05 初始化前端项目

更新时间：2019-07-24 14:26:47

> 如果不想在世界上虚度一生，那就要学习一辈子。
>
> ——高尔基

这一节我将带着大家来初始化一个前端项目，创建前端项目我们会使用到 vue cli 3 这个工具，是一个脚手架工具，所谓脚手架就是一个项目初期的结构，vue cli 3 帮助我们规范了项目初期的目录结构，构建配置等等，然后我们可以多把时间花在逻辑上，减少了繁琐的添加各种配置，但是也不排除我们在项目开发过程中会修改一些配置来满足我们的项目需求。

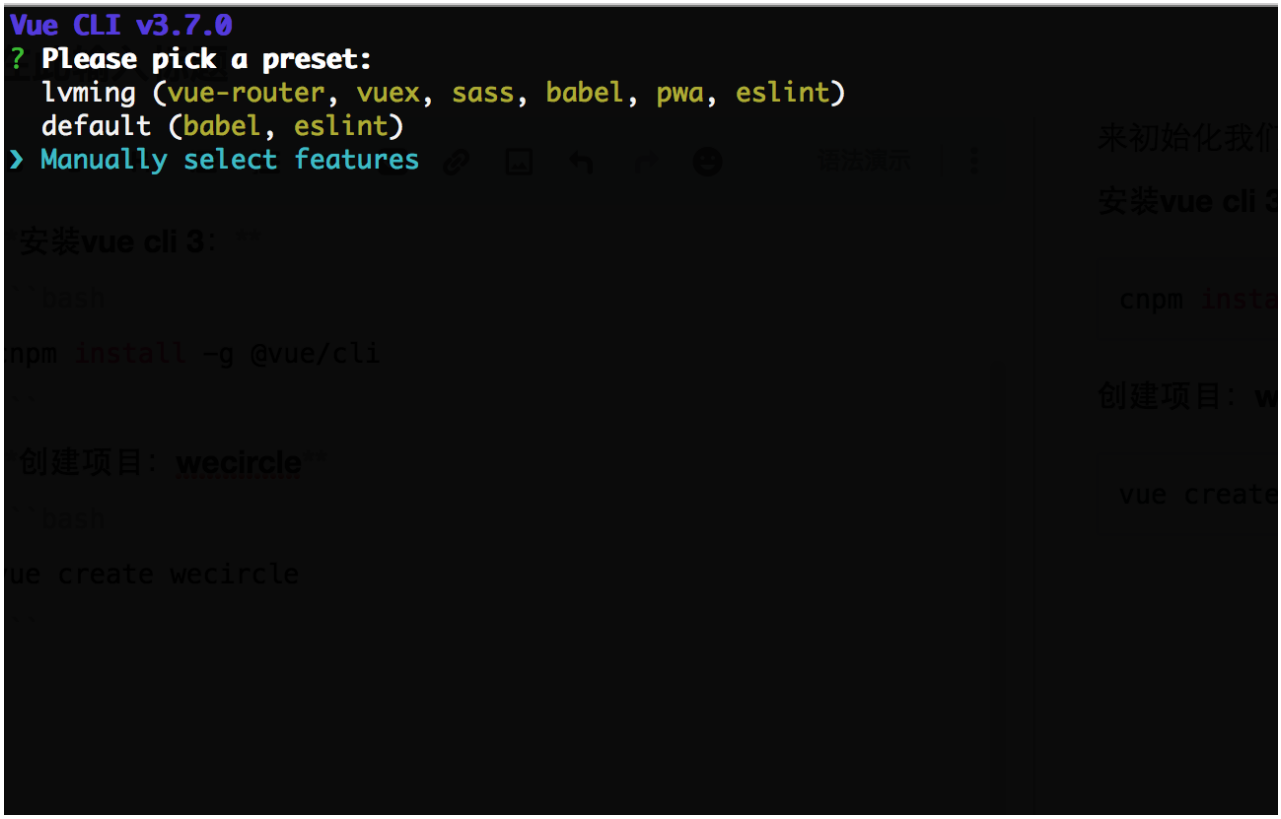
初始化前端项目

首先我们需要安装vue cli来初始化我们的前端项目框架：

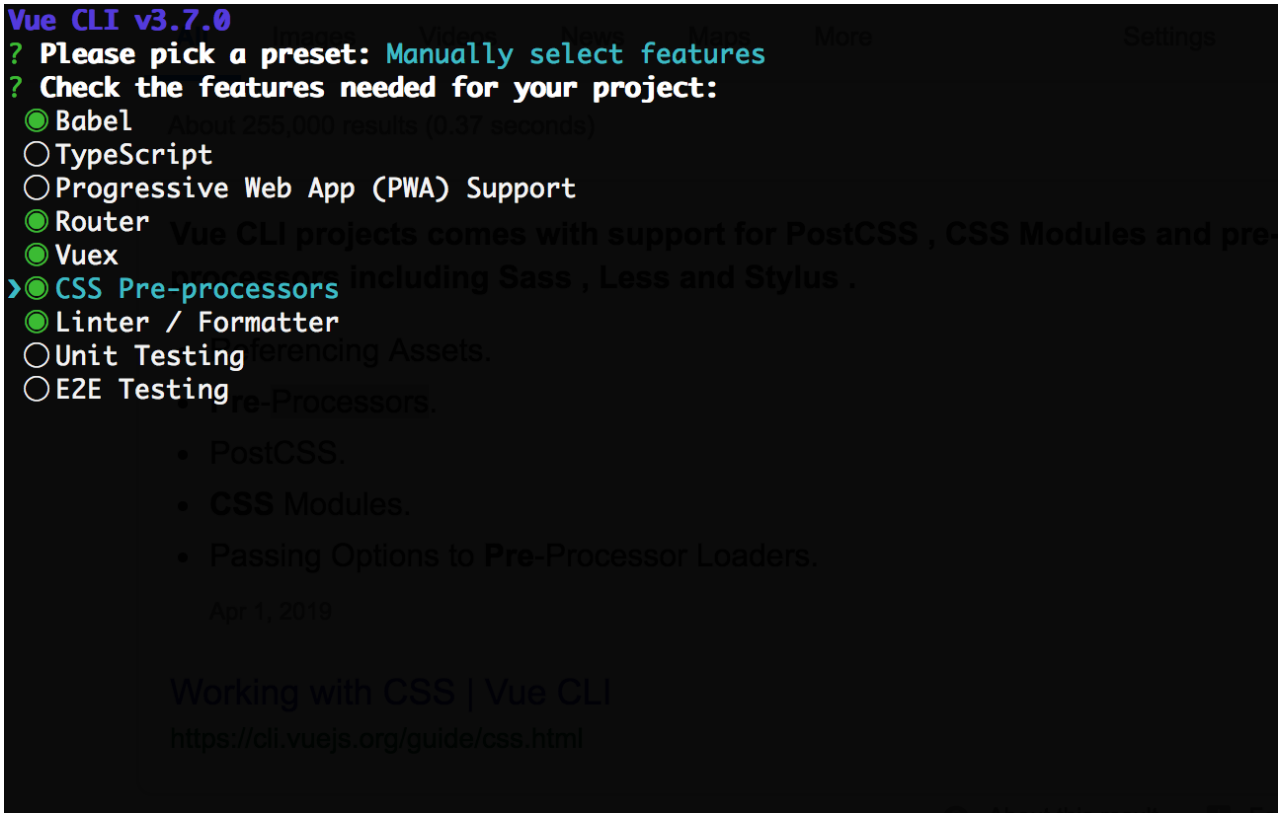
```
npm install -g @vue/cli
```

然后我们使用 vue/cli 创建一个项目：wecircle

```
vue create wecircle
```



选择Manually select features，表示我们不采用默认的模版，而是根据自己的情况选择需要安装的模块，例如vue-router，ESLint等等。



这一步主要选择我们需要的模块，这里说明一下：

- **Babel**：给我们提供了能够使用ES6的条件，Babel将我们的ES6代码转换成浏览器兼容性更强的ES5，这意味着，你可以现在就用ES6 编写程序，而不用担心现有环境是否支持，基本上现在的项目都会选择它；
- **Router**：这里的Router指的是vue-router，属于vue全家桶的一项，我们用它主要是帮助我们实现单页应用的页面

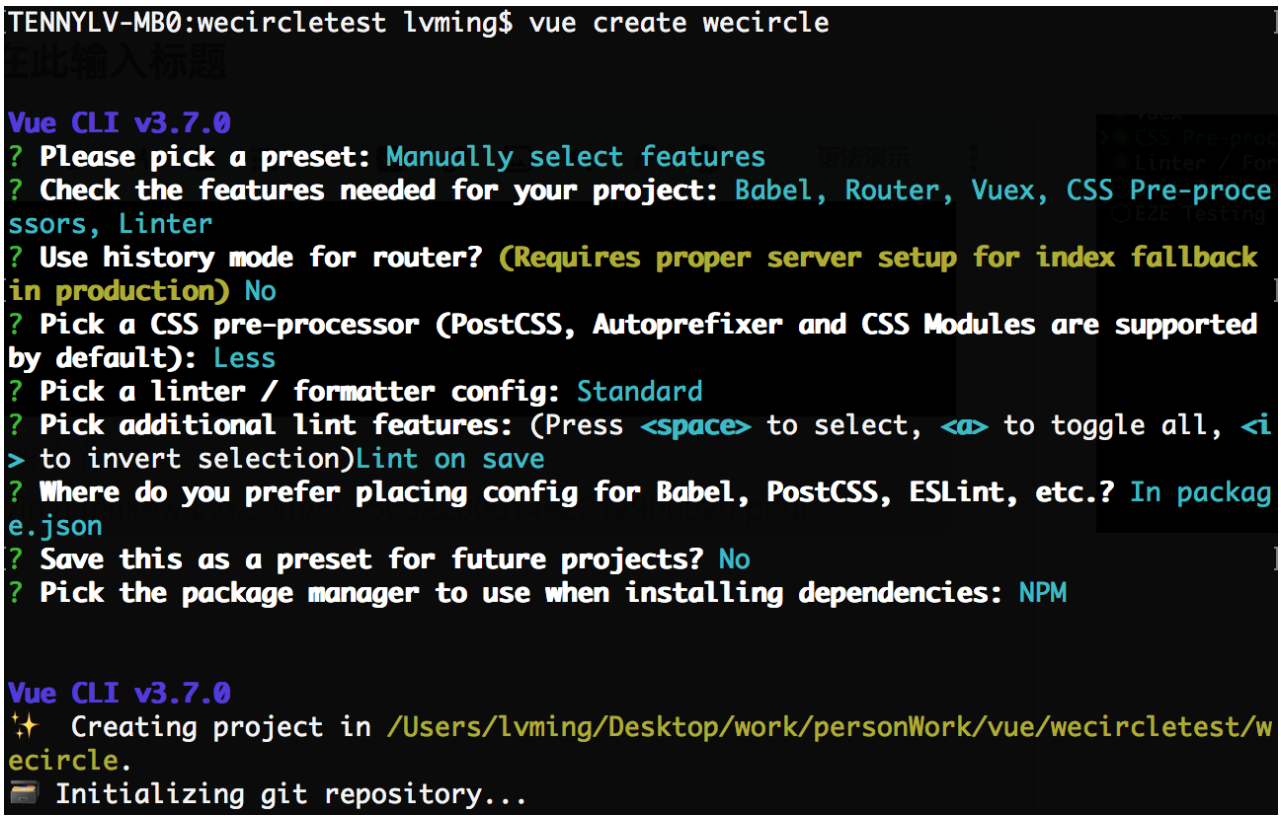
路由；

- **Vuex**：Vuex 是专门为 Vue.js 设计的状态管理库，它采用集中式存储管理应用的所有组件的状态，同时利用Vuex可以实现跨组件的通信。
- **CSS Pre-processors**：CSS的预处理工具选择，例如Sass，Less，stylus，同时默认会集成PostCss，PostCss和他们的区别在于：

1. PostCss是将最后生成的CSS进行处理，包括补充和提供一些而外的功能，比较典型的功能是将我们的CSS样式添加上不同浏览器的前缀例如：autoprefixer。
2. 另外三者称为CSS预处理工具，强调的是提供一些API，让我们编写CSS样式时更加具有代码逻辑，使我们的CSS更加有组织性，例如可以定义变量等等。

- **Linter/Formatter**：代码规范工具选择，现在主要用的就是ESLint，来帮我们处理代码规范问题。

创建项目的最后一步会看到我们选择的模块清单，这里解释几个地方：



history mode：是 vue-router 的路由模式，我们并没有选择 history，而是选择了默认的hash，这样我们可以 在url里清晰的看到页面的参数和当前的路径，更加清晰。

where do you prefer placing config for babel…有两个参数：

`In dedicated config files`：单独创建Bable，ESlint的配置文件。

`In package.json`：将Bable，ESlint这些配置文件继承在package.json里。

我们的项目选择单独的配置文件这种模式，也就是`In dedicated config files`，截图上选择的package.json生成文件之后表现就像下面这样：

```
/* package.json */
 {
  "name": "wecircle",
  "version": "0.1.0",
  "private": true,
  "scripts": {
   "serve": "vue-cli-service serve",
   "build": "vue-cli-service build",
   "lint": "vue-cli-service lint"
  },
  "dependencies": {
   "core-js": "^2.6.5",
   "vue": "^2.6.10",
   "vue-router": "^3.0.3",
   "vuex": "^3.0.1"
  },
  "devDependencies": {
   "@vue/cli-plugin-babel": "^3.7.0",
   "@vue/cli-plugin-eslint": "^3.7.0",
   "@vue/cli-service": "^3.7.0",
   "@vue/eslint-config-standard": "^4.0.0",
   "babel-eslint": "^10.0.1",
   "eslint": "^5.16.0",
   "eslint-plugin-vue": "^5.0.0",
   "vue-template-compiler": "^2.5.21"
  },
  "eslintConfig": {
   "root": true,
   "env": {
    "node": true
   },
   "extends": [
    "plugin:vue/essential",
    "@vue/standard"
   ],
   "rules": {},
   "parserOptions": {
    "parser": "babel-eslint"
   }
  },
  "postcss": {
   "plugins": {
    "autoprefixer": {}
   }
  },
  "browserslist": [
   "> 1%",
   "last 2 versions"
  ]
 }
```

这里是为了演示 package.json 的效果，我们项目里还是使用的单独的配置文件这种模式。

- **save this as a preset…**：表示你是否愿意将这次选择的模块存储成一个模版，以便下次创建项目时，可以直接选择。

OK,到这里我们的项目的初始文件和目录结构就已经完成了，它的目录结构应该是下面这个样子，看起来很清爽有木有，相比vue cli 2减少了很多东西，不过随着我们后面项目的进行，会不断新增代码和逻辑，就像建造一栋大楼，不断添砖加瓦。

```
├── public                  // 静态文件目录
│   ├── index.html          // 首页html
├── dist                    // 打包输出目录（首次打包之后生成）
├── src                     // 项目源码目录
│   ├── assets
│   ├── components
│   ├── views
│   ├── App.vue
│   ├── main.js
│   ├── router.js
│   ├── store.js
├── .editorconfig           // 编辑器配置项
├── .eslintrc.js            // eslint 配置项
├── .eslintignore.js        // eslint 忽略目录
├── postcss.config.js       // postCss配置项
├── babel.config.js         // babel配置项
├── vue.config.js           // 项目配置文件，用了配置或者覆盖默认的配置
└── package.json            // package.json
```

启动前端项目

打开 package.json 的 scripts，我们可以看到3个命令：`serve`、`build` 和 `lint`，分别使用 `npm run` 来运行这3个命令效果如下：

- 启动开发模式：npm run serve
- 启动生产模式打包：npm run build
- 启动代码规范检查，处理语法错误。：npm run build lint

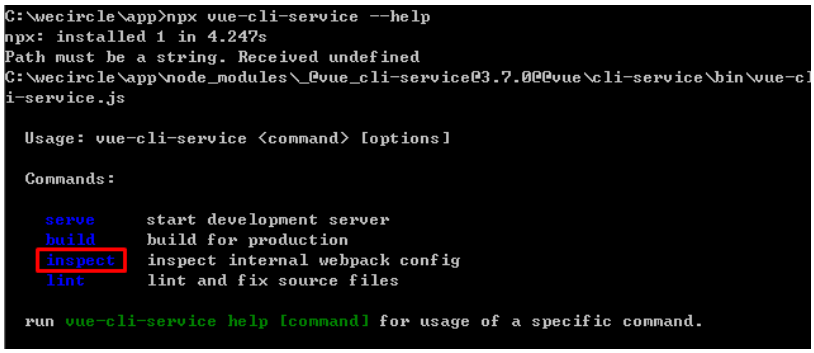
这三个命令都是基于vue-cli-service提供的命令，我们可以查看更多的配置参数：

```
# 命令：npm run serve 其他参数说明：
--open    在服务器启动时打开浏览器
--copy    在服务器启动时将 URL 复制到剪切版
--mode    指定环境模式 (默认值：development)
--host    指定 host (默认值：0.0.0.0)
--port    指定 port (默认值：8080)
--https   使用 https (默认值：false)
```

```
# 命令：npm run build 其他参数说明：
--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
```

```
#  命令：npm run lint 其他参数说明：
--format [formatter]  指定一个formatter （默认值：codeframe）
--no-fix              不修复错误
--no-fix-warnings     除了warnings（警告）错误不修复，其他的都修复
--max-errors [limit]  超过多少个错误就标记本次构建失败 (默认值：0)
--max-warnings [limit] 超过多少个warnings（警告）错误标记本次构建失败 (默认值：Infinity)
```

通过npx vue-cli-service --help命令查看，会发现有另外一个inspect命令：



vue-cli-service inspect，通过这个命令可以得到项目的webpack配置文件，由于vue cli 3将整个默认的webpack配置集成到了内部，所以单独对于webpack配置文件是不便于查看的，使用这个命令可以在当前项目的根目录得到一个webpack.config.xxx.js的配置文件。

```
npx vue-cli-service inspect
--mode    指定环境模式 (默认值：development)
```

小结

本章节主要讲解了使用vue cli 3生成我们的项目目录，并介绍了相关的命令配置，为后续的项目打下了基础。
相关知识点如下：

1. 前端脚手架的概念解释。
2. vue cli 3能够帮助我们生成使用的前端框架，省去了繁琐的配置，但是需要注意的是，基本的配置项还是需要掌握。
3. vue-cli-service的命令参数和使用方法。

}

04 屏幕适配方案选择

06 初始化后端项目