组件：

全局组件：

Vue.compontens（‘name’,{

}）

局部组件：

（需要在父级上注册一下）

组件的传值

data（）{

return {

}

}

computed：{

get（）{

}；

set（）{

}

}

父子组件传值

父级在子级上绑定事件

·····子级上触发事件并传值

······Vue.$emit（‘name’，）触发自定义事件

----------------------------------------------------------

第1节(vue-router)构建单页面应用 single page application

1，main.js(入口文件) App.vue(入口组件)

引入Vue框架

引入入口组件

new Vue（{

el : '#app',

components:{

app

},

templete:'<app></app>',

router

}）

2，组件的概念（懒加载： 使第一次加载的文件不那么大）

3，引入vue-router

Vue.use(vue-router) // 使用vue-router 这个组件 （具体去vue官网api查看）

创建一个vue-router实例

const router = new VueRouter ({

})

4，将其放入一个单独的文件夹（在main.js中引入）

import home from '' (引入所需的组件)

export default new VueRouter({

routes:[

{

path: ' /home ', (对应路径的是一个组件)

name： ’home‘， （防止页面路径过长）

component：home

}

],

mode: ‘hash’ / ‘history’ (有一个刷新的动作)

})

\* 将组建展示出来<router-view></router-view>

\* 锚点

<a href=''></a>

<router-link to=''></router-link>不管是什么mode 页面都没有刷新

\* to 可以是

：to =" {path : "/home"}" 或 “{name : "home"}”

5 , 动态路由匹配

path: ' /home/:id?', 有无id都在home路径上

6 ,url 风格

restful 风格 home/：home\_id / mumber/ : id

访问的是 home/23/mumber.12345

params={home\_id:23,id:12345}

no restful 风格 home?id=123&home\_id= 345

访问的是 home

query={ id：123，home\_id： 345}

7 ，配置一个根路由

其他没有配置的路由

{

path：‘ \* ’,

component: err (404 not found)

}

重定向

{

path： ' \* ',

redirect: '/home'

}

redirect : { path: '/doc' }

redirect(to){

（to : 路由信息 包括params query）

return ‘/home’

return {name :home ,params:{id:123,home\_id:123}} (直接修改params中的值)

}

怎么拿到url中的数据

$route.params

配合watch：{} 方法使用

----------------------------------------------------------

第2节（vue-router）

1， 嵌套路由

routers：【

{

path：“/home",

component:home,

name :"home",

children:[

{

path:"person",

name : "person",

component: person

}

]

}

】

home组件中也要有<router-view></router-view>

<router-link></router-link>也一样

router-link 中的属性 tag 表示要渲染成什么标签

router-view 可以加class标签

2， router-link 点击后出现的class类名

router-link-active 点击的路由（子级被点击 父级也会存在）

router-link-exact-active 点击存在的路由（真路由）

\* 在VueRouter可以配置 使类名变化

linkActiveClass: '',

linkExactActiveClass: ''

也可以具体在某个router-link标签上添加

active-class

active-exact-class

3，一个路径上加载多个组件（使用多个路由）

components：{

default：doc，

top：home

}

在router-view中添加一个name字段

4，过渡动效

<transition>

</transition>

v-enter

v-leave

v-enter-active

v-leave-active

5，路由原信息

在VueRouter中routes中配置

meta：{

index：1

}

可读可写

在组件中(App.vue)添加一个watch方法监听

watch：{

$route（to,from）{

}

}

6，路由守卫

针对于路由对象（new VueRouter（））的守卫 beforeEach

在配置router中配置

router.beforeEach((to ,from,next)=>{

})

to 从哪个路由

from去哪个路由

next() 执行 就跳转

next（）中可以添加方法

针对于单个路由（router数组中每个路由）的守卫 beforeEnter

在routs中添加一个beforeEnter方法函数

针对组件（component）的守卫 beforeRouterEnter

在组件中配置添加beforeRouterEnter方法

7，路由的懒加载

把component：（）=>{

import(../components/home)

}

----------------------------------------------------------

第3节

1， 非父子组件之间通信（vuex）

a,通过父子组件传值的方式一层一层的传（层级深了非常繁琐）

vuex引入

b，相当于一个u盘 把公共数据放入进去

npm install vuex --save

import Vuex from 'vuex';

Vue.use(Vuex);

创建一个文件夹（store）单独配置vuex

new Vuex.Store（{

}）

5大部分

state 状态 数据

mutations 定义修改数据方法(同步)

actions 定义异步修改数据的方法

getters 根据state当中数据派生出新的值 computed计算属性

modules：{ 递归的形势

a：{

state

mutations

actions

getters

modules：{

}

}

}

1.state

state：{

val：‘hello’

}

在哪里拿到

1,$store.state.val

2,用es6的解构拿到vuex的mapState

import { mapState } from ‘vuex’；

export default{

computed ：mapState（【val】）(返回的是一个对象 可以用...的方式将其拆开 以便使用其他的computed的值)

}

mapState 也可以传对象{ newVal：val => state.val }和模块化有关

2,mutations:{

changeVal(state,newVal){

state.val = newVal

}

}

1,用commit 触发在mutations中定义的方法

watch:{

val(){

this.$store.commit('changeVal',this.val)

}

}

2,也可以用解构mapMutations

methods:{

...mapMutations(['changeVal'])

}

3.actions 异步执行之后，触发mutations当中定义的函数同步改变state当中的值