复习

1. 介绍框架和库的区别（框架处理数据）

2. 安装方式： script标签引入vue.js; 通过vue-cli

3. 插值表达式 {{ expression }}

4. 指令（特点） <div v-名称></div>

① 选择指令 v-if="boolean" 是否要渲染当前的标签

v-else-if="boolean" 是否渲染当前标签

v-else 默认的行为

② 循环指令 v-for="tmp in arr"

v-for="(tmp,index) in arr"

③ 事件指令 v-on:click="handleClick(1)"

new Vue({

methods: {

handleClick: function(num) { }

}

})

@click="handleClick"

④ 属性绑定 v-bind:src="expression"

v-style:style="{color: myColor, fontSize: mySZ}"

v-style:class="{box: true, myBg:false}"

:src

⑤ 其它的指令 v-html v-text (数据中html标签的解析情况，v-html会正确渲染成html标签，v-text渲染成普通的文本)；情况：会清空标签之前的内容

v-show 都会渲染，只是false会添加display： 和 v-if 选择性的渲染

⑥ v-model 使用范围： 在表单中；

data: { msg: 123 }

<input type="text" v-model="msg">

5. 组件： 是html标签的扩展，封装html标签

Vue.component('my-com', {

template: `

<div>

</div>

`

});

<my-com></my-com>

1. 自定义指令

① 创建

Vue.directive('change', {

bind: function(el,bindings){

//第一次绑定时候，执行

// el 绑定的dom元素

// bindings 指令的相关信息，

获取c参数bindinigs.value

}，

update: function (el,bindings) {

//在数据更新的时候执行

},

inserted: function () {

//完全挂载到页面上执行

}

})

②调用

<p v-change="'blue'"></p>

练习：

添加一个指令 focus, 通过这个指令可以让输入框获取光标 el.focus()

③创建局部指令

new Vue({

directives: {

'color': {

}

}

})

使用说明：局部指令只能在当前vue实例中使用。

1. 过滤器

作用：是用于将数据进行筛选，转换成我们想要的格式。

① Vue.filter('myfilter', function(myInput,arg){

return myInput.toUpperCase();

//参数就就表示要过滤的数据

// arg就表示过滤器调用时的实参

})

② {{msg | myfilter(false)}}

③ 局部过滤器

new Vue({

filters: {

'myfilter': function(){ }

}

})

练习1：

1--过滤成男孩 0--过滤成女孩

练习2：

对一组英文进行过滤，传递的参数是true转成大写，false转成小写

1. 组件

Vue.component('my-com', {

template: `<div></div>`

});

练习： 创建一个留言板组件

①创建局部组件

new Vue({

components:{

'my-com': { template: '....' }

}

})

注意事项：局部组件只能在局部使用，不能在全局组件中使用。组件渲染和template有关系的。

1. 生命周期

在Vue 实例中，或者Vue组件中，都有自己的生命周期。

beforeCreate 基本的初始化，不会初始化data中数据

created 初始化data中的数据（如果使用，可以在这个位置）

beforeMount 数据渲染到视图之前

mounted 数据已经渲染（挂载）到了视图中

beforeUpdate/updated 数据更新前/数据更新后

beforeDestroy/destroyed 组件销毁前/组件销毁后

练习：

在一个组件中，创建一个h1标签；在组件挂载到页面后（mounted），让标签的透明度 0 ~ 1变化 +0.1

① 创建组件

② 在组件的mounted中，开启一个周期性定时器

③ 把0~1变化过程中的数据，绑定给h1的样式

:style="{opacity: myOpacity}"

16：45

1. 常用属性

回顾： 在组件中一共见到多少成员

template

data

methods

directives

filters

components

beforeCreate/created

beforeMount/mounted

beforeUpdate/updated

beforeDestroy/destroyed

1. watch 监听器，用于监听某个数据的变化

前提：结合着双向数据绑定

2. computed 计算属性，常用于一些复杂业务逻辑，带有缓存的功能，提高代码的执行效率。

如果数据没有变化，只会取一次。

注意事项：

在视图中使用的时候，直接写名称。

六、总结

1. 自定义指令（创建，调用）

Vue.directive('change', {

bind: function (el,bindings) {

bindings.value//参数获取

}

});

new Vue({ directives:{ } })

<p v-change="'red'"></p>

2. 过滤器

Vue.filter('myfilter', function(myInput,arg1){

return myInput......

});

new Vue({ filters: { } })

{{ msg | myfilter(1) }}

3. 组件

Vue.component('my-com', {

template: '<div></div>',

data: function () {

return { num:1 }

}

});

//复合组件

<my-com>

<my-tem></tem>.....

</my-com>

4. 声明周期函数

created 初始化了数据，就可以使用数据

mounted 把数据挂载到了页面上

5. watch 和 computed

练习： 见图示