官网首页优化实践

一、网站性能优化工具推荐

PageSpeed Insights

谷歌的PageSpeed Insights ， 它是先分析网页的内容，然后提供关于如何提升网页加载速度的建议。
使用网址：https://developers.google.com/speed/pagespeed/insights/
然后把要分析的网页的网页输进去：

会得出一些分析结果和优化建议：

此网页的速度统计信息

统计信息显示，互联网上速度为中间值的网页需要进行 4 次阻塞渲染的往返，并且需要加载大约 89 项资源 (1.3MB)。 但此页面使用的资源似乎变少了。 PSI 预计，加载此网页需要进行 3 次阻塞渲染的往返，并且需要 89 项资源 (3.9MB)。所需的往返次数和字节数越少，网页速度就越快。

优化建议

优化图片
显示解决问题的方法

使用浏览器缓存
显示解决问题的方法

清除首屏内容中阻止呈现的 JavaScript 和 CSS
显示解决问题的方法

按优先级排列可见内容

二、优化方案

原版官网首页一共184个请求，其中有125个图片，首页资源有15.3MB。所以减少请求的数据量和优化图片是必须的；再依据pageSpeed分析结果进行相应的优化。
方案大体如下：
一、减少请求数量

- 利用webpack将小于10kb的图片编译成Base64字符串写入CSS文件；svg格式的图片使用字体图标IconFont，减少请求数量。
  在nuxt.config.js里添加如下配置，实现小于10kb的图片的编译。

```
webpackConfig.module.rules.push({
    test: /\.(png|jpe?g|gif|svg)$/,
    use: [
        {
            loader: 'url-loader',
            options: {
                limit: 10000, // 10KO
                name: 'img/[name].[hash:7].[ext]'
            }
        },
    ],
});
```

- 合并js和css代码。
- 整理代码，删除无用的图片、代码、引入文件。
- 提取公共样式，避免重复加载

二、优化静态资源

- 静态资源cdn化，通过webpack插件直接将js、css、图片等静态资源打包在一起，上传cdn，可以明显加快这部分资源的加载速度
- 修改KrImg组件，只下载处理后的阿里云图片，不下载原图。（修改bug）；
- 动态获取的阿里云图片地址设置参数已达到减少图片大小。阿里云图片处理访问规则

三、优化渲染路径

- css文件写在头部，javascript放在尾部。避免js将阻塞解析dom，导致dom绘制延后，出现白屏或者样式错乱。
- 首页设置第一屏下面的内容按需加载和延迟加载，以减少首屏请求数，提升首屏加载速度。
- 整理官网埋点和统计信息，抽离出单独文件，埋点部分设置异步延迟加载。
- 整理首页head内的信息。针对nuxt框架问题：nuxt使用了prefetch来预取下一页的资源文件,打开首页时会load很多其他页面的js和css。

```
render: {
    resourceHints: false
}
extractCSS:true
```

四、 缓存

- 静态资源增加浏览器强缓存和协商缓存。
  - 1.强缓存：不会向服务器发送请求，直接从缓存中读取资源，在chrome控制台的network选项中可以看到该请求返回200的状态码，并且size显示from disk cache或from memory cache；
  - 2.协商缓存：向服务器发送请求，服务器会根据这个请求的request header的一些参数来判断是否命中协商缓存，如果命中，则返回304状态码并带上新的response header通知浏览器从缓存中读取资源；
  - 两者的共同点是，都是从客户端缓存中读取资源；区别是强缓存不会发请求，协商缓存会发请求。

	获取资源形式	状态码	发送请求到服务器
强缓存	从缓存取	200（from cache）	否，直接从缓存取
协商缓存	从缓存取	304（not modified）	是，正如其名，通过服务器来告知缓存是否可用

http协商缓存VS强缓存