静态资源

图片

JPEG：
不适合：不支持透明度。
非常适合：颜色丰富的照片、彩色图大焦点图、通栏 banner 图；
PNG：
不适合：由于是无损存储，彩色图像体积太大，所以不太适合。
非常适合：纯色、透明、线条绘图，图标；边缘清晰、有大块相同颜色区域；颜色数较少但需要半透明。Icon
GIF：
不适合：每个像素只有 8 比特，不适合存储彩色图片。
非常适合：动画，图标。
Webp：
不适合：最多处理 256 色，不适合于彩色图片。
非常适合：适用于图形和半透明图像。

响应式图片

- CSS 媒体查询

```
@media screen and (max-width:640px) {
 my_image{ width:640px; }
}
```

- img 标签属性

```
<img srcset="img-320w.jpg, img-640w.jpg 2x, img-960w.jpg 3x"
src="img-960w.jpg" alt="img">  (x 描述符：表示图像的设备像素比)
```

设备像素比：https://www.zhangxinxu.com/wordpress/2012/08/window-devicepixelratio/
普通屏幕使用 img-320w.jpg，为高分屏使用 img-640w.jpg，如果更高的分辨率则使用 img-960w

懒加载

- IntersectionObsrever
- 滚动判断图片有没有出现在视窗

HTML 优化细则

- 减少 DOM 节点数：:before :after
- 删除 http 或者 https，如果URL的协议头和当前页面的协议头一致的，或者此URL 在多个协议头都是可用的，则可以考虑删除协议

```
<img src="https://abc.com" />
<img src="//baidu.com" />
```

- 语义化标签：

```
aside header footer section
```

文件放在合适位置

- CSS 样式文件链接尽量放在页面头部

CSS 加载不会阻塞 DOM tree 解析（解析DOM 树是同时进行的），但是会阻塞 DOM Tree 渲染（渲染时候需要CSSom），也会阻塞后面 J行（js 执行也需要 css）。

```
dom.getBoundingclientreact()
```

- JS 引用放在 HTML 底部

防止 JS 的加载、解析、执行对阻塞页面后续元素的正常渲染。

CSS 优化细则

- 尽量减少样式层级数

如div ul li span i {color: blue;}, 给 i 加个类名

- 使用外链的 CSS：CDN 缓存
- 尽量避免使用 @import：缺点

```
// a.css
// b.css
@import('./b.css');
// link html
// @import  css 语法
```

- 动画优先使用css 动画

JavaScript 优化细则

语言本身优化的原则：切勿提前优化，不是为了优化而优化，逻辑清晰即可。

非要回答的话：

DOM 操作：

- 尽量使用 id 选择器
- 使用事件节流函数
- 使用事件委托

JavaScript 动画优化

- 尽量使用 CSS3 动画
- 合理使用 requestAnimationFrame 动画代替 setTimeout、setInterval

合理使用缓存

- 逻辑层面的缓存
- 使用可缓存的 Ajax：tab1 (data1) tab2(data2)：data1 缓存起来 data2 缓存起来

减少 relayout 和 repaint

Relayout

width，height，display:none/block，dom.offsetHeight（为了数据准确，relayout一次）
What forces layout / reflow：https://gist.github.com/paulirish/5d52fb081b3570c81e3a

Repaint

color，opacity

- 动画使用绝对定位，可以让动画元素脱离文档流，减少对其他元素的影响
- 避免频繁设置样式，最好把新 style 属性设置完成后，进行一次性更改
- 多使用 能直接到 **合成阶段** 的css 属性：transfoem opacity

压缩

- HTML 压缩工具

html-minifier https://www.npmjs.com/package/html-minifier

- CSS 压缩工具

clean-css https://www.npmjs.com/package/clean-css

- JavaScript 压缩工具：

uglify-js https://www.npmjs.com/package/uglify-js

文件缓存

防止用户看不最新内容，可加版本号，hash 值

前端构建工具介绍

- Gulp

功能偏任务：通过流（Stream）来简化多个任务间的配置和输出，配置代码相对较少

一次构建：

1. 打包
2. 资源 移动
3. 发个消息

- Webpack

功能偏打包

- Rollup

treeshaking，scopehosting 等特性，公共库打包（Vue, react）

webpack 打包优化

- 定位体积大的模块

```
webpack-bundle-analyzer: https://www.npmjs.com/package/webpack-bundle-analyzer
```

- 生产模式进行公共依赖，基础包抽离