2、2

输出hello world

<div id="app">{{ content }}</div>//差值表达式{{ ‘’ }};

//新建一个vue实例

var app = new Vue({

el: '#app', //el: ‘#app’ 负责接管id为app的div标签里的内容

data: {

content: 'hello world'

}

})

2、4

MVVM模式

Model、View、ViewModel

2、5

自定义组件：

全局注册组件

Vue.Component(‘component-a’, {

template: “<div></div>”

})

<div id="app">

<component-a></component-a>//作为一个标签使用

</div>

局部注册：

new Vue({  
 el: '#app'  
 components: {  
 'component-a': ComponentA, //驼峰式写法  
 'component-b': ComponentB  
 }  
})

2、3列表渲染

<ul>

<li v-for=”item in list”>{{ item }}</li>

</ul>

<script>

Var example = new Vue({

el: ‘#example’,

data: {

List: [

{‘abac’ },

{‘mnbacac’ },

]

}

})

</script>

//渲染为 1.abab

2.mnbacac

2、6组件实现遍历list

<div id="app-4">

<todo-item v-bind:content="item" v-for="item in todos"></todo-item>

</div>

Js:

var TodoItem = {

props:['content'], //子组件从父组件接受值

template: "<li>{{content}}</li>" //局部注册组件

}

var app4 = new Vue({

el:'#app-4',

components: {

TodoItem: TodoItem

},

data:{

todos:[

{ text:'学习javascript'},

{ text:'学习java'},

{ text:'学习script'}

]

}

})

//渲染结果

1. { "text": "学习javascript" }
2. { "text": "学习java" }
3. { "text": "学习script" }

2、7组件间传值

<todo-item v-bind:content="item"

v-bind:index="index"

v-for="(item,index) in todos"

@delete="handleItemDelete">

<!-- 监听子组件delete事件 index表示数组下标 -->

</todo-item>

Js

var TodoItem = {

props:['content','index'],

template: "<li @click='handleItemClick'>{{content}}</li>",

methods: {

handleItemClick: function() {

this.$emit("delete",this.index) //向外触发事件,同时将index传给父组件

}

}

}

var app4 = new Vue({

el:'#app-4',

components: {

TodoItem: TodoItem

},

data:{

todos:[

{ text:'学习javascript'},

{ text:'学习java'},

{ text:'学习script'}

]

},

methods: {

handleItemDelete: function(index) { //接受从子组件传过来的参数

this.todos.splice(index,1) //从下标为index开始，删除一项

}

}

})

当点击列表项时会删除该项。