

PERFORMANCE

陈磊

2018 年 3 月 29 日

目录

| | | |
|----------|----------------------------------|----------|
| 1 | 文件 | 1 |
| 1.1 | 图片 | 1 |
| 1.1.1 | 图片格式的选择 | 1 |
| 1.1.2 | icon font | 1 |
| 1.1.3 | 图片延迟加载/懒加载 (lazy load) | 1 |
| 2 | 体验优化 | 3 |
| 2.1 | 骨架屏/Skeleton Screen | 3 |

1 文件

1.1 图片

1.1.1 图片格式的选择

webp, gif, png, jpg, icon font

1.1.2 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 上传后生成 CSS 文件和字体.

1.1.3 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 的实现方法是:

```
1 <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">
2   <div data-lazyloadid="lazyload_item_36" class="fade" style="
      ↳ opacity: 1;background-image: url(&quot;//gw.alicdn.com/
      ↳ tips/i3/638737216/TB2vvwZtVXXXXXOXXXXXXXXXXXXX_
      ↳ !!638737216.jpg_400x400q75.jpg_.webp&quot;);"
3     data-reactid=".0.$=1$trip_home_arbitrary_gate_product_0.0.$=1
      ↳ $regular_item_1.0.$=10.$=11" data-imageloaded="true">
      ↳ </div>
4 </div>
```

1. 父元素上设置默认的背景图片.

```
1 .skin-yellow .base-bg {
2   background: #f2f3f4 url(data:image/png;base64,
      ↳ iVBORwOKGgoAAAANSUheEUgAAALkAAABPCAMAAACAUJRqAAAAq1BMV...
      ↳ mgg7e+vIXHxHbzIMosU7LAtcvNOAUKpxf6kSUl8MPvAnj+
      ↳ AYRcPQeahlKYAAAAAE1FTkSuQmCC) 50% no-repeat;
3   background-size: auto .8rem;
4 }
```

2. 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.

```
1 opacity: 1;
2 background-image: url(//gw.alicdn.com/tips/i3/638737216/
  ↳ TB2vvwZtVXXXXXOXXXXXXXXXXXXX_!!638737216.jpg_400x400q75.jpg_.
  ↳ webp);
```

```
1 .base-bg>div {
```

```
2 width: 100%;  
3 height: 100%;  
4 background-repeat: no-repeat;  
5 background-position: 50%;  
6 background-size: cover;  
7 }
```

3. 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.

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

判断元素是否在可视区域

```
1 // 判断元素是否在可视区域  
2 function isInView(obj) {  
3   var e = obj.getBoundingClientRect();  
4   return !(e.top > window.innerHeight || e.bottom < 0 || e.left >  
           ↪ window.innerWidth || e.right < 0)  
5 }
```

参考扩展

1. stackoverflow, How to tell if a DOM element is visible in the current viewport?
2. mozilla, `Element.getBoundingClientRect()`

2 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.

2.1 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
2. How to Speed Up Your UX with Skeleton Screens
3. Building Skeleton Screens with CSS Custom Properties

实例

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