前端开发中，对图片的优化技巧有哪些？

按照先后顺序有以下：

1. **去掉无意义的修饰。**嗯，我会瞎说吗？除了内容图片，其他的图片的作用是修饰，也就是对于传达信息来说并**非本质性**的。最大的优化就是压根不要图片！所以在优化之前要做的，首先是确认设计，设计本身是否需要用那么多图片？还是说可以更简洁？

2. **不用图片。**嗯，切图是一件扯淡的事情！不要隔靴搔痒了少年，直接**使用CSS替代图片**来实现修饰效果吧！如半透明、边框、圆角、阴影、渐变等，在当前主流浏览器中都可以用CSS达成。将来CSS滤镜得到广泛支持后，还可以做到alpha混合、正片叠底等各种效果。

3. **使用矢量图替代位图。**对于绝大多数图案、图标等，矢量图更小，且可缩放而无需生成多套图。现在主流浏览器都支持SVG了，所以可放心使用！

4. **使用恰当的图片格式。**我们常见的图片格式有JPEG、GIF、PNG。基本上，内容图片多为照片之类的，适用于JPEG。而修饰图片通常更适合用**无损压缩的PNG**。而GIF基本上**除了GIF动画外不要使用**。且动画的话，也更建议用video元素和视频格式，或用SVG动画取代。除了这些格式之外，Chrome、新版Opera、Android 4+支持WebP格式，IE 9+、IE mobile 10+支持JPEG XR。这两个新格式都支持无损和有损压缩，都具有更良好的压缩比。当然这需要为不同的浏览器返回不同的图片，增加了开发成本，也增加存储成本。不过你省了流量或者相同流量下改善了图片质量，提升了用户体验。你会如何取舍呢？对了，别忘了**使用优秀的图片编码器及合适的参数**。好的图片编码器，尤其是有损图片格式的编码器，能通过算法或手动调整，获得更高的压缩比。

以下是普遍适用各种资源而不限于图片的优化手段：

5. **使用data url。**资源内嵌于CSS或HTML中，而不必单独请求。注意，多个地方都要使用的资源不一定适合用此优化方式，因为图片数据重复多了，增加流量。另外许多浏览器对data url有长度限制，注意资源的大小。

6. **按照HTTP协议设置合理的缓存。**具体的缓存策略（如永久缓存+重命名）、部署策略（如反向代理、CDN等）这里就不展开了。

7. **使用支持SPDY的服务器。**SPDY可认为是未来的HTTP 2.0的早期实现，Chrome、Firefox 13+、Opera 12+、IE 11+均已支持SPDY。SPDY和HTTP2可参考此中文演讲：[http://www.youtube.com/watch?v=r74RAcrc1ZA](//link.zhihu.com/?target=http%3A//www.youtube.com/watch%3Fv%3Dr74RAcrc1ZA" \t "_blank)（请自备梯子），这里就不展开了。

8. **资源的lazyload或postpone。**（lazyload：延迟到其他资源下载完成后再加载，postpone：延迟到元素可见再加载。）目前基本上都要用脚本控制。未来HTML和CSS会增加相关的控制属性，见：[Resource Priorities](//link.zhihu.com/?target=https%3A//dvcs.w3.org/hg/webperf/raw-file/tip/specs/ResourcePriorities/Overview.html" \t "_blank)。

9. **资源的prefetch。**可用<link rel=prefetch>，见[http://www.whatwg.org/specs/web-apps/current-work/#link-type-prefetch](//link.zhihu.com/?target=http%3A//www.whatwg.org/specs/web-apps/current-work/%23link-type-prefetch" \t "_blank)。注意prefetch只是**hint**，Firefox会预取资源（如果网络空闲的话），而IE 9则是对该资源的hostname进行DNS预解析。如果你真的需要更强的控制，则得用脚本。注意：Chrome支持与prefetch相近但更进一步的<link rel=prerender>，另外SPDY加入了与prefetch相近但语义不同的subresource link支持，这两个新特性我也没用过，有兴趣的可以尝试。

图片的其他优化技巧如字体图标、CSS Sprites等，不过我不推荐。用字体图标不如用SVG。使用了SPDY和data url后，CSS Sprites完全没有必要用了。

再有各种特定的图片问题，超出了一般优化的范畴。如许多手机浏览器有黑夜模式，其中有的浏览器允许定制黑夜模式；有的手机浏览器允许在用户开启不加载图片选项的情况下让开发者设置必须加载的图片（有点绕）；又如许多手机浏览器有所谓云加速模式，即在服务器端对图片进行处理后再发送给客户端，应该返回怎样的图片给这些服务器有待研究和实践。

10. 最后是**responsive设计**所需的图片优化，可能要产生多套不同大小和分辨率的图片，配合media query、以及srcset属性、picture元素、src-N等标准提案，这个话题比较大，尚未形成普遍认可的最佳实践，这里也不多展开了。

以上。

补充一个 mxhr，比较经典的自定义协议的实现，应用场景在 大量细小图片加载的地方，比如照片墙或者瀑布流，尤其手机端，如果做的细致一些，我个人觉得还是比较屌的。。

原理很简单，xhr加载base64的图片，自定义分隔符，利用xhr对象状态为3的这个值，对文本内容做解析，再扔到页面里，不仅仅用于加载图片，你也可以拿它来加载js，html，css等资源。。

这个概念来源于 邮箱附件实现思路。

补充一点，服务器开启gzip压缩。

补充一下两点：

1. 提升图片的 DNS 解析时间不限于 prefetch，别看 DNS 时间不长但它的影响是全部的性能指标。
2. 使用 data URI 需要十分小心，如果 CSS 还是放在 head 内，文件的增大必然导致性能（开始渲染时间、首屏）变差。