# 第六周总结：

## 大致总结

本周继续学习Vue,在上周Vue基础语法的基础上开始学习组件化编码。

学习了两周的Vue,对Vue也是又爱又恨。Vue其简洁性与方便性注定会使得其受到很多前后端人员的青睐，但是又因其简洁，使得在前期的接触过程当中，需要花费大量的精力来理解各个对象（组件、js、router等）之间错综复杂的关系，例如：它们是如何通信，如何嵌套，数据对象等该放在哪个组件哪个位置，何时该使用set/get来监视对象。

另外，在学习时，涉及到大量ES6的知识点，如：箭头函数、数组的操作函数(filter、reduce、map等)。所以有必要对ES6的语法以及API进行系统的学习，扎实基础。

进度比预期的要慢一些，但坚信，前期‘慢’的重要性，相信会在后期的学习以及以后的工作中体现出来。

## 学习笔记

包含Vue学习笔记（基础语法(续上周)、组件化编码、ajax）、String学习笔记

### (一) Vue学习笔记——基础篇(续上周)

\*\*\*\*\*\*vue简单介绍\*\*\*\*\*\*

\*\*Vue基础篇\*\*

八、事件处理

1. 绑定监听:

v-on:xxx="fun"

@xxx="fun"

@xxx="fun(参数)"

默认事件形参: event

隐含属性对象: $event

示例说明:

view:

<button @click="test3($event)" >hello</button>

vm：

test3(event){

alert("test3"+event.target.textContent)//获取标签文本

}

2. 事件修饰符:

.prevent : 阻止事件的默认行为 event.preventDefault()

//阻止跳转到百度而执行test事件

<a href="http://www.baidu.com" @click.prevent="test">百度一下</a>

.stop : 停止事件冒泡 event.stopPropagation()

示例说明:

//如果不加入stop,触发内嵌div会先后执行test2、test1事件

<div style="width: 200px;height: 200px;background: red" @click="test1">

<div style="width: 100px;height: 100px;background: blue" @click.stop="test2"></div>

</div>

3. 按键修饰符

.keycode : 操作的是某个keycode值的健

.enter : 操作的是enter键

<input type="text" @keyup.13="test">//按下enter键触发事件

或<input type="text" @keyup.enter="test">

九.表单输入绑定

1.使用v-model(双向数据绑定)自动收集数据

text/textarea

checkbox

radio

select

示例说明：

view:

<div id="demo">

<form @submit.prevent="submit">

<span>性别: </span>

<input type="radio" id="female" value="女" v-model="sex">

<label for="female">女</label>

<input type="radio" id="male" value="男" v-model="sex">

<label for="male">男</label><br>

<input type="checkbox" v-model="likes" value="篮球">

<label>篮球</label>

<input type="checkbox" v-model="likes" value="足球">

<label>足球</label>

<input type="checkbox" v-model="likes" value="乒乓球">

<label>乒乓球</label>

<select v-model="cityId">

<option value="">未选择</option>

//注意此处value书写时，为了获取到id值，需要从datas中取，则需要写成':value'

<option :value="city.id" v-for="(city,index) in cities" :key="index">{{city.name}}</option>

</select>

<input type="submit" value="提交">

</form>

</div>

VM书写：

new Vue({

el:'#demo',

data:{

likes:[],

cities:[

{id:1,name:'武汉'},

{id:2,name:'成都'},

{id:3,name:'重庆'},

{id:4,name:'云南'}

],

cityId:''

}

)}

十、Vue生命周期

1. vue对象的生命周期

1). 初始化显示

\* beforeCreate()

\* created()

\* beforeMount()

\* mounted()

2). 更新状态

\* beforeUpdate()

\* updated()

3). 销毁vue实例: vm.$destory()

\* beforeDestory()

\* destoryed()

2. 常用的生命周期方法

created()/mounted(): 发送ajax请求, 启动定时器等异步任务

beforeDestory(): 做收尾工作, 如: 清除定时器

十一、过渡&动画

1. vue动画的理解

操作css的trasition或animation

vue会给目标元素添加/移除特定的class

2. 基本过渡动画的编码

1). 在目标元素外包裹<transition name="xxx">

2). 定义class样式

1>. 指定过渡样式: transition

2>. 指定隐藏时的样式: opacity/其它

3. 过渡的类名

xxx-enter-active: 指定显示的transition

xxx-leave-active: 指定隐藏的transition

xxx-enter: 指定隐藏时的样式

十二、过滤器

1. 理解过滤器

功能: 对要显示的数据进行特定格式化后再显示

注意: 并没有改变原本的数据, 可是产生新的对应的数据

2. 编码

1). 定义过滤器

Vue.filter(filterName, function(value[,arg1,arg2,...]){

// 进行一定的数据处理

return newValue

})

2). 使用过滤器

//mydata：需要过滤的对象

<div>{{myData | filterName}}</div>

<div>{{myData | filterName(arg)}}</div>

示例说明：

显示格式化的日期时间

1）引入时间格式化库moment.js

<script type="text/javascript" src="https://cdn.bootcss.com/moment.js/2.22.1/moment.js"></script>

2)定义过滤器

<script>

// 定义过滤器

Vue.filter('dateString', function (value, format='YYYY-MM-DD HH:mm:ss') {

return moment(value).format(format);

})

new Vue({

el: '#test',

data: {

time: new Date()

}

)}

<script>

3)调用过滤器

<p>最完整的: {{time | dateString}}</p>//输出：2018-8-21 21:03:32

<p>年月日: {{time | dateString('YYYY-MM-DD')}}</p>//输出：2018-8-21

十三、指令directive

1.常用内置指令

v-text : 更新元素的 textContent

v-html : 更新元素的 innerHTML

v-if : 如果为true, 当前标签才会输出到页面

v-else: 如果为false, 当前标签才会输出到页面

v-show : 通过控制display样式来控制显示/隐藏

v-for : 遍历数组/对象

v-on : 绑定事件监听, 一般简写为@

v-bind : 强制绑定解析表达式, 可以省略v-bind

v-model : 双向数据绑定

ref : 为某个元素注册一个唯一标识, vue对象通过$refs属性访问这个元素对象

<p ref="msg">abcd</p>

获取：this.$refs.msg.innerHTML

v-cloak : 使用它防止闪现表达式, 与css配合: [v-cloak] { display: none }

2.自定义指令

1). 注册全局指令

写在Vue实例外，即被<script></script>包含

//el：指令属性所在标签对象

//binding:包含指令相关信息数据的对象

Vue.directive('my-directive', function(el, binding){

el.innerHTML = binding.value.toupperCase()

})

2). 注册局部指令

写在Vue实例内

directives : {

'my-directive' : {

bind (el, binding) {

el.innerHTML = binding.value.toupperCase()

}

}

}

3). 使用指令

v-my-directive='xxx'

十五、插件

声明使用插件(安装插件: 调用插件的install())

Vue.use(MyPlugin) // 内部会调用插件对象的install() QueryVo非空

过滤掉空值和null

### （二）Vue学习笔记——组件化编码篇

\*\*\*\*\*\*vue简单介绍\*\*\*\*\*\*

\*\*Vue组件化编码篇\*\*

一、vue脚手架

用来创建vue项目的工具包

创建项目:

npm install -g vue-cli

vue init webpack VueDemo

开发环境运行:

cd VueDemo

npm install(安装依赖)

npm run dev

生产环境打包发布

npm run build

npm install -g serve

serve dist

http://localhost:5000

二、eslint

用来做项目编码规范检查的工具

基本原理: 定义了很多规则, 检查项目的代码一旦发现违背了某个规则就输出相应的提示信息

有相应的配置, 可定制检查

三、组件化编程说明

1.vue文件包含3个部分

<template>

<div></div>

</template>

<script>

export default {

props: []/{}//声明接收属性：该属性会成为组件对象的属性

data(){return{}},

computed: {}

methods: {},

watch: {}

filters: {}

directives: {}

components: {}

}

</script>

<style>

</style>

2.组件使用

基本使用

1) 引入组件

2) 映射成标签

3) 使用组件标签

<template>

<!--3.使用组件标签-->

<HelloWorld></HelloWorld>

<hello-world></hello-world>

</template>

<script>

//1.引入组件

import HelloWorld from './components/HelloWorld'

export default {

//2.映射成标签

components: {

HelloWorld

}

}

</script>

3.组件化编码的基本流程

1). 拆分界面, 抽取组件

2). 编写静态组件

3). 编写动态组件

css同对应的标签一起（拆分css）

a)初始化数据

b)动态显示初始化界面

c)实现与用户交互功能

四、组件间通信

1.组件间通信基本原则

1) 不要在子组件中直接修改父组件的状态数据(跨组件更新数据)

2) 数据在哪, 更新数据的行为(函数)就应该定义在哪()

2.组件通信的5种方式

props

vue的自定义事件

pubsub第三方库

slot

vuex

3.props:

父子组件间通信的基本方式

属性值的2大类型:

一般: 父组件-->子组件

函数: 子组件-->父组件

隔层组件间传递: 必须逐层传递(麻烦)

兄弟组件间: 必须借助父组件(麻烦)

4.vue自定义事件

子组件与父组件的通信方式

用来取代function props

不适合隔层组件和兄弟组件间的通信

5.pubsub第三方库(消息订阅与发布)

适合于任何关系的组件间通信

6.slot

通信是带数据的标签

注意: 标签是在父组件中解析

7.vuex

多组件共享状态(数据的管理)

组件间的关系也没有限制

功能比pubsub强大, 更适用于vue项目

示例说明：

1.List.vue获取App.vue的数组comments/方法deleteComment

a).在App.vue传递：

<List :comments="comments" :deleteComment="deleteComment" />//为了方便,注意名称需相同

b).在List.vue接收:

props:{

comments:{//接受的对象作为该组件的属性，可以直接使用

type:Object,

required:true

},

deleteComment:{

type:Function,

required:true

}

}

此处props可以简写：

方式一：props:['comments','deleteComment']

方式二：props:{

comments:Object,

deleteComment:Function

}

五、深度监视——localStorage与watch

1.将数据实时更新存储到文件 随时可调用

2.示例说明：对数组todos进行深度监视

a).从localStorage读取数组todos

data(){

return{

//window.localStorage.getItem("todo\_key")获取的是字符串，需要将其转换为JSON格式

todos:JSON.parse(window.localStorage.getItem("todo\_key")||'[]')

}

}

b).监视——将数据实时更新到localStorage

watch:{//监视

todos:{

deep:true,

handler:function (value) {//此处vlaue为数组格式，存储时需转换为json

window.localStorage.setItem('todo\_key',JSON.stringify(value))

}

}

}

3.需要注意的是：

a)window.localStorage.getItem("todo\_key")获取的是字符串，需要将其转换为JSON格式，才能符合Vue格式要求

JSON.parse(字符串)

b)存取皆为JSON格式

c)将数组转换为json格式：JSON.stringify(value)

### （三）Vue学习笔记——ajax篇

\*\*\*\*\*\*vue简单介绍\*\*\*\*\*\*

\*\*Vue-ajax篇\*\*

一、ajax

a)相关库:

vue-resource: vue插件, 多用于vue1.x

axios: 第三方库, 多用于vue2.x

b)vue-resource使用

// 引入模块

import VueResource from 'vue-resource'

// 使用插件

Vue.use(VueResource)

// 通过vue/组件对象发送ajax请求

this.$http.get('/someUrl').then((response) => {

// success callback

console.log(response.data) //返回结果数据

}, (response) => {

// error callback

console.log(response.statusText) //错误信息

})

c)axios使用

// 引入模块

import axios from 'axios'

// 发送ajax请求

axios.get(url)

.then(response => {

console.log(response.data) // 得到返回结果数据

})

.catch(error => {

console.log(error.message)

})

### （四）String StringBuffer StringBuilder 学习笔记

\*\*\*\*String StringBuffer StringBuilder 学习笔记\*\*\*\*

一、对String的基本理解

基本概念理解：

字符串数据都是一个对象

字符串数据一旦初始化就不可以被改变了

字符串对象都存储于常量池中，字符串常量池。

示例说明：

String str1 = "abc";

String str2 = "abc";

String str3 = new String("abc");

问题说明：

1.为什么str1==str2结果为true呢？

先会在内存里创建str1以及其指向的内容"abc",当再次创建str2时，会在常量池里找是否有"abc"，

有就不会再开辟内存空间，str2直接指向已有的"abc",也就是str2和str1指向了同一个对象。

2为什么str1==str3结果为false呢？

==比较的地址值，而new是重新开辟的。

3.为什么str1.equals(str3)结果为true呢？

String类复写了Object类的equals方法，比较的是字符串中的内容是否相同。

二、String重要API

equals(Object obj)字符串类重写了Object的方法,用于比较字符串的值是否相等。

isEmpty()判断字符串是否为空。

int length()获取字符串的长度。

int indexOf(String str)返回指定子字符串在此字符串中第一次出现处的索引。

String substring(int beginIndex)返回一个新的字符串，它是此字符串的一个子字符串。

String toUpperCase() 所有字符都转换为大写 。

String toLowerCase() 所有字符都转换为小写。

String replace(String oldString, String newString)返回一个新的字符串，它是通过用 newString 替换此字符串中出现的所有 oldString 得到的。

trim() 去除字符串的前后空格。

int compareTo(String anotherString) 按字典顺序比较两个字符串。

三、StringBuffer&StringBuilder

1.字符串String是定长的，StringBuffer是变长的.

注意：使用final关键字修饰一个变量时，是指引用变量不能变，引用变量所指向的对象中的内容还是可以改变的

2.常用API

1)增: StringBuilder append(String str);在当前字符串末尾追加给定字符串。

2)删: StringBuilder delete(int start,int end);

删除当前字符串中给定范围内的字符串。

StringBuilder deleteCharAt(index);删除当前字符串中给定下标的字符。

3)改: StringBuilder replace(int start,int end,String str);

将当前字符串中给定范围内的字符串替换为给定的字符串。

4)插: StringBuilder insert(int index,String str);

将给定的字符串内容插入到指定位置处。

5)倒置: StringBuilder reverse();将当前字符串内容反转

3.StringBuffer,StringBuilder区别

StringBuffer 线程安全的,效率低(同步)

StringBuilder 线程不安全的,效率高(不同步)