Vue

第一天

自定义过滤

全局：

Vue.filter('name',function(数据,参数,参数...){})

局部：

new Vue({

filters:{

name:function(数据,参数,参数...){

}

}

})

自定义指令

全局：

简单定义：

Vue.directive(namem,function(dom,value,value...){})

生命周期：

Vue.directive(name,{

bind(){},//初始化调用

inserted(),//虚拟dom已存在

update(),//传入的数据发生改变

componentupdated(),//子级组件

unbind()//解绑

})

局部：

简单定义：

new Vue({

directives:{

name:function(dom,value,value...){

}

}

})

生命周期：

new Vue({

directives:{

name:{

bind(){},//初始化调用

insert(),//虚拟dom已存在

update(),//传入的数据发生改变

componentUpdate(),//子级组件

unbind()//解绑

}

}

})

定义组件：

<template id="name"></template>

Vue.component('names',{

template:'#name',

data(){

return {

}

}

})

调用：<names></names>

第二天

插槽：

slot

定义插槽： <slot name="default"></slot>

插槽插入内容： <template v-slot:default>内容</template>

<div slot="no1">内容</div>

slot内可包含任何模板代码，包括 HTML，其它的组件

组件传参：

自定义属性等于data中数据

<name :自定义属性='数据'></name>

Vue.component('dLee', {

template: '#d',

props:['imgarr'],//用props接受 父级传参给子级时使用

data: function() {

return {

arr: this.imgarr,

}

}

})

自定义事件：

this.$emit('事件名称')//可以传参给父级

this.$on:

监听当前实例上的自定义事件。事件可以this.$emit触发

this.$on('test', function (msg) {

console.log(msg)

})

this.$emit('test', 'hi')

bus:

兄弟级的数据管理

var bus = new Vue()

bus.$on('自定义事件名称',(a){

//a为$emit('name',value)中的value

})

我问你 组件传参的三种方式

1.自定义属性绑定传参 父级传到子级 用props接收

2.$emit('事件名称',参数) 子级传参到父级

3.bus.$on('事件名称',(a){

a是$emit传过来的参数

})

路由：

第一步创建模板：var index = {template: '#index'}

第二步定义一个routes

var routes = [

{

path:'/',

component:index

}

]

第三步 定义一个router为VueRouter实例

var router = new VueRouter({

routes

})

第四步 将router放在vue实例中

new Vue({

el: "#box",

router

})

<router-view></router-view>

<router-link to="/"></router-link>/////或者用a标签页可以进行跳转

递归组件 ->只能是全局组件

递归组件/组件之间的循环引用： 树形结构

例子：省市县数据

二级路由：children:[]

var routes = [

{

path: '/',

component: no1,

children:[

{path:'no1\_1/',component:{template:'#no1\_1'}}

]

}

]

动态路由：

var routes = [

{

path: '/no1:uid',

component: no1,

}

]

<a :href="'#/>

{{i}}

</a>或者<router-link to="'/no1/'+i></router-link>

$route: 路由信息

$route.path:路径

$route.params：参数

动态路由匹配

动态路由：路径以冒号开头，this.$route.params

补充生命周期：

activated:keep-alive 组件激活时调用。

deactivated:keep-alive 组件停用时调用。

内置组件：

<keep-alive></keep-alive>

用来对组件进行缓存,可以保留它的状态或避免重新渲染

<component></component>//组件

<slot></slot>插槽

<router-view></router-view>

<router-link to="/"></router-link>

<template></template>

props验证:

type 支持类型检测：String、Number、Boolean、Function、Object、Array、Date、Symbol、自定义的构造函数

动态组件：

:is

什么是动态组件：

就是几个组件放在一个挂载点下，然后根据父组件的某个变量来决定显示哪个，或者都不显示。

动态切换：（:is后面的是对应的组件名，必须一致）

在挂载点使用component标签，然后使用v-bind:is=”组件名”，会自动去找匹配的组件名，如果没有，则不显示；

改变挂载的组件，只需要修改is指令的值即可

//动态组件没有课件不会整理 晚上课件下来你自己再补充补充吧

第三天

作业

第四天

Git：

创建管理

git init

加入版本管理

git add 文件名

git add .

添加到版本库

git commit -m ‘备注’

查看修改过的

git status

添加到远程仓库

git push 地址 分支名

在本地声明一个远程地址的名字

git remote add 名字 地址

克隆远程

git clone 地址

第五天

动画

隐藏到显示前.lee-enter

隐藏到显示时.lee-enter-active

隐藏到显示后.lee-enter-to

显示到隐藏前.lee-leave

显示到隐藏时.lee-leave-active

webpack

cnpm i webpack webpack-cli webpack-dev-server -g

webpack 4

开发模式

development

生产模式

Production

记住带注释的 死记



cnpm i vue-cli -g

vue init webpack-simple app

npm run dev

npm run build

打包

代理路径

indexhtml引入

图片引入需要注意

$nextTick 当数据改变时，页面渲染时，触发

有一个变量a 变量b等于变量a

变量a发生改变 变量b跟着改变 浅拷贝

变量a发生变化 变量b不发生变化 深拷贝

Deep 深度监听

state 跟data一个概念

getter 跟计算属性是一个概念

mutation 修改state的唯一途径，对内不对外的函数

action 跟前台的methods是同理，对外，不对内

module 模块

获取数据module-> getter->state

修改数据 module->action->mutation->state

**后期一直写项目，开发app软件 团队合作写项目**

vue的课程

1：指令 13个

2：生命周期 10个

3：计算属性 与事件函数的区别

4：侦听属性

5：vue-resource get post jsonp

6: 自定义过滤

7：自定义指令 全局 局部

8：组件

组件定义

传值 父级到子级

子级到父级

兄弟级

Bus=本地存储=登陆页面

插槽

动态组件

：ist

递归组件

9：路由 四步

动态路由 ：/：？

上一级 下一级 传值

10：动画 6个

11：git 上传作业

12:webpack 两种打包模式 各个参数

13:vue-cli 初 中 高

14:element-uli

15:minui\_ul

16:vue X