文本插值

{{内容}}

内容支持js表达式，但只支持单个表达式

v-html=”内容”

输出html指令

v-blind：属性名=”属性值”

标签属性输入

\*属性值为布尔型时，fasle，null,undefined为不显示

\*class属性的特别用法，v-blind:class=”{属性内容：布尔类型判断class是否存在，....}”

v-if=”布尔类型”

条件渲染，为true时才会显示该模板

v-else

v-else-if=”布尔类型”

\*当条件渲染的组件为多个时，可以用template将组件进行包裹

\*vue会复用相同元素，通过添加key属性让该元素独立不复用

v-for=”(item ,index )in items”

列表渲染

\*也可以用于遍历对象的所有属性v-for=”(value,name) in object”

\*也可以用template标签包裹其他标签

v-on:事件=”绑定的方法”

事件监听

v-model:[参数名] 双向绑定

v-bind的缩写:

v-on的缩写@

Vue.component(`组件名`,{

data:function(){

},

template:”模板”

}

)

props定义接受属性，可以拿到组件中的属性值

data定义属性

methods定义函数方法

template定义模板，最外层必须要被标签包裹

局部注册组件

在根组件上声明

components:{

组件名:{

template:””,

methods:””,

data:””

}

}

实例生命周期：

options即使new Vue({})中传入的参数

里面包括el,data,method,等数据

created在实例被创建之后执行

mounted钩子函数

data就是注册组件的容器，数据注册到data里面，在展示在页面

method就是方法：在dom中需要{{method()}}区调用

computed计算属性:计算属性可以直接当成数据去使用

计算属性中实际有set方法和get方法，平时写的函数return实际就是get方法的简写，一般不写set方法

computed是缓存，method是方法

watch 监听data里的数据，如果被监听的数据发生改变，就会触发函数watch:{data:function(newdata){}}，尽量不要用太多，性能会下降

init

beforeCreate

init

created

处理el

处理template

beforMount

mounted

beforeupdate

更新dom

updated

beforeDestroy

destroy

组件：

创建组件构造器，

接受对象：

component

{

template：模板,

data:

methods:

........

}

注册组件，

全局组件，被注册的组件可以在多个Vue实例中使用

Vue.component()

接受参数，组件名，组件对象

局部组件，在Vue中options定义components属性添加对象

组件中可以继续定义组件，components属性

components: {  
 atest: {  
 template: "#t"  
 },  
 itest: {  
 template: `<div> </div>`  
 }  
}

Vue实例可以看做一个根组件root

父子组件，谁注册谁管理

语法糖：可以不写extend创建组件，直接传对象进入注册器

分离模板：

写一个template模板定义id属性，把id传入到Vue的options中的

components:{

“组件名”：{template:”#组件id”}

}

在Vue实例中，如果同时存在el和template属性，vue会将el绑定的模块替换为template

es6

以后不要在使用var，只有函数有作用域

let变量需要改变时使用，

const常量，

开发时优先使用const，只有在需要改标志符时才使用let

语法糖：在对象中a:a 可以简写为a

前端模块化：

将自己的函数方法闭包防止与别人的模块命名冲突

早期将自己的函数方法以全局对象的形式return出去，在由其他人引用全局属性，解决代码复用问题

常见的模块化规范

CommomJs,AMD,CMD,ES6

引入javascript的时候，可以添加type = ‘module’，表示引入的js是模块化的文件

es6的导入导出

在script上添加type=module是js成为模块，

export {}导出

import {变量名} from “路径”导入，ES6对象解构语法

export default 的方式可以在导入时不用命名，直接导入使用，一个模块只能有一个default

import \* as 别名 from “路径” 导入全部的导出，使用 别名.属性 的方式调用

commonjs

module.exports = {方法名}导出文件，在对象中的所有方法函数都会被导出

const {方法名...} = require(“路径”)导入文件，

使用变量保存也是一种语法，它会将被导入对象中的属性自动赋值到保存变量中，ES6语法

这前样只要webpack打包后就会自动生成一个新的js文件，这样就可以使用依赖了

es6模块化

export const 变量名

import {变量名字} from “地址”

webpack：

模块化和打包工具，

可以让浏览器不支持的格式被解析成浏览器可以使用的格式

依赖nodejs

npm包管理工具，node packages manager

npm全局安装和局部安装。一个生效整个系统，一个专门在某个应用中使用

webpack.config.js是webpack的配置文件的名字，在运行webpack的时候会自动调用

导入const path = reuire(‘path’)，path是node的全局属性，path模块path.resolve(\_\_dirname,”dist”),把webpack的路径和””中的字符串拼接

\_\_dirname也是node的全局变量，保存webpack.config.js的路径

npm init初始化时会自动生成一些文件

package.json文件，负责告诉当前项目的一些信息,如果改文件中有依赖，还需要npm install导入依赖

dist是项目打包后的存储目录

src原文件目录

assets资源文件夹，包括图片,css

commponents组件，这里放入的公共组件，小组件

common公共组件，可以给其他项目使用的组件

conntent和当前业务相关的组件

common公共js,负责抽取一些公用的常量

views视图，负责首页，购物车等大的视图组件

Vue.js

通过模块化引入vue.js

npm install vue

import Vue from ‘vue’

这里没有指定路径，是当没有路径时，webpack就会从node\_modules中去寻找模块

所有的Vue插件都需要使用Vue.use()将插件导入，然后在主函当中导入配置js

router会向每一个组件注册$router

$router.push()就可以在代码中进行路由跳转

$router.replace()无痕浏览

动态路由

在js中配置

/user/:userId

就可以拦截所有的user了

$route当前处于活跃的路由

this.route.params.userId就可以拿到user的Id

组件之间的通讯

父组件向子组件中传递信息，props可以获取子组件标签上的参数，而父组件可以给子组件标签赋值

props:[‘属性名’]或者props:{属性名:类型}或者props:{属性名:{type:属性类型,default:默认值}}

子组件向父组件中传递信息，自定义事件$emit，this.$emit(‘事件名字’,{事件数据})定义一个父组件事件，通过父组件的@’事件名字’来监听子组件的事件

总结就是，父传子，子获取标签属性。子传父，父监听子的自定义事件

如果使用浏览器的事件，则默认接受event参数

而子组件自定义事件，则默认传自定义的数据

$refs会拿到所有定义了ref=”名字”属性的组件，$refs.名字 的方式拿到对象

this.$children获得子组件数组，一般不常用

$parent

加强for循环,for(let a of list)

插槽：

插槽的基本使用

插槽内可以定义默认值

插槽内有多个值，会一起作为替换元素

具名插槽：给插槽定义名字，然后在替换元素中使用name属性，与插槽绑定

Promise异步开发的解决方案es6的

网络请求🡪返回数据后调用回调函数

有事会出现非常复杂的回调函数调用回调函数的情况，简称套娃

new Promise((resolve,reject)=>{

setTimeout (()=>{

setTimeout (()=>{

setTimeout (()=>{

})

})

})

})

优化后

new Promise((resolve,reject)=>{

setTimeout (()=>{

resolve(data)

reject(err)

},1000)

}).then((data)=>{

setTimeout ((res)=>{

resolve(res)

},1000)

return new Promise((res) =>{})

简写 return Promise.resolve(res)

继续简写 return res

}).catch((err)

)

Promise.all([]).then(results=>{

results[0]第一个请求的结果

results[1]第二个请求的结果

})当数组里的请求都完成，才会执行调用then

VueX

在store/index.js的文件中导入Vue和Vuex,Vue.use(Vuex)

export default new Vuex.Store(

{

state:{},变量

mutations:{}方法

getters:{}属性计算

actions:{}异步操作

modules:{}模块划分

}

)

通过this.$store.state.变量 和this.$store.commit(“方法名”)调用变量和方法

单一状态树：就是一个vue项目中只能有一个store别整多个

axios

axios({

url:

method:

}).then(res=>{

天然集成Promise

})

axios.all([])和Promise的效果一样

CSS

@import是在style标签内引用css的格式

:root{

--变量名:内容

}

伪类，用于获取根元素，获取的根元素就是html

--名：内容用于定义变量 其他地方就可以用var(--变量名，来使用这个变量)