1. SPA概念

	单页面应用（SinglePage Web Application，SPA）	多页面应用（MultiPage Application，MPA）
组成	一个外壳页面和多个页面片段组成	多个完整页面构成
资源共用(css,js)	共用，只需在外壳部分加载	不共用，每个页面都需要加载
刷新方式	页面局部刷新或更改	整页刷新
url 模式	a.com/#/pageone a.com/#/pagetwo	a.com/pageone.html a.com/pagetwo.html
用户体验	页面片段间的切换快，用户体验良好	页面切换加载缓慢，流畅度不够，用户体验比较差
转场动画	容易实现	无法实现
数据传递	容易	依赖 url传参、或者cookie 、localStorage等
搜索引擎优化(SEO)	需要单独方案、实现较为困难、不利于SEO检索 可利用服务器端渲染(SSR)优化	实现方法简易
试用范围	高要求的体验度、追求界面流畅的应用	适用于追求高度支持搜索引擎的应用
开发成本	较高，常需借助专业的框架	较低，但页面重复代码多
维护成本	相对容易	相对复杂

2.vue-router

- 开始

```
const Foo = { template: '<div>foo</div>' }
const Bar = { template: '<div>bar</div>' }

const routes = [
  { path: '/foo', component: Foo },
  { path: '/bar', component: Bar }
]
const router = new VueRouter({
  routes // (缩写) 相当于 routes: routes
})
```

- 动态路由匹配
- 嵌套路由
- 编程式导航 (js跳转) vs 声明式导航<router-link>
- 命名路由 （**$route.name** 获取命名路由的名字)
- 重定向和别名

```
const router = new VueRouter({
  routes: [
    { path: '/a', redirect: '/b' }
  ]
})
```

```
const router = new VueRouter({
  routes: [
    { path: '/a', component: A, alias: '/b' }
  ]
})
```

- HTML5 History模式

vue支持两种模式

a. hash #/home
b. history /home

- 路由守卫&路由拦截
  - 全局拦截
  - 单个拦截
- 路由懒加载

```js
const Foo = () => import('./Foo.vue')
```

在路由配置中什么都不需要改变，只需要像往常一样使用 `Foo`：

```js
const router = new VueRouter({
  routes: [
    { path: '/foo', component: Foo }
  ]
})
```

3. 路由原理：

(1)hash路由 ==> location.hash 切换

window.onhashchange 监听路径的切换

(2)history路由==> history.pushState 切换

window.onpopstate 监听路径的切换

4. 项目

(1) 启动案例项目开发

(2) 利用vue-router搭建项目SPA结构

5. 全家桶

- vue cli
- vue router
- vuex