

图片懒加载实现

图片的加载是比较耗费性能的，如果图片没有出现在可视区域内，则可以先不加载，等待图片出现在可视区域了，再加载，以节省流量和提高首屏加载速度。

图片只要 `src` 属性赋值了，就会发起一个 AJAX 请求，如果一开始就给大量的图片赋上正确的地址，则会同时发起大量请求，但是 chrome 只能允许一次最多 6 个请求，所以后面的图片会等待前面的图片请求发出以后再发起，这样还可能造成卡顿。

实现思路：

1. 所有 `` 标签 `src` 属性先用一个尺寸很小的图片来显示，比如 `loading.gif` 这可以保证有一个加载中的动画效果。
2. 给 `` 标签自定义一个属性，比如 `data-src`，用来存放真实的图片地址。
3. 当图片出现在可视区域了就将图片的真实地址赋值给 `src` 属性（获取 `data-src` 的值，再赋值给 `src` 属性）。
4. 监听页面滚动事件，来判断图片是否出现在了可视区域内。

```
<style>
  .animal {
    border: 1px solid #999;
    border-radius: 4px;
    margin-bottom: 32px;
  }
  .animal p {
    margin-bottom: 16px;
  }
  .animal img {
    display: block;
    width: 300px;
    height: 200px;
    margin-right: auto;
    margin-left: auto;
  }
</style>

<div class="container">
  <div class="animal">
    <p>斑马</p>
    
  </div>
  <div class="animal">
    <p>海豚</p>
    
</div>
<div class="animal">
    <p>老虎</p>
    
</div>
<div class="animal">
    <p>狮子</p>
    
</div>
<div class="animal">
    <p>犀牛</p>
    
</div>
<div class="animal">
    <p>熊</p>
    
</div>
<div class="animal">
    <p>熊猫</p>
    
</div>
</div>

<script src="https://cdn.bootcdn.net/ajax/libs/lodash.js/4.17.21/lodash.min.js">
</script>
<script>
    loadImg(); // 初始化时执行一次，让初始阶段就在可视区域内的图片显示出来

    // 在滚动中使用节流措施，避免造成性能开销大
    window.addEventListener(
        "scroll",
```

```
_.throttle(() => {
  loadImg();
}, 200)
);

function loadImg() {
  const images = document.querySelectorAll("img[data-src]");
  if (!images.length) return;
  images.forEach((image) => {
    const rect = image.getBoundingClientRect();
    if (rect.top < window.innerHeight) {
      console.log("image", image.src);
      image.src = image.dataset.src;

      // 给 src 赋值后, 移除 img 标签的 data-src 属性, 上面的获取 img[data-src]
      // 集合时就获取不到已经被赋值过个图片了, 避免多余重复获取
      image.removeAttribute("data-src");
    }
  });
}
</script>
```

`Element.getBoundingClientRect()` 方法返回一个 `DOMRect` 对象, 其提供了元素的大小及其相对于视口的位置。

为了避免滚动页面造成性能开销大, 滚动时需要采用节流措施。