1. Vue CLI

1.1. 什么是Vue CLI

使用Vue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，如果每个项目都要手动完成这些工作，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帮助完成这些事情

CLI是什么意思

- CLI是Command-Line Interface，翻译为命令行界面，俗称脚手架
- Vue CLI是一个官方发布的Vue.js项目脚手架
- 使用Vue-CLI可以快速搭建Vue开发环境以及对应的webpack配置

1.2. Vue CLI的使用

安装Vue脚手架

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

测试是否安装成功

```
vue --version
```

1.3. Vue CLI2

项目初始化

```
vue webpack 项目名称（英文，不要用中文）
```

项目作者

项目作者默认读取 git config
需要更改的话在文件目录：C:\Users\Admin中的.gitconfig中更改

单元测试

e2e:end to end 端到端测试，安装Nightwatch，是一个利用selenium或webdriver或phantomjs等进行自动化测试的框架

Vue CLI 安装不成功： 使用命令：

```
clean cache --force
```

相当于清空文件夹：
C:\Users\Admin\AppData\Roaming\npm-cache

初始化时选择ESlint开启，之后想要关闭怎么办?

在config文件夹中index.js文件内 useEslint由true改为false

项目初始化时runtime+compiler和runtime-only如何选择

区别:

runtimecompiler:

```
new Vue({
    el:'#app',
    template:'<App/>',
    components:{APP}
})
```

编译过程：template->ast->render->vdom->UI

runtimeonly:

```
new Vue({
    el:'#app',
    render:h=>h(App)
})
```

编译过程：render->vdom->UI(性能更高，代码量更少)

2. Vue CLI3

2.1. 认识Vue CLI3

Vue-CLI3与2版本有很大区别：

- vue-cli3是基于webpack4打造
- vue-cli3的设计原则是0配置，移除配置文件根目录下的build和config等目录
- vue-cli3提供了vue ui命令，提供了可视化配置，更加人性化
- 移除了static文件夹，新增了public文件夹，并且index.html移动到public中

初始化:

Please pick a preset(配置):

删除自定义初始设置选项:

在文件目录下：C:\Users\Admin.vuerc 删除presets中的想要删除的配置即可
rc:run command(运行终端)

2.2. ES6补充：箭头函数

```
<script>
    // 箭头函数：也是一种定义函数的方式
    // 1. 定义函数的方式：function
    const a=function(){
```

```
    }

    // 2. 对象字面量中定义函数
    const obj={

        // 2.1.
        b1:function(){

        },
        // 2.2.
        b2(){

        }
    }

    // 3. 箭头函数
    const c=(参数列表)=>{

    }
</script>
```

2.3. vue-router

2.3.1. 什么是路由

> 路由(routing)就是通过互联的网络把信息从源地址传输到目的地地址的活动

路由器提供了两种机制：路由和转送：

- 路由是决定数据包从来源到目的地的路径
- 转送是将输入端的数据转移到合适的输出端

路由中有一个非常重要的概念叫路由表：
路由表本质上是一个映射表，决定了数据包的指向

```
映射表：[
    内网ip1:mac地址1,
    内网ip2:mac地址2
]
```

补充知识：前端渲染与后端渲染

1. 网络发展第一个阶段：后端渲染

例如：jsp:java server page
在服务器端渲染网页，并把渲染好的数据发送给客户端
一个网站有多个页面，每个页面有自己对应的网址，即URL
URL会发送到服务器，服务器会通过正则对该URL进行匹配，并且最后交给一个Controller进行处理
Controller进行各种处理，最终会生成HTML或者数据，返回给前端

上述操作就是后端路由：

- 当我们页面中需要请求不同的路径内容时，交给服务器来进行处理，服务器渲染好整个页面，并且将页面返回给客户端
- 这种情况下渲染好的页面不需要单独加载任何的js和css，可以直接交给浏览器展示，这样也有利于SEO的优化

后端路由的缺点：

- 一种情况是整个页面的模块由后端人员来编写和维护的
- 另一种情况是前端开发人员如果要开发页面，需要通过PHP和Java来编写页面代码
- 通常情况下HTML代码和数据以及对应的逻辑会混在一起，编写和维护都是非常糟糕的事情

> 后端路由： 后端处理URL与页面之间的映射关系

2. 前后端分离阶段

后端只负责数据，不负责任何阶段的内容

- 随着Ajax的出现，有了前后端分离的开发模式
- 后端只提供API来返回数据，前端通过Ajax获取数据，并且可以通过JavaScript将数据渲染到页面中
- 这样做最大的优点是前后端责任的清晰，后端专注于数据上，前端专注于交互和可视化上
- 当移动端出现后，后端不需要进行任何处理，依然使用之前的一套API即可

后端服务器分为：提供API接口的服务器和静态资源服务器
HTML,css,javascript代码都在静态资源服务器上
当用户从客户端浏览器请求URL地址时，浏览器从静态资源服务器上获取HTML,css,javascript代码，
html和css代码直接在浏览器上渲染
浏览器执行js代码中的Ajax代码从提供API接口的服务器中请求数据
请求回来的数据由其他js代码插入到页面中进行渲染

> 前端渲染：浏览器中显示的网页中的大部分内容，都是由前端写的js代码在浏览器中执行，最终渲染出来的网页

3. 单页面富应用阶段：

- 单页面富应用(Single Page Application SPA)最主要的特点就是在前后端分离的基础上加了一层前端路由
- 前端来维护一套路由规则

SPA页面：整个网页只有一个html页面

前端路由的核心是：

- 改变URL，但是页面不进行整体的刷新

2.3.2. 修改URL地址但不刷新网页

- 修改URL的hash值
- HTML5的history模式：pushState

2.3.2.1. URL的hash

url的hash也就是锚点(#)，本质上是改变window.location的href属性
我们可以通过直接赋值location.hash来改变href，但是页面不发生刷新

2.3.2.2. HTML5的history模式

```
// history.pushState({data},'title','url')
// 例如：
history.pushState({},'','home')
```

history的push和back就是栈结构的入栈和出栈，也相当于浏览器的前进和后退操作
使用pushState和back在浏览器中进行网页的切换时浏览器会保存之前和之后的页面信息，使用浏览器的前进和后退按钮就可以进行跳转

使用replaceState()在浏览器中进行网页切换时是替代先前的页面，先前页面的数据不保存，也就无法返回

history.go(-1)效果等同于history.back()

2.3.3. 认识vue-router

目前前端流行的三大框架，都有自己的路由实现：

- Angular的NGRouter
- React的ReactRouter
- Vue的vue-router

> vue-router是Vue.js官方的路由插件，它和Vue.js是深度集成的，适合于构建单页面应用

vue-router是基于路由和组件的：

- 路由用于设定访问路径，将路径和组件映射起来
- 在vue-router的单页面应用中，页面的路径的改变就是组件的切换

2.3.4. 安装和使用vue-router

步骤一：安装vue-router

```
npm install vue-router --save
```

也可以在使用Vue CLI初始化项目的时候选择使用vue-router

步骤二：在模块化工程中使用它（因为是一个插件，所以可以通过Vue.use()来安装路由功能）

- 第一步：导入路由对象，并且调用Vue.use(VueRouter)
- 第二步：创建路由实例，并且传入路由映射配置
- 第三步：在Vue实例中挂载创建的路由实例

使用vue-router的步骤：

- 创建路由组件
- 配置路由映射：组件和路径映射关系
- 使用路由：通过<router-link>和<router-view>

<router-link>和<router-view>

- <router-link>:该标签是一个vue-router中已经内置的组件，它被渲染成一个<a>标签
- <router-view>：该标签会根据当前的路径，动态渲染出不同的组件
- 网页的其他内容，比如顶部的标题/导航，或者底部的一些版权信息会和<router-view>处于同一个等级
- 在路由切换时，切换的是<router-view>挂载的组件，其他内容不会发生改变

2.3.5. router-link补充

在前面的<router-link>中，我们只是使用了一个属性：to，用于指定跳转的路径

<router-link>还有一些其他属性：

- tag:tag可以指定<router-link>之后渲染成什么组件，可以渲染成li 、button等，默认的是a
- replace:replace不会留下history记录，所以指定replace的情况下，后退键返回不能返回到上一个页面中
- active-class:当<router-link>对应的路径匹配成功时，会自动给当前元素设置一个router-link-active的class,设置active-class可以修改默认的名称
    - 在进行高亮显示的菜单导航或者底部tabbar时，会使用到该类
    - 但是通常不会修改类的属性，会直接使用默认的router-link-active即可