预习资料

名称	链接	备注
Vue2.0	https://github.com/vuejs/vue	2.0源码
Vue3.0	https://github.com/vuejs/vue-next	3.0源码
Vuex	https://github.com/vuejs/vuex	vuex源码
Vue-Router	https://github.com/vuejs/vue-router	vue-router源码
Nuxt	https://github.com/nuxt	Nuxt源码

课程目标

Vue CLI

Vue CLI 是一个基于 Vue.js 进行快速开发的完整系统，提供：

- 通过 `@vue/cli` 实现的交互式的项目脚手架。
- 通过 `@vue/cli` + `@vue/cli-service-global` 实现的零配置原型开发。
- 一个运行时依赖 (`@vue/cli-service`)，该依赖：
  - 可升级；
  - 基于 webpack 构建，并带有合理的默认配置；
  - 可以通过项目内的配置文件进行配置；
  - 可以通过插件进行扩展。
- 一个丰富的官方插件集合，集成了前端生态中最好的工具。
- 一套完全图形化的创建和管理 Vue.js 项目的用户界面。

Vue CLI 致力于将 Vue 生态中的工具基础标准化。它确保了各种构建工具能够基于智能的默认配置即可平稳衔接，这样你可以专注在撰写应用上，而不必花好几天去纠结配置的问题。与此同时，它也为每个工具提供了调整配置的灵活性，无需 eject。

安装

Vue.config.js 配置

vue.config.js 是一个可选的配置文件，如果项目的 (和 `package.json` 同级的) 根目录中存在这个文件，那么它会被 `@vue/cli-service` 自动加载。 Vue CLI 3.X有默认的config配置但是都集成到webpack当中，开发者看不到，如果需要自己配置，需要自己手写vue.config.js覆盖相关默认的config配置.自定义相关配置如下：(部分)

```
//vue.config.js
// 部分配置如需要完全配置 可查看 https://cli.vuejs.org/zh/config/#vue-config-js
module.exports = {
 //部署应用包时的基本 URL 可相对 可绝对
 publicPath:'/',
 //生成的生产环境构建文件的目录
 outputDir: 'dist',
 //放置生成的静态资源 (js、css、img、fonts) 的 (相对于 outputDir 的) 目录
 assetsDir: 'static',
 //指定生成的 index.html 的输出路径 (相对于 outputDir)。也可以是一个绝对路径
 indexPath:'index.html',
 //生成的静态资源在它们的文件名中包含了 hash 以便更好的控制缓存。静态资源更好的进行hash，以便更新缓存
 filenameHashing: true,
 //在 multi-page 模式下构建应用。每个“page”应该有一个对应的 JavaScript 入口文件。其值应该是一个对象，对象的 key 是入口的名字，value 是：
 // 一个指定了 entry, template, filename, title 和 chunks 的对象 (除了 entry 之外都是可选的)；
 // 或一个指定其 entry 的字符串。
 pages: {
 index: {
 // page 的入口
```

```
  entry: 'src/main.js',
  // 模板来源
  template: 'public/index.html',
  // 在 dist/index.html 的输出
  filename: 'index.html',
  // 当使用 title 选项时，
  // template 中的 title 标签需要是 <title><%= htmlWebpackPlugin.options.ti
tle %></title>
  title: '首页2',
  // 在这个页面中包含的块，默认情况下会包含
  // 提取出来的通用 chunk 和 vendor chunk。
  chunks: ['chunk-vendors', 'chunk-common', 'index']
  }
  },
  //是否使用包含运行时编译器的 Vue 构建版本。设置为 true 后你就可以在 Vue 组件
中使用 template 选项了，但是这会让你的应用额外增加 10kb 左右。
  runtimeCompiler:false,
  //默认情况下 babel-loader 会忽略所有 node_modules 中的文件。如果你想要通过
Babel 显式转译一个依赖，可以在这个选项中列出来。
  transpileDependencies:[],
  //如果你不需要生产环境的 source map，可以将其设置为 false 以加速生产环境构
建。
  productionSourceMap:false,
  //设置生成的 HTML 中 <link rel="stylesheet"> 和 <script> 标签的 crossorig
in 属性。
  // 需要注意的是该选项仅影响由 html-webpack-plugin 在构建时注入的标签 - 直接
写在模版 (public/index.html) 中的标签不受影响。
  crossorigin:undefined,
  //在生成的 HTML 中的 <link rel="stylesheet"> 和 <script> 标签上启用 Subres
ource Integrity (SRI)。如果你构建后的文件是部署在 CDN 上的，启用该选项可以提供额
外的安全性。
  // 需要注意的是该选项仅影响由 html-webpack-plugin 在构建时注入的标签 - 直接
写在模版 (public/index.html) 中的标签不受影响。
  integrity:false,
  devServer:{
  hot:true,
  open:true
  }

};
```

单文件组件

- 提供了更好的封装性
- 优雅的模板支持
- scoped css支持(以便css全局污染)
- 通过vue-loader可以配合各种预处理器进行构建

命名规则

camelCase(小驼峰):videoExampleCompontent
PascalCase(大驼峰)：VideoExampleCompontent
kebab-case(烤串命名)：video-example-compontent

命名规范在vue中的使用场景:

camelCase(小驼峰)：js函数 变量 组件间传值props:{bigProp:String}

PascalCase(大驼峰)：js类 组件文件 components/VideoList.vue VideoItem.vue

kebab-case(烤串命名): HTML 引入组件 components:{'video-item':videoItem}
模板中使用最好都使用烤串命名方式 <video-item big-prop='2' /> 注意(HTML是大小写不敏感的)