Vue 项目构建优化

大多数 Vue 项目是基于 VueCLI 搭建的，而 VueCLI 的底层建筑是 webpack。webpack 是现在主流的功能强大的模块化打包工具，在使用 webpack 时，如果不注意性能优化，有非常大的可能会产生性能问题，性能问题主要分为**开发时打包构建速度慢**、**开发调试时的重复性工作**、以及**输出文件质量不高**等，因此性能优化也主要从这些方面来分析。

本文主要针对是在 Vue 项目中关于底层建筑 webpack 优化以及 Vue 本身相关的一些优化事宜。

1. 把 VueCLI 升级到最新稳定版

尤其是 VueCLI 3 之前创建的项目，强烈建议升级到最新版 Vue CLI。

2. 分析打包结果

- 了解打包的内容
- 找出最大的模块是什么
- 查找错误到达的模块
- 优化它!

2.1. vue-cli-service build

> 参考：
>
> - https://cli.vuejs.org/zh/guide/cli-service.html#vue-cli-service-build

```
用法：vue-cli-service build [options] [entry|pattern]

选项：

  --mode         指定环境模式 (默认值：production)
  --dest         指定输出目录 (默认值：dist)
  --modern       面向现代浏览器带自动回退地构建应用
  --target       app | lib | wc | wc-async (默认值：app)
  --name         库或 Web Components 模式下的名字 (默认值：package.json 中的 "name" 字段或入口文件名)
  --no-clean     在构建项目之前不清除目标目录
  --report       生成 report.html 以帮助分析包内容
  --report-json 生成 report.json 以帮助分析包内容
  --watch        监听文件变化
```

`vue-cli-service build` 会在 `dist/` 目录产生一个可用于生产环境的包，带有 JS/CSS/HTML 的压缩，和为更好的缓存而做的自动的 vendor chunk splitting。它的 chunk manifest 会内联在 HTML 里。

这里还有一些有用的命令参数：

- `--modern` 使用现代模式构建应用，为现代浏览器交付原生支持的 ES2015 代码，并生成一个兼容老浏览器的包用来自动回退。
- `--target` 允许你将项目中的任何组件以一个库或 Web Components 组件的方式进行构建。更多细节请查阅构建目标。

- `--report` 和 `--report-json` 会根据构建统计生成报告，它会帮助你分析包中包含的模块们的大小。
    - 内部使用的 Webpack Bundle Analyzer 插件。

2.2. 使用图形界面打包

> 自带分析功能

在命令行中执行 `vue ui` 启动 VueCLI 图形界面。

```
vue ui
```

3. Gzip 压缩

网站加载的速度很大程序取决于网站资源文件的大小，减少要传输的文件的大小可以使网站不仅加载更快，而且对于那些宽带是按量计费的用户来说也更友好。

HTTP协议上的GZIP编码是一种用来改进WEB应用程序性能的技术。大流量的WEB站点常常使用GZIP压缩技术来让用户感受更快的速度。这一般是指WWW服务器中安装的一个功能，当有人来访问这个服务器中的网站时，服务器中的这个功能就将网页内容压缩后传输到来访的电脑浏览器中显示出来.一般对纯文本内容可压缩到原大小的40%.这样传输就快了，效果就是你点击网址后会很快的显示出来.当然这也会增加服务器的负载. 一般服务器中都安装有这个功能模块的。

Gzip 是什么?

```
一种文件压缩格式。
```

开启 Gzip 有什么好处?

```
Gzip 开启以后会将输出到用户浏览器的数据进行压缩的处理，这样就会减小通过网络传输的数据量，提高文件传输的速度。
```

> 注意：`gzip`不一定适用于所有文件的压缩。例如，文本文件压缩得非常好，通常会缩小两倍以上。另一方面，诸如JPEG或PNG文件之类的图像已经按其性质进行压缩，使用`gzip`压缩很难有好的压缩效果或者甚至没有效果。压缩文件会占用服务器资源，因此最好只压缩那些压缩效果好的文件。
>
> 资源太小的文件也不会压缩。

如何开启?

不同服务器软件配置不一样，具体由部署项目的人负责，一般是运维、后端开发人员，如果想要自行配置，可自行百度查询。大多数服务器软件都是默认开启的。

- Nginx
- Tomcat
- Apache
- IIS
- ...

> 实际工作中，如果服务器软件没有开启，可以和负责运维部署的人员沟通。

如何检测内容是否已开启了 Gzip 压缩？查看响应头中是否有下面的字段信息。

```
Content-Encoding: gzip
```

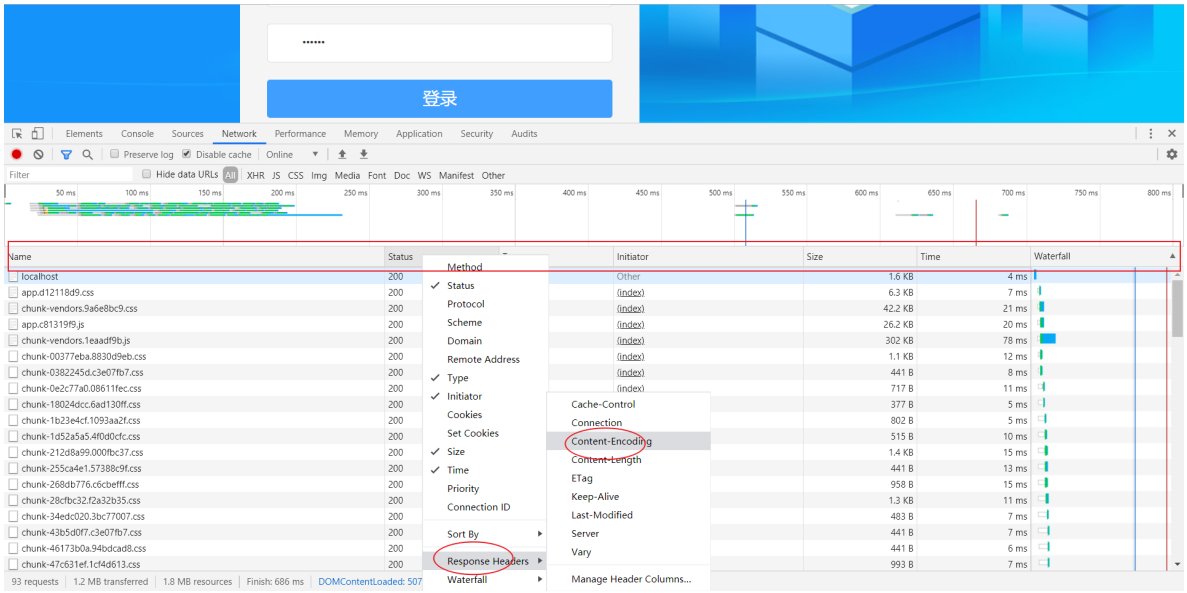
前端没有调整配置服务器软件麻烦，该怎么测试？

使用 VueCLI 官方推荐的 serve 命令行工具。

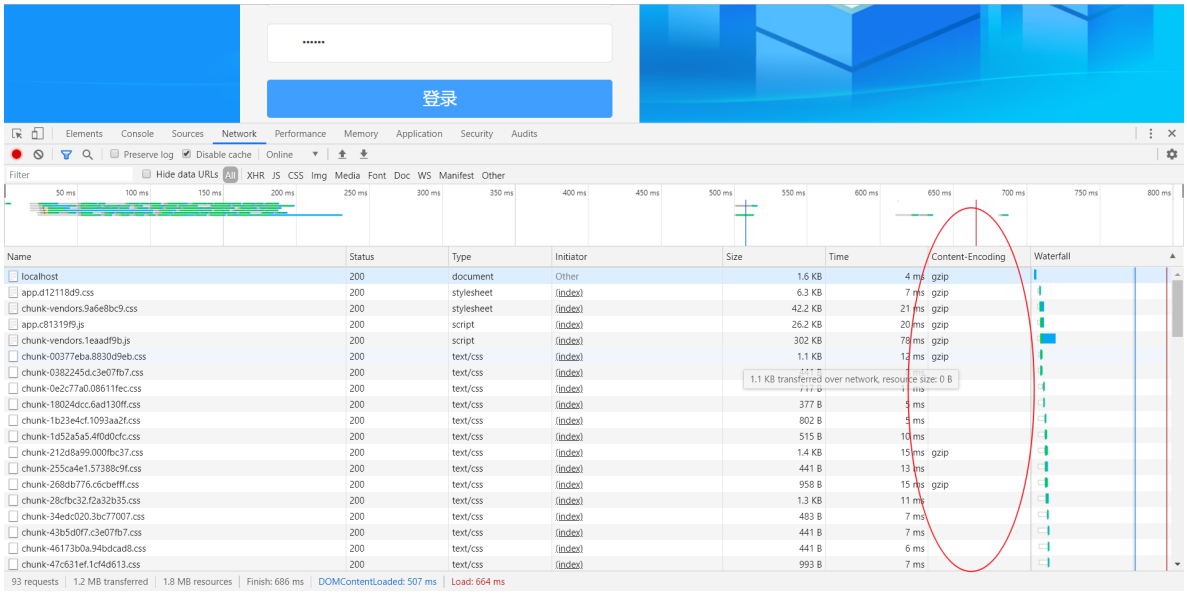
```
# 1、如果你已经安装了就不需要安装了
npm install --global serve

# 使用该命令检测是否已安装或者是否安装成功，如果能看到输出一个版本号，则证明安装好了
serve --version

# 2、在打包的结果目录中执行下面的命令启动一个 HTTP 静态服务（默认开启 Gzip 压缩启动服务）
serve -s ./

# 使用 -u 参数禁用 Gzip 压缩
serve -s -u ./
```

3.1. Gzip 压缩原理

不是每个浏览器都支持gzip的，如何知道客户端是否支持gzip呢，请求头中有个 `Accept-Encoding` 来标识对压缩的支持。客户端http请求头声明浏览器支持的压缩方式，服务端配置启用压缩，压缩的文件类型，压缩方式。当客户端请求到服务端的时候，服务器解析请求头，如果客户端支持gzip压缩，响应时对请求的资源进行压缩并返回给客户端，浏览器按照自己的方式解析，在http响应头，我们可以看到 `Content-encoding: gzip`，这是指服务端使用了 `gzip` 的压缩方式。



> 在红色矩形区域右键选择：Response Headers > Content-Encoding



> 然后就可以观测到所有请求记录的响应内容编码了。

4. 在 Vue CLI 中自定义 webpack 配置

> 参考文档：
>
> - https://cli.vuejs.org/zh/guide/webpack.html

5. CDN

- CDN 主要针对的静态资源
- 我们可以把我们自己的代码放到 CDN 上
- 也可以使用一些免费的第三方 CDN 服务
  - 一般提供的都是一些常用的第三方包的 CDN 服
  - 例如 bootcdn
- 我们最好把项目中比较大的第三方包都使用 CDN 链接
  - 不仅能提高构建的速度
  - 还能提高页面的响应速度
- 当我们在项目中使用 CDN 链接之后，就没必要下载打包第三方包了

6. 路由懒加载

7. 异步组件

和路由懒加载是一个道理。

```
import MyComponent from 'my-component'

new Vue({
  // ...
  components: {
    'my-component': MyComponent
  }
})
```

```
// import MyComponent from 'my-component'
const MyComponent = () => import('my-component')

new Vue({
  // ...
  components: {
    // 'my-component': MyComponent
    'my-component': () => import('my-component')
  }
})
```

8. prefetch

link rel="prefetch" 是一种 resource hint，用来告诉浏览器在页面加载完成后，利用空闲时间提前获取用户未来可能会访问的内容。

```
<!DOCTYPE html>
<html lang="en">
<head>
  <link rel="prefetch" href="./h5.html">
</head>
<body>
  <div>Hello Index page</div>
  <a href="./h5.html">go to H5 Page</a>
</body>
</html>
```

默认情况下，一个 Vue CLI 应用会为所有作为 async chunk 生成的 JavaScript 文件 (通过动态 `import()` 按需 code splitting 的产物) 自动生成 prefetch 提示。

这些提示会被 @vue/preload-webpack-plugin 注入，并且可以通过 `chainWebpack` 的 `config.plugin('prefetch')` 进行修改和删除。

示例：

```
// vue.config.js
module.exports = {
  chainWebpack: config => {
    // 移除 prefetch 插件
    config.plugins.delete('prefetch')

    // 或者
    // 修改它的选项：
    config.plugin('prefetch').tap(options => {
      options[0].fileBlacklist = options[0].fileBlacklist || []
      options[0].fileBlacklist.push(/myasyncRoute(.)+?\.js$/)
      return options
    })
  }
}
```

当 prefetch 插件被禁用时，你可以通过 webpack 的内联注释手动选定要提前获取的代码区块：

```
import(/* webpackPrefetch: true */ './someAsyncComponent.vue')
```

webpack 的运行时会在父级区块被加载之后注入 prefetch 链接。

> 提示
>
> Prefetch 链接将会消耗带宽。如果你的应用很大且有很多 async chunk，而用户主要使用的是对带宽较敏感的移动端，那么你可能需要关掉 prefetch 链接并手动选择要提前获取的代码区块。

9. 按需加载第三方组件

- Vant

10. 缓存和并行处理

VueCLI 内置了：

- `cache-loader` 会默认为 Vue/Babel/TypeScript 编译开启。文件会缓存在 `node_modules/.cache` 中——如果你遇到了编译方面的问题，记得先删掉缓存目录之后再试试看。
- `thread-loader` 会在多核 CPU 的机器上为 Babel/TypeScript 转译开启

11. 参考链接