# 介绍

加载优化，JS 优化，渲染优化，体验优化，防抖，节流。

## 性能优化目的

性能可以分为文件级和代码级两种。文件级的比如文件的加载速度，是和代码无关的；代码级的比如 HTML 嵌套太深会影响渲染，由代码编写影响性能。那么，可以从文件级和代码级两个方面优化性能，最终目的是：如果是处理数据的应用，就应该尽量快；如果是交互类的应用，就应该尽量流畅。

## 发现性能问题

1. [PageSpeed Insights](https://developers.google.cn/speed/pagespeed/insights/)，Google 出的，输入网址在线测试 Web 应用的性能问题，并给出修改建议。
2. 浏览器开发工具，通常会有性能相关的工具，比如 Chrome Devtools 有检查 JavaScript 执行的 Performance，Audits 可以检查整个应用的运行，Network 可以查看资源请求相应情况。
3. [Lighthouse](https://github.com/GoogleChrome/lighthouse)，Chrome 的 Audits 集成的是 Lighthouse，这个工具也可以通过脚本的方式运行生成报告。

# 文件

## CDN

### JS 库

1. [jsDelivr](https://www.jsdelivr.com/)

* <script src="https://cdn.jsdelivr.net/npm/jquery@3.2.1/dist/jquery.min.js"></script>

1. [cdnjs](https://cdnjs.com/)

* <script src="https://cdnjs.cloudflare.com/ajax/libs/jquery/3.3.1/jquery.min.js"></script>

### 字体

1. Google Fonts

* <link href='https://fonts.googleapis.com/css?family=Open+Sans' rel='stylesheet'>

## 图片

### 图片格式的选择

webp, gif, png, jpg, icon font

### icon font

字体图片有两个优点: 矢量图放大后不失真; 起到图片精灵的作用, 减少图片请求次数.

图片转成字体文件, 作为矢量图, 常用于图标. 工具可用 [iconfont](http://www.iconfont.cn/) 上传后生成 CSS 文件和字体.

### 图片延迟加载/懒加载(lazy load)

#### 思路

延迟加载通常是将暂不需要的资源延后加载. 懒加载是延迟加载的一种, 即达到某个条件(或某个事件触发)时加载图片.

延迟加载可处理为, 当必要的资源加载完后再加载其余资源. 懒加载基本思路:

1. 暂存一张图片, 显示该默认图片.
2. 显示图片的元素在可视区域时, 加载该图片.

#### 实例

具体到技术, [飞猪H5](https://h5.m.taobao.com/trip/home/index.html?_projVer=0.1.125) 的实现方法是:

<div class="base-bg base-bg-m regular-product\_\_image\_\_\_Bu73a" data-reactid=".0.$=1$trip\_home\_arbitrary\_gate\_product\_0.0.$=1$regular\_item\_1.0.$=10">  
 <div data-lazyloadid="lazyload\_item\_36" class="fade" style="opacity: 1;background-image: url(&quot;//gw.alicdn.com/tips/i3/638737216/TB2vvwZtVXXXXX0XXXXXXXXXXXX\_!!638737216.jpg\_400x400q75.jpg\_.webp&quot;);"  
 data-reactid=".0.$=1$trip\_home\_arbitrary\_gate\_product\_0.0.$=1$regular\_item\_1.0.$=10.$=11" data-imageloaded="true"></div>  
</div>

1. 父元素上设置默认的背景图片.

* .skin-yellow .base-bg {  
   background: #f2f3f4 url(…mgg7e+vIXHxHbzIMosU7LAtcvNOAUKpxf6kSUl8MPvAnj+AYRcPQeahlKYAAAAAElFTkSuQmCC) 50% no-repeat;  
   background-size: auto .8rem;  
  }

1. 子元素内联样式背景图片链接, 外链样式图片相关属性. 初始化时 opacity: 0, 并且不包含背景设置.

* opacity: 1;  
  background-image: url(//gw.alicdn.com/tips/i3/638737216/TB2vvwZtVXXXXX0XXXXXXXXXXXX\_!!638737216.jpg\_400x400q75.jpg\_.webp);
* .base-bg>div {  
   width: 100%;  
   height: 100%;  
   background-repeat: no-repeat;  
   background-position: 50%;  
   background-size: cover;  
  }

1. **满足条件时**, 设置子元素的背景图片(或者设置 img src 属性), 然后标识已加载的标签一个属性(比如 data-imageloaded="true"), 如果是 img 标签, 加载后删除 data-src.

#### 关键点

这里的**满足条件时**, 可用以下逻辑. 检查元素是否在可视区域, 可全局循环检查, 至于是否有性能问题, 待考察.

loadIfVisible() // 如果在可视区域则加载  
onScroll(loadIfVisible()); // 滚动事件触发时, 检查

判断元素是否在可视区域

// 判断元素是否在可视区域  
function isInView(obj) {  
 var e = obj.getBoundingClientRect();  
 return !(e.top > window.innerHeight || e.bottom < 0 || e.left > window.innerWidth || e.right < 0)  
}

#### 参考扩展

1. [stackoverflow, How to tell if a DOM element is visible in the current viewport?](https://stackoverflow.com/questions/123999/how-to-tell-if-a-dom-element-is-visible-in-the-current-viewport#7557433)
2. [mozilla, Element.getBoundingClientRect()](https://developer.mozilla.org/zh-CN/docs/Web/API/Element/getBoundingClientRect)

# 体验优化

对页面性能的优化算起来都是体验优化, 这里主要指具有进一步提升性质的. 比如, 骨架屏实际上也可以用转圈圈来替代, 但其使用感受更好.

## 骨架屏/Skeleton Screen

骨架屏指的是数据呈现之前, 一般用浅色的色条勾勒渲染后的轮廓. 相对通常的空白区域或者加 loading, 体验会好一些. 其次还起到了占位的作用.

**文章**

1. [Skeleton Screen -- 骨架屏](http://www.bestvist.com/2018/01/19/skeleton-screen/)
2. [How to Speed Up Your UX with Skeleton Screens](https://www.sitepoint.com/how-to-speed-up-your-ux-with-skeleton-screens/)
3. [Building Skeleton Screens with CSS Custom Properties](https://css-tricks.com/building-skeleton-screens-css-custom-properties/)

**实例**

Ant Design 的 loading card, <https://ant.design/components/card/>