放级别都可以显示得非常清晰，占用的空间也很小。
* 网络字体。现在连很多图标库都是用字体方式提供，保持文字的可搜索性同时扩展显示的样式。

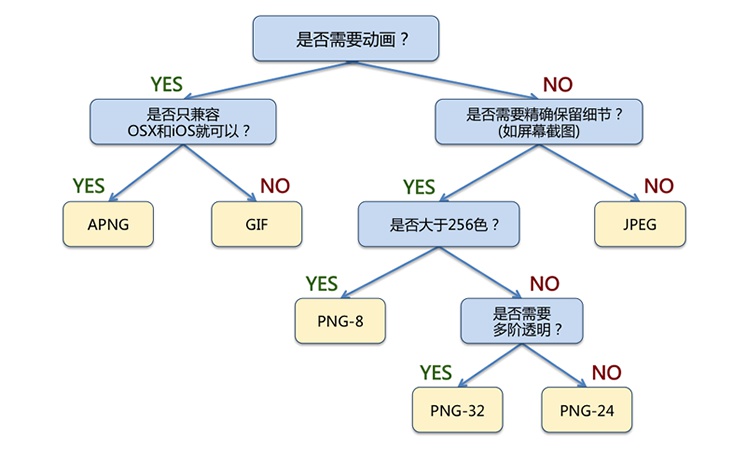
**前端工程师最好能和设计师、产品经理保持沟通**，帮助他们了解到什么样的效果比较“简洁、高效、可维护”，毕竟对于CSS来说改变圆角矩形的Radius可以实时看到效果，用图片的话至少要重新生成图片、切图并替换资源。Retina、高分辨率屏幕、多尺寸的设备，这些都加快了非图片特效的发展，想想在高分辨率屏幕下Windows 7的惨不忍睹，就知道原生的图片资源绝对不是多多益善。

**图片格式的选择**

如果效果真的需要图片来表现，那么选择图片格式是优化的第一步。我们经常听到的词语包括矢量图、标量图、SVG、有损压缩、无损压缩等等，我们首先说明各种图片格式的特点

| **图片格式** | **压缩方式** | **透明度** | **动画** | **浏览器兼容** | **适应场景** |
| --- | --- | --- | --- | --- | --- |
| JPEG | 有损压缩 | 不支持 | 不支持 | 所有 | 复杂颜色及形状、尤其是照片 |
| GIF | 无损压缩 | 支持 | 支持 | 所有 | 简单颜色，动画 |
| PNG | 无损压缩 | 支持 | 不支持 | 所有 | 需要透明时 |
| APNG | 无损压缩 | 支持 | 支持 | Firefox Safari iOS Safari | 需要半透明效果的动画 |
| WebP | 有损压缩 | 支持 | 支持 | Chrome Opera Android Chrome Android Browser | 复杂颜色及形状 浏览器平台可预知 |
| SVG | 无损压缩 | 支持 | 支持 | 所有（IE8以上） | 简单图形，需要良好的放缩体验 需要动态控制图片特效 |

其中APNG和WebP格式出现的较晚，[尚未被Web标准所采纳](http://caniuse.com/" \l "search=apng)，只有在特定平台或浏览器环境可以预知的情况下加以采用，虽然均可以在不支持的环境中较好的功能降级，但本节暂不讨论这两种格式。图片格式选择过程如下：



**颜色丰富的照片，JPG是通用的选择**

* 人眼的结构很适合查看JPG压缩后的照片，可以充分的忽略并在脑中补齐细节
* JPG在压缩率不高时保留的细节还是不错的
* WebP能够比JPG减少30%的体积，但目前[兼容性较差](http://caniuse.com/" \l "search=webp)

**如果需要较通用的动画，GIF是唯一可用的选择**

* GIF支持的颜色范围为256色，而且仅支持完全透明/完全不透明
* GIF在显示颜色丰富的动画时可能出现颜色不全、边缘锯齿等问题

**如果图片由标准的几何图形组成，或需要使用程序动态控制其显示特效，可以考虑SVG格式**

* SVG是使用XML定义的矢量图形，生成的图片在各种分辨率下均可自由放缩
* SVG中可以通过JavaScript等接口自由变换图片特效，可以完成其中部分元素的自由旋转、移动、变换颜色等

**如果需要清晰的显示颜色丰富的图片，PNG比较好**

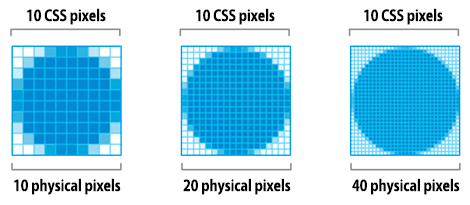
* PNG-8能够显示256种颜色，但能够同时支持256阶透明，因此颜色数较少但需要半透明的情景（如微信动画大表情）可以考虑PNG-8
* PNG-24可以显示真彩色，但不支持透明，颜色丰富的图片推荐使用（如屏幕截图、界面设计图）
* PNG-32可以显示真彩色，同时支持256阶透明，效果最好但尺寸也最大

**图片尺寸的选择**

尺寸，曾经是最不需要讨论的话题，但自从Retina出现之后世界就变得复杂多了。关于移动设备上的像素和尺寸，展开说足够写一篇论文，我建议想详细了解的同学参考下面的文章：

[浅谈移动Web开发（上）：深入概念](http://www.infoq.com/cn/articles/development-of-the-mobile-web-deep-concept)

这里只说我们关心的部分和结论，我们需要分清不同类型的像素：CSS像素和设备像素。一个 CSS像素可能包含多个设备像素。对于图片来说，在高DPI的屏幕上需要使用分辨率更高的图片，如果我们讨论的是Retina，那么就需要2倍分辨率（几乎4倍尺寸）的图片。这几乎没有取巧的空间，屏幕就是那么大，需要的图片也就是那么大。（鸽子为什么那么大？^\_^）



我们能够控制的地方是“恰好”显示所需尺寸的图片。例如在屏幕中通过CSS或者标签的wihth/height属性，将一副200×200的图片调整为100×100大小，那么这其中就有(200×200)-(100×100)=30000个像素是浪费的，这占到了图片尺寸的75%！

之所以有这么大的浪费，是因为图片的尺寸与面积基本成正比，与宽高的平方成正比。因此良好的计算客户端实际显示的图片尺寸，能够大大减小图片的大小。即使只有长和宽都只有10px被浪费，但是当图片足够大时，这部分也将产生很大影响。

**响应式图片**

上面提到“恰好”显示客户端所需大小的图片，听上去很容易不是吗？但当响应式布局出现后，这就变得极其困难。我们要支持上至1920宽度，下至320宽度的无数种设备，如果使用1920宽度的图片，那么在小型设备（这类设备往往对网速和流量更加敏感）上每个用户都要付出额外的带宽和等待时间，如果使用320宽度的图片，那么在1920的屏幕上就像是在高清屏上使用DOS那么让人难以接受。

很自然的，我们需要图片也能“响应式”加载，根据所在设备的不同，加载不同尺寸的图片。响应式图片尚没有写入Web标准，实现起来也有诸多不便和兼容性限制。我建议参考百度EFE团队的这篇文章：

[实战响应式图片](http://efe.baidu.com/blog/responsive-images-in-practice/)

响应式图片虽然[尚未成为标准](http://caniuse.com/" \l "search=srcset)，但这是Web图片优化的一柄利器，一旦被广泛支持，再没有比缩小图片尺寸更有效的优化方法了。

**优化JPG和PNG**

选择了正确的图片格式，按照正确的大小生成了图片后，我们还需要对图片进行进一步优化，这种优化一般分两步进行：

1. 有损优化，删除没有出现或极少出现过的颜色，合并相邻的相近颜色。这一步并不必须，如PNG格式就直接进入下一步
2. 无损优化，压缩数据，删除不必要的信息

JPG和PNG格式的图片生成后，一般还有进一步优化的空间，例如JPG格式的照片中，可能携带有相机的Exif信息，PNG格式的图片中可能带有Fireworks等软件的图层信息等。去除这些额外信息后，还可以通过减小图片的调色板，去除没有出现过的颜色，以及合并相邻的相同颜色等手段来进行优化。原理性的内容这里不再赘述，仅介绍工程中可用的优化工具。

不同格式的图片有一系列工具，这些工具有有更多种参数调节方案，常见的几种调节工具有：

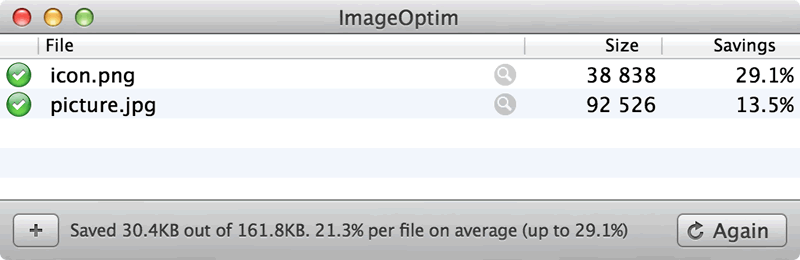
| **工具** | **用途** |
| --- | --- |
| [jpegtran](http://jpegclub.org/jpegtran/) | 优化JPG图片 |
| [OptiPNG](http://optipng.sourceforge.net/) | 无损PNG优化 |
| [AdvPNG](http://advancemame.sourceforge.net/doc-advpng.html) | 无损PNG优化 |
| [PNGQuant](http://pngquant.org/) | 有损PNG优化 |

如果你真的需要追求各种图片的极限压缩，可以参阅这些工具的文档，但是对于一般的Web应用，面对的图片种类多样，几乎不可能在工程中实现对每种工具的独立配置，因此推荐使用以下工具来进行优化。这些工具往往使用了上表中的一种或几种优化工具。

**ImageOptim (Mac)**

主页：<https://imageoptim.com/>

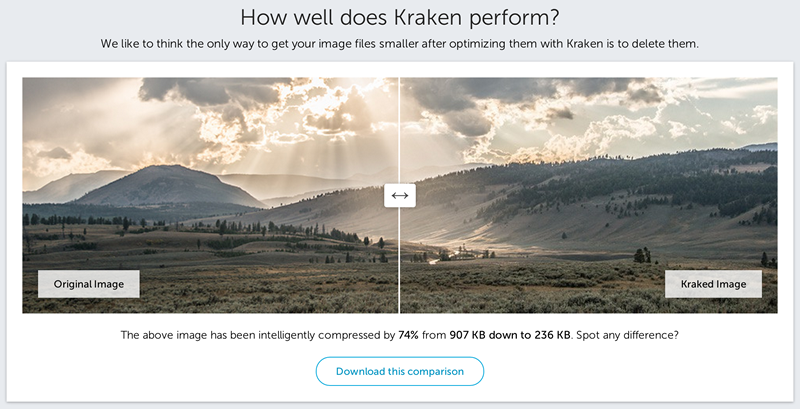
Mac平台下非常赞的图片优化工具，只需要把需要优化的图片拖拽进ImageOptim，就能够完成对图片的优化。设置选择的也很丰富，目前支持JPG和PNG的优化。这是我在写文章时最常用到的工具，把网站用到的图片拖进去，优化就完成了~



**Kraken (Web)**

主页：<https://kraken.io/>

在免费模式下可以上传图片，优化后打包下载，很多国外企业也选择了它的收费服务。亲自测试Kraken的图片优化结果比ImageOptim一般要小3%左右，效果不错，当然价格也不错。适合偶尔有图片优化需求，或者不在开发机上没有优化软件可以使用的情况。

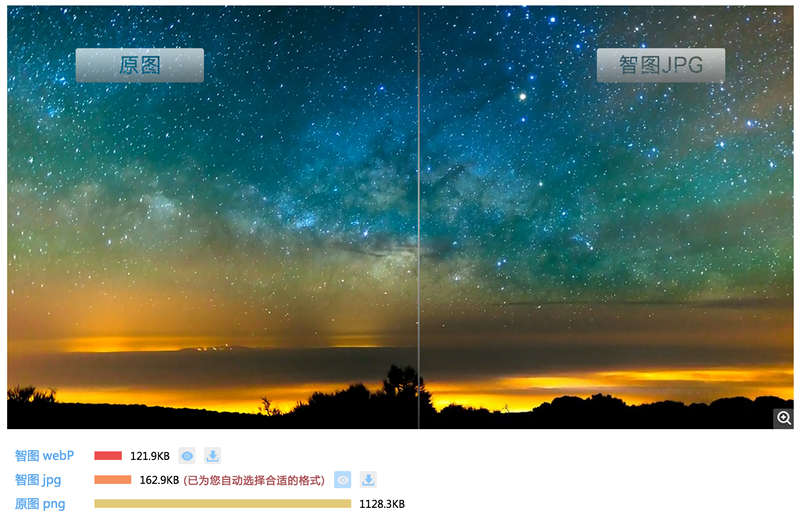


**智图 (Web)**

主页：<http://zhitu.tencent.com/>

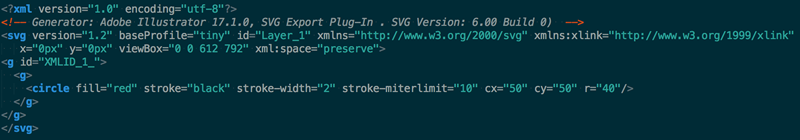
腾讯ISUX团队有篇文章介绍智图：<http://isux.tencent.com/zhitu.html>

国货当自强，腾讯的智图工具推出不久，但实测效果很好，而且提供了[Gulp的自动化支持](https://www.npmjs.org/package/gulp-imageisux)，这部分会在后面自动优化章节介绍。只想建议一句，Kraken的首页比智图美好几百倍…… 而且把压缩前的PNG和压缩后的JPG放在一起对比大小，真的没关系么~



**优化SVG**

[IE9及以上的浏览器](http://caniuse.com/#search=svg)都支持可缩放矢量图(SVG)，SVG是基于XML的图片格式，适用于二维图片。可以将SVG标记直接嵌入网页，也可以作为外部资源嵌入。可以通过大多数基于矢量的绘图软件创建SVG文件。这是一段简单的SVG图形：



这个圆形轮廓为黑色，背景为红色，从Adobe Illustrator直接导出。可以从中看到大量元数据，例如图层信息、注释和XML名称空间等等，在浏览器中呈现资源时，通常不需要这些数据。因此我们需要使用一些工具去除这些不必要的元数据，仅保留必须的标记。

[SVGO](https://github.com/svg/svgo)工具可以缩减SVG文件的体积，在这个的例子中，SVGO能够将Illustrator生成的SVG文件大小减小58%，从470字节缩减到199字节。

在我自己的网站中，对于网站Logo一类的图标比较倾向于使用SVG，而且由于SVG是基于XML的格式，本质上是纯文本，所以，还可以采用GZIP压缩来进一步减小传输大小，当然这需要一些服务器配置，例如在apache服务器中设置：

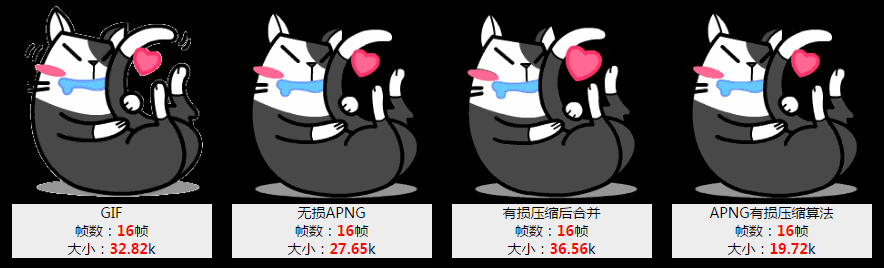
|  |  |
| --- | --- |
| 1  2  3 | AddType image/svg+xml .svg  AddOutputFilterByType DEFLATE image/svg+xml |

来对SVG文件启用GZip压缩（当然你还需要先加载deflate模块并进行适当配置，GZip的配置超出了本文的范畴，这部分内容请自行Google）

**优化GIF和APNG**

GIF有很多好处，在颜色数较低的时候能够大幅减小图片体积，而且他也是唯一能够较为通用的展示动画的图片格式。关于GIF格式的优化原理我并不熟悉，只是在工程中直接使用成型的压缩工具，在后文[自动优化](http://cabbit.me/web-image-optimization/" \l "auto)章节的Grunt中，会介绍通过Grunt Task进行自动优化的方法。

关于APNG，目前浏览器对他的支持还不够好，不过在支持HTML5的场景中，有成熟的开源工具[apng-canvas](https://github.com/davidmz/apng-canvas)可以用于支持APNG。



腾讯ISUX团队有篇文章介绍[iSparta](https://github.com/iSparta/iSparta)工具：<http://isux.tencent.com/introduction-of-apng.html>。这是目前几乎唯一能够批量处理APNG文件的工具，感兴趣的同学可以在那篇文章里得到更多地了解。

**自动优化**

前面说了太多关于如何优化各种不同格式图片的方法和工具，优化图片需要大量重复性的劳动，作为工程师显然不会忍受这一点，因此也产生出了很多工具对图片进行自动优化，这里主要介绍CDN、Grunt/Gulp、Google PageSpeed三种方式。

**自动优化：CDN**

使用CDN对图片自动进行优化，我在国外的CDN提供商处很少见到这类服务，倒是国内的两大新秀CDN[七牛](http://www.qiniu.com/)和[又拍](https://www.upyun.com/index.html)在这方面都做了大量工作。其工作方式为，向CDN请求图片的URL参数中包含了图片处理的参数（格式、宽高等），CDN服务器根据请求生成所需的图片，发送到用户浏览器。

七牛云存储的[图片处理接口](http://developer.qiniu.com/docs/v6/api/reference/fop/image/)极其丰富，覆盖了图片的大部分基本操作，例如：

* 图片裁剪，支持多种裁剪方式（如按长边、短边、填充、拉伸等）
* 图片格式转换，支持JPG, GIF, PNG, WebP等，支持不同的图片压缩率
* 图片处理，支持图片水印、高斯模糊、重心处理等

七牛云存储的图片处理接口使用并不复杂，例如下面这张原图：



我们通过如下URL请求，裁剪正中部分，等比缩小生成200×200缩略图：

|  |  |
| --- | --- |
| 1  2 | http://qiniuphotos.qiniudn.com/gogopher.jpg?imageView2/1/w/200/h/200 |

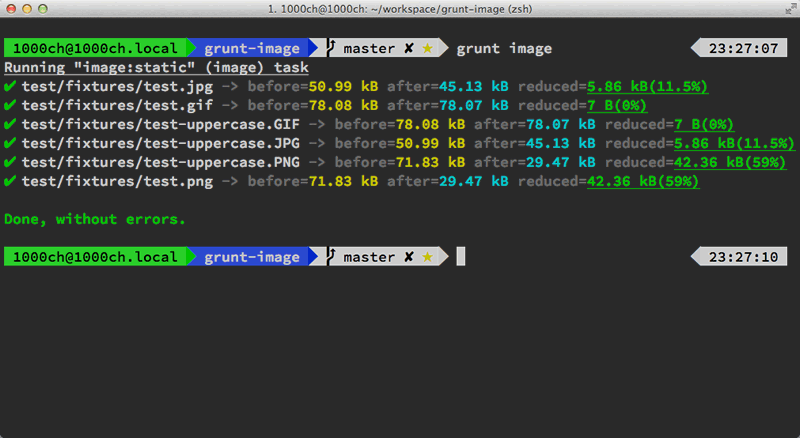


**自动优化：Grunt/Gulp**

这里介绍用于图片优化的Grunt组件：[grunt-image](https://github.com/1000ch/grunt-image)。前端工程师的重复性工作，例如合并静态资源、压缩JS和CSS文件、编译SASS等都可以使用Grunt等自动化工具批量完成，图片优化也是如此。

grunt-image非常强大，按照作者的介绍，其内部加载的图片优化工具包括了pngquant, optipng, advpng, zopflipng, pngcrush, pngout, mozjpeg, jpegRecompress, jpegoptim, gifsicle和svgo。支持批量自动优化PNG, JPG, SVG和GIF，速度也不错，配置方式支持单图片优化和全目录优化：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40 | module.exports = function (grunt) {    grunt.initConfig({      image: {        // 指定单独的图片优化        static: {          options: {            pngquant: true,            optipng: true,            advpng: true,            zopflipng: true,            pngcrush: true,            pngout: true,            mozjpeg: true,            jpegRecompress: true,            jpegoptim: true,            gifsicle: true,            svgo: true          },          files: {            'dist/img.png': 'src/img.png',            'dist/img.jpg': 'src/img.jpg',            'dist/img.gif': 'src/img.gif',            'dist/img.svg': 'src/img.svg'          }        },        // 指定图片目录进行优化        dynamic: {          files: [{            expand: true,            cwd: 'src/',            src: ['\*\*/\*.{png,jpg,gif,svg}'],            dest: 'dist/'          }]        }      }    });      grunt.loadNpmTasks('grunt-image');  }; |



**自动优化：Google PageSpeed**

Google做事风格比较彻底，看见哪个软件不好用就拿来直接fork出新版本或者干脆重写，对于Web优化，Google发布了了[Google PageSpeed](https://developers.google.com/speed/pagespeed/module)这个服务器模块，可以在apache或ngnix中加载，通过在服务器配置文件中进行设置来进行自动化的优化。对于图片格式转换、图片优化甚至图片LazyLoad都有相关选项。这部分展开会非常长，请感兴趣的同学参考Google的手册。