目录

23 让加载“懒”一点

更新时间：2019-09-05 10:16:57

> 人生的价值，并不是用时间，而是用深度去衡量的。
>
> ——列夫·托尔斯泰

引入

不知道大家是否还记得我们在图片优化部分，讲了图片懒加载的相关知识点，这一节我们介绍的则是资源的懒加载，既会对图片懒加载做一些小的补充，也会讲到其他资源的懒加载方案。之所以这一节依旧要讲懒加载，是因为结合上一节所讲的防抖和节流，我们可以对前面介绍的懒加载重新进行更好的优化，废话不多说，开始这一小结的内容。

结合throttle优化图片懒加载

这里其实就是把我们前面2小节内容做一个整合。在图片优化章节我们介绍图片懒加载的时候，需要不断监听的scroll事件，然后判断图片是否已经在首屏页面当中，如果已经在首屏就进行加载，如果没有则无需进行拉取。而且通过上一节的学习，我们也知道scroll这类事件会被频繁触发，对性能的影响是非常大的，所以才有了防抖和节流。那么针对这个scroll事件，我们完全可以使用节流函数包一下，让它隔一段时间再去触发，避免多余性能消耗，如下：

```
const imgLazyLoad = throttle(() => console.log('懒加载操作'), 1000)

document.addEventListener('scroll', imgLazyLoad)
```

用我们封装好的throttle去包装好懒加载操作，这样用户在频繁滚动滚动条的时候就不会产生因为频繁触发而带来的性能问题，这也是节流非常典型的一个应用。这里我们两个小节的内容也非常好的串联了起来，大家在平时的学习和工作当中也要注意各种知识点之间的联系。

高级特性Intersection Observer

我们就来揭晓如何使用Intersection Observer来进行图片懒加载。

之前我们使用的方法需要监听scroll事件，上面我们也介绍了可以使用节流来解决这个问题，说明之前的图片懒加载方案是存在一定问题的；而Intersection Observer不需要监听scroll事件，可以做到只要图片元素出现在可视区域内，我们就能进行回调，具体如下：

```
document.addEventListener("DOMContentLoaded", function() {
  var lazyImages = [].slice.call(document.querySelectorAll("img.lazy"));

  if ("IntersectionObserver" in window) {
    let lazyImageObserver = new IntersectionObserver(function(entries, observer)
      entries.forEach(function(entry) {
        if (entry.isIntersecting) {
          let lazyImage = entry.target;
          lazyImage.src = lazyImage.dataset.src;
          lazyImage.srcset = lazyImage.dataset.srcset;
          lazyImage.classList.remove("lazy");
          lazyImageObserver.unobserve(lazyImage);
        }
      });
    });

    lazyImages.forEach(function(lazyImage) {
      lazyImageObserver.observe(lazyImage);
    });
  }
});
```

上面就是使用Intersection Observer完成图片懒加载的方法，该方法唯一的缺点就是兼容性还不是很好，如果需要兼容版本较低的浏览器，则要根据浏览器的版本封装更通用的方法。

延迟加载视频

前面我们就说过图片和视频这类静态资源资源占比最大，并且介绍了图片懒加载的相关方法。与图片一样，视频同样可以延迟加载，来达到优化性能的目的。正常情况下加载视频，我们使用的是 `<video>` 标签，那么对于一些需要由用户自己播放的视频，最好指定 `<video>` 标签的preload属性为none，这样浏览器就不会预加载任何视频数据。为了占用空间，我们使用poster属性为 `<video>` 占位。如下：

```
<video controls preload="none" poster="replace.jpg">
  <source src="main.webm" type="video/webm">
  <source src="main.mp4" type="video/mp4">
</video>
```

使用第三方延迟加载库

除了上面介绍的一些延迟加载方法之外，我们还可以借助一些已经封装好的第三方库，下面我就来介绍一些成熟的第三方库。如下：

- lozad.js 是超轻量级且只使用 Intersection Observer 的库， 因此它的性能极佳，但如果要在旧版本浏览器上使用，则需要配置polyfill。

定图像网址，该库还可以通过许多插件进行扩展，执行延迟各种资源等操作。

- 如果你使用React，可以使用 react-lazyload来进行图片懒加载操作，这个库是React图片懒加载的主流解决方案。

小结

这一节我们主要介绍了一些延迟加载的方法，有的是对前面的补充，有的是新知识。延迟加载虽然能在性能方面发挥积极作用，但是要注意使用场景，如果是首屏页面，那么关于首屏的所有资源都无需使用延迟加载，用了反而会适得其反，所以性能优化方法的使用仍要结合具体业务来分析。

← 22 防抖节流背后那些事儿 24 服务端渲染 →

精选留言 0

欢迎在这里发表留言，作者筛选后可公开显示

目前暂无任何讨论

千学不如一看，千看不如一练