Join us for the first **#AskChrome Live** session to learn about Implementing Performance Budgets and live Q&A. **Register now** (https://events.withgoogle.com/askchrome-live/registrations/new/)

图像优化

By Ilya Grigorik
(https://developers.google.com/web/resources/contributors/ilyagrigorik?hl=zh-cn)
Ilya is a Developer Advocate and Web Perf Guru

图像通常占据了网页上下载字节的大部分，通常也占据了大量的视觉空间。 因此，优化图像通常可以最大限度地减少从网站下载的字节数以及提高网站性能：浏览器需要下载的字节越少，占用客户端的带宽就越少，浏览器下载并在屏幕上渲染有用内容的速度就越快。

图像优化既是一门艺术，也是一门科学：说它是一门艺术，是因为单个图像的压缩并不存在明确的最佳方案，说它是一门科学，则是因为有许多发展成熟的方法和算法都能够显著缩减图像的大小。 找到图像的最佳设置需要在许多方面进行认真分析：格式能力、编码数据的内容、质量、像素尺寸等。

消除和替换图像

TL;DR

- 消除多余的图像资源
- 尽可能利用 CSS3 效果
- 使用网页字体取代在图像中进行文本编码

首先要问问自己，要实现所需的效果，是否确实需要图像。 好的设计应该简单，而且始终可以提供最佳性能。 如果您可以消除图像资源（与 HTML、CSS、JavaScript 以及网页上的其他资产相比，需要的字节数通常更大），这种优化策略就始终是最佳策略。 不过，如果使用得当，图像传达的信息也可能胜过千言万语，因此需要由您来找到平衡点。

接下来您应该考虑是否存在某种替代技术，能够以更高效的方式实现所需的效果：

- **CSS 效果**（渐变、阴影等）和 CSS 动画可用于产生与分辨率无关的资产，这些资产在任何分辨率和缩放级别下始终都能清晰地显示，并且需要的字节数往往只是图片文件的几分之一。
- **网页字体**可以在保留选择文本、搜索文本和调整文本大小能力的同时使用漂亮的字体，大大提高了易用性。

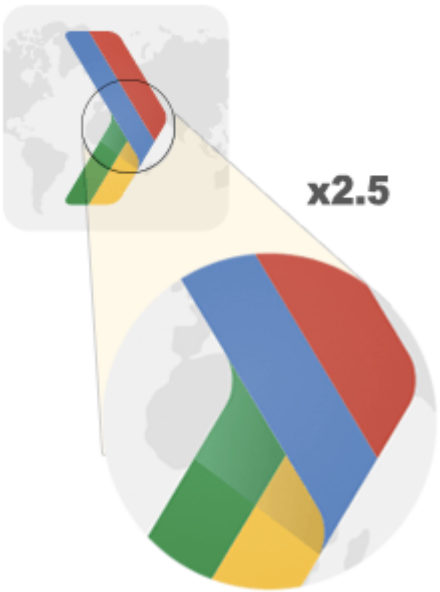
如果您发现自己正在图像资产中编码文本，不妨停下来重新考虑一下。 出色的字体排印对实现良好的设计、品牌推广和可读性至关重要，但图像内文本提供的用户体验较差：无法选择、搜索、缩放、访问文本，并且文本也不适用于高 DPI 设备。 使用网页字体需要其自己的一组优化 (https://www.igvita.com/2014/01/31/optimizing-web-font-rendering-performance/)，但它可以解决所有上述问题，因此对显示文本而言，始终是更好的选择。

矢量与 光栅图像

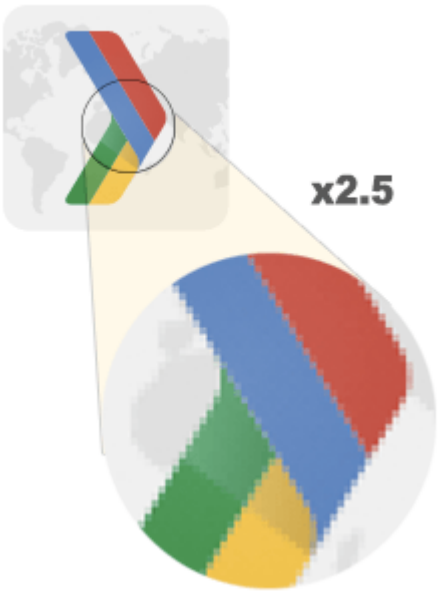
TL;DR

- 矢量图像最适用于包含几何形状的图像
- 矢量图像与缩放和分辨率无关
- 光栅图像应用于包含大量不规则形状和细节的复杂场景

一旦您确定了图像确实是实现所需效果的最佳格式，接下来的关键选择就是选择合适的格式：



放大后的矢量图像



放大后的光栅图像

- 矢量图形 (https://en.wikipedia.org/wiki/Vector_graphics)使用线、点和多边形来表示图像。
- 光栅图形 (https://en.wikipedia.org/wiki/Raster_graphics)通过对矩形格栅内的每个像素的值进行编码来表示图像。

每一种格式都各有其优缺点。 矢量格式最适用于包含简单几何形状（例如徽标、文本、图标等）的图像，能够在任何分辨率和缩放设置下呈现清晰的效果，因此这种格式最适用于高分辨率屏幕和需要以不同尺寸显示的资产。

不过，如果场景复杂（例如照片），矢量格式就不能满足要求了：描述所有形状所需的 SVG 标记量可能高得离谱，即便如此，输出效果可能仍然无法达到“照片级真实感”。 如果出现这种情况，您就应该使用光栅图片格式（例如 GIF、PNG、JPEG 或 JPEG-XR 和 WebP 等某种较新的格式）。

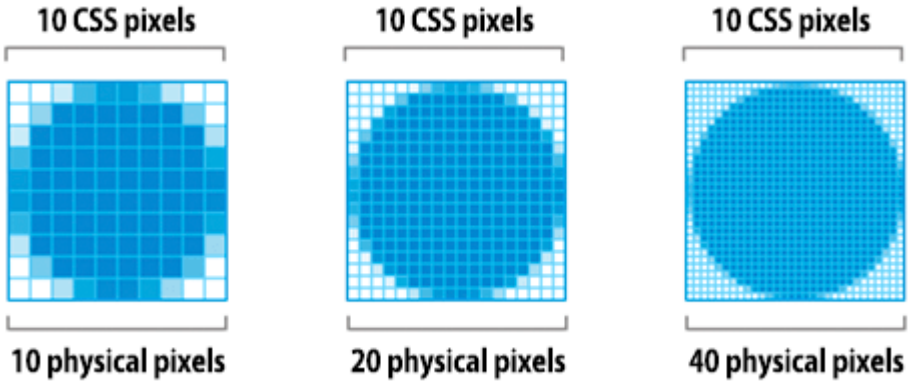
光栅图像没有与分辨率或缩放无关这么好的属性，当您放大光栅图像时，图形会出现锯齿并且模糊不清。 因此，您可能需要在不同分辨率下保存多个版本的光栅图像，以便为用户提供最佳体验。

高分辨率屏幕的含义

TL;DR

- 高分辨率屏幕的每个 CSS 像素包含多个设备像素
- 高分辨率图像需要的像素数和字节数要多得多
- 任何分辨率都采用相同的图像优化方法

在我们谈论图像像素时，需要分清不同类型的像素：CSS 像素和设备像素。 单个 CSS 像素可能包含多个设备像素。例如，单个 CSS 像素可能直接对应于单个设备像素，也可能依托于多个设备像素。 这是什么意思？那就是，设备像素越多，屏幕上所显示内容的细节就越丰富。



高 DPI (HiDPI) 屏幕可以产生绚丽的效果，但也有一个明显的折衷之处：图像资产需要更多的细节，才能对更高的设备像素数加以利用。 好在矢量图像最适用于这项任务，因为它们在任何分辨率下都能渲染出清晰的效果。为了渲染出更丰富的细节，我们可能会招致更大的处理开销，但基础资产是相同的，并且与分辨率无关。

另一方面，因为光栅图像是以像素为单位编码图像数据，所以面临的挑战要大得多。 因此，像素数越大，光栅图像的文件大小就越大。 例如，让我们看一看以 100x100 (CSS) 像素显示的照片资产之间的差异：

屏幕分辨率	总像素数	未压缩文件大小（每像素 4 字节）
1x	100 x 100 = 10,000	40,000 字节
2x	100 x 100 x 4 = 40,000	160,000 字节
3x	100 x 100 x 9 = 90,000	360,000 字节

如果我们将物理屏幕的分辨率加倍，总像素数将增加四倍：双倍的水平像素数乘以双倍的垂直像素数。 因此，如果是“2x”的屏幕，所需的像素数不只是加倍，而是增加到原来的四倍！

那么，这有何实际意义呢？我们可以通过高分辨率屏幕提供绚丽的图像，这可以作为产品的一大特色。 不过，高分辨率屏幕也需要高分辨率图像：尽可能优先使用矢量图像，因为这类图像与分辨率无关，并且能够始终提供清晰的效果，而如果需要使用光栅图像，请借助 `srcset` 和 `picture` (https://developers.google.com/web/fundamentals/design-and-ux/responsive/images?hl=zh-cn#images-in-markup)
提供并优化每个图像的多个变体。

优化矢量图像

TL;DR

- SVG 是一种基于 XML 的图像格式
- SVG 文件应进行缩减，以减小其大小
- SVG 文件应使用 GZIP 进行压缩

所有现代浏览器都支持可缩放矢量图形 (SVG)，这种基于 XML 的图像格式适用于二维图形：我们可以将 SVG 标记直接嵌入网页，也可将其作为外部资源嵌入网页。 然后，可通过大多数基于矢量的绘图软件创建一个 SVG 文件，或直接在您喜欢的文本编辑器中手动创建。

```
<?xml version="1.0" encoding="utf-8"?>
<!-- Generator:Adobe Illustrator 17.1.0, SVG Export Plug-In . SVG Version:6.0
<svg version="1.2" baseProfile="tiny" id="Layer_1" xmlns="http://www.w3.org/2
   x="0px" y="0px" viewBox="0 0 612 792" xml:space="preserve">
<g id="XMLID_1_">
  <g>
    <circle fill="red" stroke="black" stroke-width="2" stroke-miterlimit="10"
  </g>
</g>
</svg>
```

上例渲染的是一个具有黑色轮廓和红色背景的简单圆形，并且是从 Adobe Illustrator 导出。您可以看出，它包含大量元数据，例如图层信息、注解和 XML 命名空间，而在浏览器中渲染资产时通常不需要这些数据。 因此，通过 svgo (https://github.com/svg/svgo) 之类的工具将您的 SVG 文件缩小绝对有益。

举个有说服力的例子，svgo 能够将上面这个由 Illustrator 生成的 SVG 文件的大小减少 58%，使其从 470 个字节缩小到 199 个字节。 而且，由于 SVG 是一种基于 XML 的格式，因此我们还可以应用 GZIP 压缩来减小其传送大小 - 确保将您的服务器配置为对 SVG 资产进行压缩！

优化光栅图像

TL;DR

- 光栅图像是像素栅格
- 每个像素都编码了颜色和透明度信息
- 图像压缩程序使用各种方法来减少每个像素所需的位数，以减小图像的文件大小

光栅图像就是一个 2 维“像素”栅格，例如，100x100 像素的图像是 10,000 个像素的序列，而每个像素又存储有“RGBA (https://en.wikipedia.org/wiki/RGBA_color_space)”值：(R) 红色通道、(G) 绿色通道、(B) 蓝色通道和 (A) alpha（透明度）通道。

在内部，浏览器为每个通道分配 256 个值（色阶），也就是每个通道 8 位 (2 ^ 8 = 256)，每个像素 4 个字节（4 个通道 x 8 位 = 32 位 = 4 个字节）。 因此，如果我们知道栅格尺寸，就能轻易计算出文件大小：

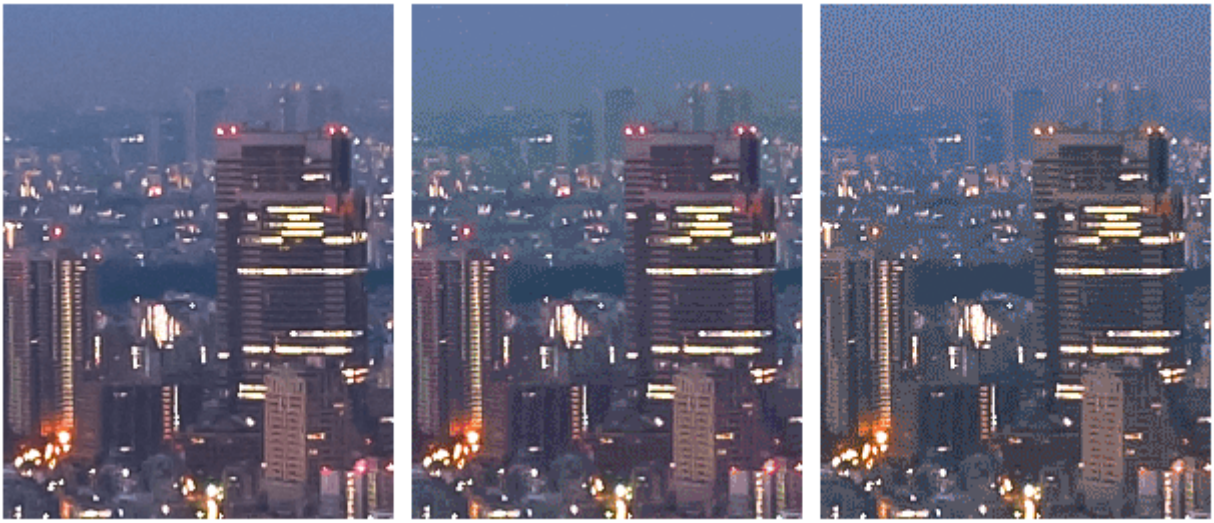
- 100 x 100 像素的图像由 10,000 个像素组成
- 10,000 个像素 x 4 个字节 = 40,000 个字节
- 40,000 个字节/1024 = 39 KB

注：顺便提一句，无论从服务器向客户端传送数据时使用哪一种图片格式，在浏览器对图像进行解码时，每个像素始终占用 4 个字节的内存。 对于大型图像和可用内存不充裕的设备（例如低端移动设备），这可能成为一个重要的约束条件。

尺寸	像素	文件大小
100 x 100	10,000	39 KB
200 x 200	40,000	156 KB
300 x 300	90,000	351 KB
500 x 500	250,000	977 KB
800 x 800	640,000	2500 KB

100x100 像素图像的文件大小只有 39KB，可能似乎不是什么大问题，但对于更大的图像，文件大小会迅速暴增，并使图像资产的下载既速度缓慢又开销巨大。 幸运的是，目前为止我们所描述是“未压缩”图片格式。 我们可以采取什么措施来减小图片文件的大小呢?

一个简单的策略是将图像的“位深”从每个通道 8 位减少为更小的调色板：每个通道 8 位为每个通道提供 256 个值，共计提供 16777216 (256 ^ 3) 种颜色。 如果我们将调色板减少为 256 色，会出现什么情况？那样的话，RGB 通道一共只需要 8 位，每个像素立即可以节约两个字节，与原来每个像素 4 个字节的格式相比，通过压缩节约了 50% 的字节！



注：从左至右 (PNG)：32 位（16M 色）、7 位（128 色）、5 位（32 色）。 包含渐变色过渡的复杂场景（渐变、天空等）需要较大的调色板，以避免在 5 位资产中产生马赛克天空之类的视觉伪影。 另一方面，如果图像只使用几种颜色，较大的调色板只会浪费宝贵的位数！

接下来，在优化了各个像素中存储的数据之后，我们可以多动动脑筋，看看能不能对相邻像素也做做优化：其实，许多图像（尤其是照片）的大量相邻像素都具有相似的颜色 - 例如，天空、重复的纹理等。 压缩程序可以利用这些信息采用“增量编码 (https://en.wikipedia.org/wiki/Delta_encoding)”，在这种编码方式下，并不为每个像素单独存储值，而是存储相邻像素之间的差异：如果相邻像素相同，则增量为“零”，我们只需存储一位！但是这显然还不够...

人眼对不同颜色的敏感度不同：为此，我们可以通过减小或增大这些颜色的调色板来优化颜色编码。 “相邻”像素构成二维栅格，这意味着每个像素都有多个相邻像素：我们可以利用这一点进一步改进增量编码。 我们不再只是关注每个像素直接相邻的像素，而是着眼于更大块的相邻像素，并使用不同设置对不同的像素块进行编码。 当然还不止这些...

您可以看出，图像优化很快就会复杂（或者有趣起来，全凭您怎么看了），这也是学术和商业研究都很活跃的一个领域。 由于图像占据了大量字节，因此开发更好的图像压缩方法具有极大价值！如果您很想了解更多信息，请访问 Wikipedia 网页 (https://en.wikipedia.org/wiki/Image_compression)，或查看 WebP 压缩方法白皮书 (https://developers.google.com/speed/webp/docs/compression?hl=zh-cn)中提供的实例。

所以说，还是那句话，这一领域极具价值，但学术性也很强：那么它如何帮助我们优化网页上的图像呢？我们当然没有能力发明新的压缩方法，但一定要了解问题的基本概念：RGBA 像素、位深和各种优化方法。 我们一定要了解并牢记上述所有概念，才能深入讨论不同的光栅图像格式。

无损图像压缩与有损图像压缩

TL;DR

- 由于人眼的工作方式的缘故，图像是进行有损压缩的不错的选择
- 图像优化依赖有损和无损压缩来实现
- 图像格式上的差异是由于优化图像时使用的有损和无损算法的差异和使用方式的差异所致
- 并不存在任何适用于所有图像的最佳格式或“质量设置”：每个特定压缩程序与图像内容的组合都会产生独特的输出

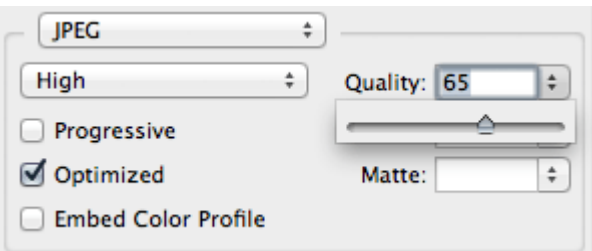
对于某些类型的数据（例如网页的源代码或可执行文件），至关重要的是，压缩程序不能改动或丢失任何原始信息：即使只有一位数据丢失或出错，也可能完全改变文件内容的含义，或者更糟，将其完全破坏。 对于某些其他类型的数据（例如图像、音频和视频），提供原始数据的“近似”表示可能完全能够接受。

实际上，由于人眼工作方式的缘故，我们往往可以偷个懒，通过舍弃每个像素的某些信息来减小图像的文件大小。例如，人眼对不同颜色的敏感度不同，这意味着我们可以使用较少的位数来编码某些颜色。 因此，典型的图像优化过程由两个高级步骤组成：

1. 使用“有损 (https://en.wikipedia.org/wiki/Lossy_compression)”过滤器处理图像，去除某些像素数据
2. 使用“无损 (https://en.wikipedia.org/wiki/Lossless_compression)”过滤器处理图像，对像素数据进行压缩

第一步是可选步骤，具体算法将取决于特定的图像格式，但一定要了解，任何图像都可通过有损压缩步骤来减小其大小。实际上，不同图像格式（例如 GIF、PNG、JPEG 以及其他格式）之间的差异在于它们在执行有损和无损压缩步骤时所使用（或省略）特定算法的组合。

那么，有损和无损优化的“最佳”配置是什么？这个问题的答案取决于图像内容以及您自己的标准（例如在文件大小与有损压缩带来的伪影之间的权衡）：在某些情况下，您可能想跳过有损优化，完全真实地传递复杂的细节，但在其他情况下，也许可以采用激进的有损优化来减小图像资产的文件大小。 这取决于您自己的判断和环境，并不存在任何通用的设置。



举一个实例，在使用有损格式（例如 JPEG）时，压缩程序通常提供可自定义的“质量”设置（例如，Adobe Photoshop 中“Save for Web”功能提供的质量滑块），该设置一般是一个 1-100 的数字，用于控制特定有损和无损算法集合的内部工作。 为获得最佳效果，请为您的图像试验不同的质量设置，不要害怕调低质量，调低后的视觉效果通常很不错，并且文件大小的缩减程度可能相当大。

注：请注意，由于对图像进行编码所使用的算法不同，不同图片格式的质量级别无法直接进行比较：质量级别为 90 的 JPEG 与质量级别为 90 的 WebP 产生的效果截然不同。 实际上，即使是同一图像格式质量级别，也可能因压缩程序实现方法不同而产生明显不同的效果！

选择正确的图像格式

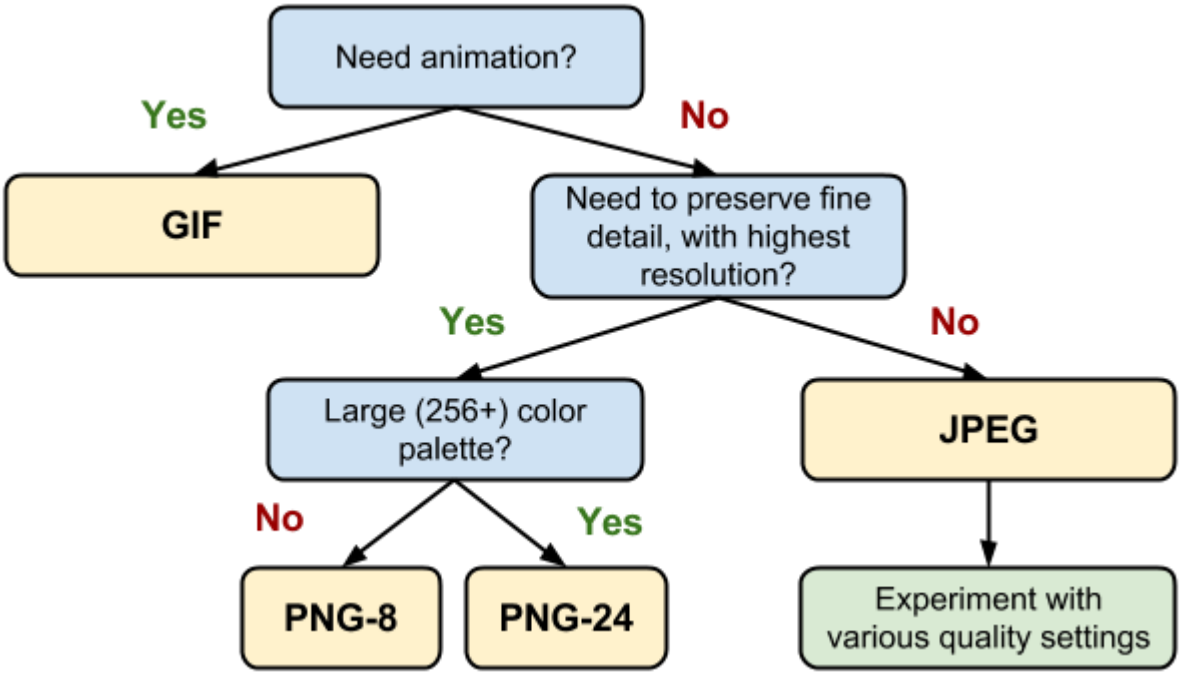
TL;DR

- 首先选择正确的通用格式：GIF、PNG、JPEG
- 通过试验选出每一种格式的最佳设置：质量、调色板大小等
- 考虑为现代化客户端添加 WebP 和 JPEG XR 资产

除了不同的有损和无损压缩算法外，不同的图像格式还支持不同的功能，例如动画和透明度 (alpha) 通道。 因此，需要将所需视觉效果与功能要求相结合来为特定图像选择“正确的格式”。

格式	透明度	动画	浏览器
GIF (http://en.wikipedia.org/wiki/Graphics_Interchange_Format)	支持	支持	所有
PNG (http://en.wikipedia.org/wiki/Portable_Network_Graphics)	支持	不支持	所有
JPEG (http://en.wikipedia.org/wiki/JPEG)	不支持	不支持	所有
JPEG XR (http://en.wikipedia.org/wiki/JPEG_XR)	支持	支持	IE
WebP (http://en.wikipedia.org/wiki/WebP)	支持	支持	Chrome、Opera、Android

获得普遍支持的图片格式有三种：GIF、PNG 和 JPEG。 除了这些格式外，一些浏览器还支持较新的格式（例如 WebP 和 JPEG XR），它们的总体压缩率更高，提供的功能也更多。 那么，您应该使用哪一种格式呢?



1. **您是否需要动画？如果需要，GIF 是唯一的通用选择。**
    - GIF 将调色板限制为最多 256 色，这对大多数图像而言都不是好的选择。 况且，对于调色板较小的图像，PNG-8 的压缩效果更佳。 因此，只有需要动画时，GIF 才是正确的选择。
2. **您是否需要使用最高分辨率保留精细的细节？请使用 PNG。**
    - 除了选择调色板的大小外，PNG 不采用任何有损压缩算法。 因此，它能生成最高质量的图像，但代价是文件大小要比其他格式大得多。 请谨慎使用。
    - 如果图像资产包含由几何形状组成的图像，请务必考虑将其转换成矢量 (SVG) 格式！
    - 如果图像资产包含文本，请停下来再做考虑。 图像中的文本无法选择、搜索或“缩放”。 如果您需要表现一种自定义外观（出于品牌推广或其他原因），请改用网页字体。
3. **您是否要优化照片、屏幕截图或类似的图像资产？请使用 JPEG。**
    - JPEG 组合使用有损和无损优化来减小图像资产的文件大小。 请尝试几种 JPEG 质量级别，为您的资产找到最佳的质量 与文件大小平衡点。

最后，为每一项资产确定了最佳图像格式及其设置之后，请考虑增加一个以 WebP 和 JPEG XR 格式编码的变体。 这两种格式均为新格式，并且遗憾的是，它们没有（尚未）得到所有浏览器的普遍支持，但尽管如此，这两种格式仍可为较新的客户端显著降低文件大小，例如，平均来说，与可比的 JPEG 图像相比，WebP 可将文件大小减小 30% (https://developers.google.com/speed/webp/docs/webp_study?hl=zh-cn)。

由于 WebP 和 JPEG XR 均未得到普遍支持，您需要向应用或服务器添加额外的逻辑来提供相应的资源：

- 有些 CDN 将图像优化作为一项服务提供，包括提供 JPEG XR 和 WebP。
- 有些开源工具（例如 PageSpeed for Apache 或 PageSpeed for Nginx）会自动化优化、转换和提供相应资产。
- 您可以添加额外的应用逻辑来检测客户端，检查客户端支持的格式，并提供最合适的图像格式。

最后请注意，如果您使用 Webview 在本机应用中渲染内容，就可以完全控制客户端，并可独占使用 WebP！Facebook、Google+ 以及许多其他应用都使用 WebP 来提供其应用内的所有图像 - 实现的文件大小缩减定然物有所值。 如需了解有关 WebP 的更多信息，请观看 Google I/O 2013 上的演讲 WebP：部署更快速、更小并且更绚丽的图像 (https://www.youtube.com/watch?v=pS8udLMOOaE&hl=zh-cn)。

工具和参数调优

没有任何一种图像格式、工具或优化参数集完美到适用于所有图像。 为获得最佳效果，您需要根据图像的内容及其视觉以及其他技术要求来挑选格式及其设置。

工具	说明
gifsicle (http://www.lcdf.org/gifsicle/)	创建和优化 GIF 图像
jpegtran (http://jpegclub.org/jpegtran/)	优化 JPEG 图像
optipng (http://optipng.sourceforge.net/)	无损 PNG 优化
pngquant (http://pngquant.org/)	有损 PNG 优化

不要害怕试验各压缩程序的参数。 调低质量，看看效果如何，然后取消重来。 找到一组合适的设置后，您可以对网站上的其他类似图像应用这组设置，但不要认为所有图像都必须使用相同的设置进行压缩。

提供缩放的图像资产

TL;DR

- 提供缩放的资产是最简单并且最有效的优化之一

- 密切关注较大的资产，因为它们会产生大量开销
- 通过将图像缩放到其显示尺寸，减少多余的像素数

图像优化可归结为两个标准：优化编码每个图像像素所使用的字节数，和优化总像素数：图像的文件大小就是总像素数与编码每个像素所使用字节数的乘积。 不多不少。

因此，最简单也是最有效的一种图像优化方法就是，确保我们提供的像素数恰好是在浏览器中按预期尺寸显示资产所需的像素数。 听起来很简单，不是吗？遗憾的是，大多数网页的许多图像资产都做不到这一点：它们提供的资产通常较大，需要依赖浏览器对其进行重新缩放（这还会占用额外的 CPU 资源）并以较低分辨率显示。

注：在 Chrome DevTools 中将光标悬停在图像元素上，可同时显示该图像资产的“自然”尺寸和“显示”尺寸。 在上例中，下载的是 300x260 像素图像，但随后在客户端上显示时，尺寸缩小为 245x212。

提供多余像素的开销只会让浏览器代替我们重新缩放图像，减少并优化渲染网页所需总字节数的大好机会因此被错过。 还要注意的是，尺寸调整不仅受图像缩减像素数的影响，还受其自然尺寸的影响。

屏幕分辨率	自然尺寸	显示尺寸（CSS 像素）	多余的像素
1x	110 x 110	100 x 100	110 x 110 - 100 x 100 = 2100
1x	410 x 410	400 x 400	410 x 410 - 400 x 400 = 8100
1x	810 x 810	800 x 800	810 x 810 - 800 x 800 = 16100
2x	220 x 220	100 x 100	220 x 220 - (2 x 100) x (2 x 100) = 8400
2x	820 x 820	400 x 400	820 x 820 - (2 x 400) x (2 x 400) = 32400
2x	1620 x 1620	800 x 800	1620 x 1620 - (2 x 800) x (2 x 800) = 64400

请注意，在上述所有情况下，显示尺寸只比各屏幕分辨率所需资产“小 10 个 CSS 像素”。 不过，多余像素数及其相关开销会随图像显示尺寸的增加而迅速上升！因此，尽管您可能无法保证以精确的显示尺寸提供每一个资产，**但您应该确保尽可能减少多余的像素数，且特别是大型资产以尽可能接近其显示尺寸的尺寸提供。**

图像优化检查清单

图像优化既是一门艺术，也是一门科学：说它是一门艺术，是因为单个图像的压缩并不存在明确的最佳方案，说它是一门科学，则是因为有一些发展成熟的方法和算法有助于显著缩减图像的大小。

在您努力优化图像时，要记住以下这些技巧和方法：

- **首选矢量格式：**矢量图像与分辨率和缩放无关，这使其成为多设备和高分辨率情况的完美选择。
- **缩小和压缩 SVG 资产：**大多数绘图应用生成的 XML 标记往往包含可以移除的多余元数据；确保您的服务器配置为对 SVG 资产采用 GZIP 压缩。
- **挑选最佳光栅图像格式：**确定您的功能要求并选择适合每个特定资产的格式。
- **通过试验为光栅格式找到最佳质量设置：**不要害怕调低“质量”设置，调低后的效果通常很不错，并且字节数的缩减很显著。
- **移除多余的图像元数据：**许多光栅图像都包含多余的资产元数据：地理信息、相机信息等。 请使用合适的工具删除这些数据。
- **提供缩放的图像：**调整服务器上的图像尺寸，并确保图像的“显示”尺寸尽可能接近其“自然”尺寸。 尤其要密切注意大型图像，因为在调整尺寸时，这类图像占用的开销最大！
- **自动化、自动化、自动化：**投资购置自动化工具和基础设施，这样可以确保您的所有图像资产始终得到优化。

反馈

Was this page helpful?

Yes

No

上一页
← Optimizing Encoding and Transfer Size of Text-based Assets
(https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/optimize-encoding-and-transfer?hl=zh-cn)

下一页
Automating Image Optimization →
(https://developers.google.com/web/fundamentals/performance/optimizing-content-efficiency/automating-image-optimization/?hl=zh-cn)

Except as otherwise noted, the content of this page is licensed under the Creative Commons Attribution 3.0 License (https://creativecommons.org/licenses/by/3.0/), and code samples are licensed under the Apache 2.0 License (https://www.apache.org/licenses/LICENSE-2.0). For details, see our Site Policies (https://developers.google.com/terms/site-policies?hl=zh-cn). Java is a registered trademark of Oracle and/or its affiliates.

上次更新日期：二月 27, 2019

Chromium Blog
The latest news on the Chromium blog.

GitHub
Fork our code samples and other open-source projects.

Twitter
Connect with @ChromiumDev on Twitter.

Videos
Check out our videos.