1. **什么是CLI脚手架？**
2. **CLI是Command-Line Interface，编译为命令行界面，但是俗称脚手架。**
3. **Vue CLI是一个官方发布vue.js项目脚手架**
4. **使用vue-cli可以快速搭建vue开发环境以及对应的webpack配置。**
5. **什么时候要使用脚手架？**

**如果你只是简单几个vue的demo程序，那则不需要vue CLI，如果是在开发大型项目，那么你需要，并且必然需要使用vue CLI。**

1. **脚手架的好处：**
2. **使用vue.js开发大型应用时，我们需要考虑代码目录结构、项目结构和部署、热加载、代码单元测试等事情。**
3. **如果每个项目都是手动完成这些工作，那无疑效率比较低效，所以通常我们会使用一些脚手架工具来半年关注完成这些事情。**
4. **什么是NPM呢？**
5. **NPM的全称是Node Package Manager**
6. **是一个NodeJS包管理和分发工具，已经成为了非官方的发布Node模块(包)的标准**
7. **NPM的官方镜像是非常慢的，这里推荐使用淘宝NPM镜像**
8. **淘宝镜像安装命令：npm install –g cnpm –registry=https://registry.npm.taobao.org**
9. **使用cnpm命令来安装模块：cnpm install [name]**
10. **vue.js官方脚手架工具就使用了webpack模块**
11. **对所有的资源会压缩等优化操作**
12. **它在开发过程中提供了一套完整的功能，能够使得我们开发过程中变的高效**
13. **webpack的全局安装：**
14. **npm install webpack –g**
15. **查看是否暗转webpack：webpack -v**
16. **安装vue脚手架**
17. **npm install –g @vue/cli：脚手架3**
18. **安装脚手架2：npm install @vue/cli-init –g**
19. **脚手架2创建项目：vue init webpack my-project[项目名称]**
20. **脚手架3创建项目：vue create my-project[项目名称]**
21. **查看脚手架：vue –version**
22. **创建项目文件夹：vue init webpack [项目名称，以英文命名]**
23. **在按住啊安脚手架项目的时候，对应的选择内容：**
24. **project name 项目名称**
25. **project description 项目的描述信息**
26. **author 作者默认有作者(获取到的是Gitconfig文件中的信息)，也可以自己输入**
27. **有两种构建项目方式runtime build**
28. **runtime+compiler(推荐)**
29. **runtime+only(用的比较多)：比上面那种内存小6KB，运行效率更高**
30. **install vue-router(y/n)：是否选择下载vue路由**
31. **use eslint to lint your code(y/n)：是否对ES(js)[lint限制]代码进行限制，使js代码变的规范**
32. **如果你上面这个选择了yes那则：pick an Eslint perset：选择一种规范：**
33. **standard：标准规范：**[**https://github.com/standard/standard**](https://github.com/standard/standard)**(使用)**
34. **airbnf：爱彼迎，做名宿：https://github.com/airbnb/javascript**
35. **none(configure it yourself)：自己写，属于自己的配置文件**
36. **set u unit tests(y/s)：单元测试(n)**
37. **setup e2e tests with nightwatch(y/n)：(e2e：端到端测试，写自动化)让你使用端到端测试，需要使用nightwatch，需要依赖selenium。(n)**
38. **should we run `npm install` for after the project has been created?你准备在管理项目的时候是准备用npm还是yarh呢？**
39. **yes use npm(使用这个)**
40. **yes use yarh(谷歌推出的管理工具)**
41. **no I will handle that myself**

**14vuecli-cli2目录结构分析**

**插槽slot**

1. **为什么实用slot插槽？**
2. **在生活中很多地方都有插槽，电脑的USB插槽，插板当中的电源插槽等**
3. **插槽的目的是让我们原来的设备具备跟多的扩展性**
4. **比如电脑的USB我们可以插入U盘、硬盘、手机等等**
5. **相当于占位符**
6. **组件的插槽：**
7. **组件的插槽也是为了让我们封装的组件更加具有扩展性。**
8. **让使用者可以决定组件内部的一些内容到底展示什么。**
9. **如何去封装这类的组件呢？**
10. **他们有很多区别，但是也有很多共性。**
11. **如果，我们每一个单独去封装一个组件，显然不合适：比如每个页面都返回，这部分内容我们就要重复去封装。**
12. **但是，如果我们封装成一个，好像也不合理：有些左侧是菜单，有些是返回，有些中间是搜索，有些是文字等等。**
13. **如何封装合适呢？抽取共性，保留不同。**
14. **最好的封装烦恼歌是就是 将共性抽取到组件中，将不同暴露为插槽。**
15. **一旦我们预留了插槽，就可以让使用者根据自己的需求，决定插槽中插入什么内容**
16. **是搜索框，还是文字，都由调用者自己来决定。**
17. **具名插槽**
18. **当子组件的功能复杂时，子组件的插槽可能并非是一个。**
19. **设置具名插槽**

**<slot name=”center”>中间</slot>**

1. **使用具名函数**

**<span slot=”center”>替换的内容</span>**

1. **编译作用域**
2. **下面代码能否被渲染出来：**
3. **A.<my-cpn v-show=”isShow”><my-cpn>中，我们使用了isShow属性。**
4. **isShow属性包含在组件中，也包含在Vue实例中**
5. **可以，因为：**
6. **父组件模板的所有东西都会在父级作用域内编译；子组件模板的所有东西都会在子级作用域内编译。**
7. **而我们在使用<my-cpn v-show=”isShow”></my-cpn>的时候，整个组件的使用过程是相当于在父组件中出现的。**
8. **那么他的作用域就是父组件，使用的属性也是属于父组件的属性。**
9. **因此，isShow使用的是vue实例中的属性，而不是子组件的属性**
10. **作用域插槽：**
11. **父组件替换插槽的标签，但是内容由子组件来提供。**