# Vue-cli

1. 安装vue-cli

（前提：已安装node.js，内含npm）

**npm install vue-cli -g**

1. 初始化项目

**vue init <template-name> <project-name>**

\*（<template-name>：表示模板名称，vue-cli官方为我们提供了5种模板，

webpack-一个全面的webpack+vue-loader的模板，功能包括热加载，linting,检测和CSS扩展。

webpack-simple-一个简单webpack+vue-loader的模板，不包含其他功能，让你快速的搭建vue的开发环境。

browserify-一个全面的Browserify+vueify 的模板，功能包括热加载，linting,单元检测。

browserify-simple-一个简单Browserify+vueify的模板，不包含其他功能，让你快速的搭建vue的开发环境。

simple-一个最简单的单页应用模板。）

初始化完成后会提示三条命令：

**cd <project-name>**

**npm install**

**npm run dev**

1. Vue-cli项目结构

├── README.md // 项目说明文档

├── node\_modules // 项目依赖包文件夹

├── build // 编译配置文件

│ ├── build.js

│ ├── check-versions.js

│ ├── dev-client.js

│ ├── dev-server.js

│ ├── utils.js

│ ├── vue-loader.conf.js

│ ├── webpack.base.conf.js

│ ├── webpack.dev.conf.js

│ └── webpack.prod.conf.js

├── config // 项目基本设置文件夹

│ ├── dev.env.js // 开发配置文件

│ ├── index.js // 配置主文件

│ └── prod.env.js // 编译配置文件

├── index.html // 项目入口文件

├── package-lock.json // npm5 新增文件，优化性能

├── package.json // 项目依赖包配置文件

├── src // 我们的项目的源码编写文件

│ ├── App.vue // APP入口文件

│ ├── assets // 初始项目资源目录，回头删掉

│ │ └── logo.png

│ ├── components // 组件目录

│ │ └── Hello.vue // 测试组件，回头删除

│ ├── main.js // 主配置文件

│ └── router // 路由配置文件夹

│ └── index.js // 路由配置文件

└── static // 资源放置目录

1. 新建组件流程
2. 在src/components目录下，新建.vue文件
3. --在router/index.js文件的上边引入组件（import newcomponent from ‘@/components/newcomponent’）

--在router/index.js文件的routes[]数组中，新增加一个对象

{

Path: ‘/newcomponent’,

Name: ‘newcomponent’,

Component: Newcomponent

}

1. Vue-router传参
2. 通过router/index.js中设置的name传参

{{ $router.name}}

1. 通过<router-link> 标签中的to传参

<router-link :to="{name:'newcomponent',params:{name:'jspang'}}">传参</router-link>

在components/newcomponent.vue下使用{{ $route.params.name }}

1. 通过url传参

Router/index.js：

{

path: '/params/:newsId/:newsTitle',

component: News

}

点击跳转处：

<router-link to="/params/198/jspang website is very good">News</router-link>

News.vue：

{{ $route.params.newsId}}

{{ $route.params.newsTitle}}

（可携带正则）

1. Vue-router重定向

{

Path: ‘/goback’,

Redirect: ‘/’

}

（可传参）

1. 路由过渡动画

<transition name=”fade”>，会有如下四个CSS类名：

fade-enter:进入过渡的开始状态，元素被插入时生效，只应用一帧后立刻删除。

fade-enter-active:进入过渡的结束状态，元素被插入时就生效，在过渡过程完成后移除。

fade-leave:离开过渡的开始状态，元素被删除时触发，只应用一帧后立刻删除。

fade-leave-active:离开过渡的结束状态，元素被删除时生效，离开过渡完成后被删除。

过渡模式mode：

in-out:新元素先进入过渡，完成之后当前元素过渡离开。

out-in:当前元素先进行过渡离开，离开完成后新元素过渡进入。

1. 404页面

{

Path: ‘\*’,

component: Error

}

1. 编程式导航

this.$router.go(-1);

this.$router.push('/');

1. 路由配置文件中的钩子函数

/src/router/index.js中钩子函数

{

path:'/params/:newsId(\\d+)/:newsTitle',

component:Params,

beforeEnter:(to,from,next)=>{

console.log('我进入了params模板');

console.log(to);

console.log(from);

next();

},

to:路由将要跳转的路径信息，信息是包含在对像里边的。

from:路径跳转前的路径信息，也是一个对象的形式。

next:路由的控制参数，常用的有next(true)和next(false)。

模板中的钩子函数

beforeRouteEnter：在路由进入前的钩子函数。

beforeRouteLeave：在路由离开前的钩子函数。

export default {

name: 'params',

data () {

return {

msg: 'params page'

}

},

beforeRouteEnter:(to,from,next)=>{

console.log("准备进入路由模板");

next();

},

beforeRouteLeave: (to, from, next) => {

console.log("准备离开路由模板");

next();

}

}

1. Vuex--State

通过computed的计算属性直接赋值

computed属性可以在输出前，对data中的值进行改变，我们就利用这种特性把store.js中的state值赋值给我们模板中的data值。

computed:{

count(){

return this.$store.state.count;

}

}

通过mapState的对象来赋值

import {mapState} from 'vuex';

computed:mapState({

count:state=>state.count

})

通过mapState的数组来赋值

computed:mapState(["count"])