Vue

一个mvvm框架（库）、和angular类似

比较容易上手、小巧

MVVM

MVC

MVP

MV\*

MVX

官网

Cn.vue.js.org

手册

Vue和angular区别？

Vue-简单易学

-指令v-xxx

-一片html代码配合json,在new处理vue实例

-个人维护项目

-适合：移动端项目，小巧

-vue发展势头很猛，github上start数量超越angular

Angular-上手难

-指令以ng-xxx

-所有属性和方法挂到$scope身上

-angular由google维护

-适合pc端项目

共同点：不兼容低版本IE

Vue基本雏形

Angular展示一条基本数据

Var app=augular.module(“app”,[])

App.controller(“xxx”,function(#scope){

$scope.msg=’welcom’;})

Html::

Div ng-controller=”xxx”

{{msg}}

Vue

Html:

<div id=”box”>{{msg}}</div>

Var c=new Vue({

El:”#box”, 选择器：id tagname class都可以

Data:{

Msg:’welcome vue”

}

})

常见指令：

Angular：ng-model ng-controller ng-repeat

指令：扩展html标签功能

v-model：一般表单元素input 数据双向绑定

循环

v-for=”name in arr”

{{$index}}

v-for=”name in json”

{{$index}} {{$key}}

v-for="(k,y) in json"

{{k}} {{y}} {{$key}}

事件

Angular ng-click

v-on:click=”函数”

v-on:click/mouseout/mouseover/dblclick/mousedown…

|  |
| --- |
| window.onload=function(){  var c=new Vue({  el:"#box",  data:{  msg:"welcome vue",  msg2:12,  msg3:true,  msg4:['aaa','bbb',"ccc"],  json:{a:"AAA",b:"BBB",c:"CCC"},  },  methods:{  show:function(){  console.log(11)  }  },  })  } |

显示隐藏

Angular ng-show

v-show=”true/false”

bootstrap + vue简易留言板todolist

bootstrap:css框架 跟jqueryMobile一样

只需要给标签赋class 角色

依赖jquery

|  |
| --- |
| <!DOCTYPE html>  <html lang="">  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1">  <title>Title Page</title>  <link rel="stylesheet" href="lib/bootstrap.css">  <script src="lib/jquery-1.7.2.js"></script>  <script src="lib/bootstrap.js"></script>  <script src="vue.js"></script>  <script>  window.onload=function(){  new Vue({  el:"#box",  data:{  myData:[    ],  username:'',  age:"",  nowIndex:-100,  },  methods:{  add:function(){  this.myData.push({  name:this.username,  age:this.age  })  this.username=this.age=""  },  deleteMsg:function(n){  if (n=-2) {  this.myData=[];  } else {  this.myData.splice(n,1)    }  }  },  })  }  </script>  </head>  <body>  <div class="container" id="box">  <form role="form">  <div class="form-group">  <label for="username">用户名</label>  <input type="text" v-model="username" placeholder="输入用户名" class="form-control" id="username">  </div>  <div class="form-group">  <label for="age">年 龄</label>  <input v-model="age" placeholder="输入年龄" type="text" class="form-control" id="age">  </div>  <div class="form-group">  <input type="button" class="btn btn-primary" value="添加" v-on:click="add()">  <input type="reset" class="btn btn-danger" value="重置">  </div>    </form>  <hr>  <table class="table table-bordered table-hover">  <caption class="h2 text-info text-center">用户信息表</caption>  <tr class="text-danger">  <th class="text-center">序号</th>  <th class="text-center">名字</th>  <th class="text-center">年龄</th>  <th class="text-center">操作</th>  </tr>  <tr class="text-center" v-for="item in myData">  <td>{{$index+1}}</td>  <td>{{item.name}}</td>  <td>{{item.age}}</td>  <td>  <button class="btn btn-primary btn-small" data-toggle="modal"  data-target="#layer" v-on:click="nowIndex=$index">删除</button>  </td>  </tr>    <tr v-show="myData.length!=0">  <td colspan="4" class="text-right">  <button class="btn btn-danger btn-sm" data-toggle="modal"  data-target="#layer" v-on:click="nowIndex=-2">删除全部</button>  </td>  </tr>  <tr v-show="myData.length==0">  <td colspan="4" class="text-center text-muted">  <p>暂无数据...</p>  </td>  </tr>  </table>  <div role="dialog" class="modal fade bs-example-modal-sm" id="layer">  <div class="modal-dialog">  <div class="modal-content">  <div class="modal-header">  <button class="close" data-dismiss="modal">  <span>&times;</span>  </button>  <h4 class="modal-title">  确认删除么？  </h4>  </div>  <div class="modal-body text-right">  <button class="btn btn-primary btn-sm " data-dismiss="modal" >取消</button>  <button class="btn btn-danger btn-sm" v-on:click="deleteMsg(nowIndex)" data-dismiss="modal">确认</button>  </div>  </div>  </div>  </div>  </div>  </body>  </html> |

01/02

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

事件深入

v-on:click/mouseover……

简写的：@click=””

事件对象：@click=”show($event)”

事件冒泡:阻止冒泡：a).ev.cancelBubble=true;

b)[.@click.stop](mailto:.@click.stop)

默认行为(默认事件)：

阻止默认行为：

a).ev.preventDefault();

b)[.@contextmenu.prevent](mailto:.@contextmenu.prevent)

键盘：

@keydown $event ev.keyCode

@keyup

常用键

回车 a)[.@keyup.13](mailto:.@keyup.13)

b)[.@keyup.enter](mailto:.@keyup.enter)

上、下、左、右

@keyup/keydown.left

@keyup/keydown.right

@keyup/keydown.up

@keyup/keydown.down

@keyup/keydown.delete

属性：

与指令是不一样的

v-bind:src=””

src width title …

简写 :src

<img src="{{url}}" alt=""> 效果能出来，但是会报错404

<img v-bind:src="url" alt="">效果相同不报错

Class和style

:class=” ” v-bind:class=” “

:style=” ” v-bind:style=” “

:class=”[red]” red是数据

:class=”[red,b,c,d]”

:clas=”{red:true,blue:false}”

:class=”json”

Data:{

Json:{red:true,blue:false}

}

Styel

:style=”[c]”

:style=”[c,d]”

注意：复合样式，采用驼峰命名格式

Style=”json”

模板：

{{msg}} 数据双向绑定 数据更新变化

{{\*msg}} 数据只绑定一次

{{{msg}}} html转译输出

过滤器—》过滤模板数据

系统提供了一些过滤器

{{msg|filterA}}

{{msg|filterA|filterB}}

Uppercase eg: {{'welcome'|uppercase}}

Lowercase

Capitalize 首字母大写（先小写后首字母大写）

Currency 钱

{{msg | filterA 空格 参数}}

交互

$http (ajax)

引入：vue-resource

Get

获取一个普通文件数据

this.$http.get('a.txt').then(

function(res){

console.log(res.data)

},function(res){

console.log(res.status)

})

给服务器发送数据

this.$http.get('get.php',{a:1,b:2}).then(

function(res){

// alert(res.status)

console.log(res.data)

// alert("22")

},function(res){

console.log(res.status)

})

Post

this.$http.post('post.php',

{a:1,b:2},

{emulateJSON:true

}).then(

function(res){

// alert(res.status)

console.log(res.data)

// alert("22")

},function(res){

console.log(res.status)

})

Jsonp

<https://sug.so.360.cn/suggest?callback=suggest_so&word=a>

https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su?wd=a

this.$http.jsonp('https://sp0.baidu.com/5a1Fazu8AA54nxGko9WTAnF6hhy/su',{

wd:'a'

},{

jsonp:'cb' b //callback名字，默认就是’callback’

}).then(

function(res){

// alert(res.status)

console.log(res.data)

// alert("22")

},function(res){

console.log(res.status)

})

02/01

微博

Vue->vue-resource ajax

服务器环境

This.$http.get()/post()/jsonp()

This.$http({

<url:地址>；

data:给后台提交的数据；

method:get/post/jsonp

jsonp:’cb’;})

Vue事件

@click

数据

添加一条留言

获取某一页数据

getPageData(3)

vue 生命周期 1.0版本

钩子函数

Created -🡪实例已经创建的时候 √

BeforeCompile ->实例创建完，编译之前。

Compiled –>编译后

Ready -🡪插入到文档中 √

销毁

BeforeDestroy 🡪销毁之前

Destroyed 🡪销毁之后

用户会看到花括号标记

V-cloak 防止闪烁 比较大的段落

<span>{{msg}}</span> -🡪 v-text

{{{msg}}} -> v-html

B随着a的值变化

Ng：$scope.$watch

计算属性的使用：computed：{

B:function(){

Return 值}

}

———————————

Computed:{

B:{

Get:

Set:

}

}

\*computed:里面可以放置一些业务逻辑代码，一定记得return

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  a => {{a}}  <br>  b => {{b}}  </div>  <script>  var vm=new Vue({  el:"#box",  data:{  a:1,  },  computed:{  b:function(){  //业务逻辑  return this.a+1;  }  }  })  document.onclick=function(){  vm.a=101;  }  </script>    </body>  </html> |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  a => {{a}}  <br>  b => {{b}}  </div>  <script>  var vm=new Vue({  el:"#box",  data:{  a:1,  },  computed:{  b:{  get:function(){  return this.a+2;  },  set:function(val){  this.a=val    }    }  }  })  document.onclick=function(){  //vm.a=101;  vm.b=101;  }  </script>    </body>  </html> |

Vue实例简单方法

Vm.$el -🡪就是元素

Vm.$data 🡪就是data

Vm.$mount 🡪手动挂载vue程序

Vm.$options 🡪获取自定义属性

Vm.$destroy 🡪销毁对象

Vm.$log() 🡪查看现在数据的状态

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  {{a}}  </div>  <script>  var vm=new Vue({  el:"#box",  data:{  a:1,  },  })  console.log(vm.$el)  vm.$el.style.background="red";  console.log(vm.$data.a)  </script>    </body>  </html> |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  {{a}}  </div>  <script>  // var vm=new Vue({  // data:{  // a:1,  // },  // })  // vm.$mount('#box')//手动挂载  //  var vm=new Vue({  data:{a:1}  }).$mount("#box")  </script>    </body>  </html> |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  {{a}}  {{aa}}  </div>  <script>  var vm=new Vue({  aa:11,//自定义属性  show:function(){alert(222)},  data:{  a:1,  },  }).$mount("#box")  console.log(vm.$el)  vm.$options.show()  console.log(vm.$options.aa)  </script>    </body>  </html> |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  {{a}}    </div>  <script>  var vm=new Vue({  data:{  a:1,  b:2,  },  }).$mount("#box")    console.log(vm.$log())  </script>    </body>  </html> |

循环

V-for=”value in data”

会有重复数据

Track –by=”索引” 提高循环性能

Track-by=”$index/uid”(不重复即可)

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  <input type="button" value="添加" @click="add">  <ul>  <li v-for="val in arr" track-by="$index">{{val}}</li>  </ul>    </div>  <script>  var vm=new Vue({  data:{  arr:['aaa','bbb','cc'],  },  methods:{  add:function(){  this.arr.push("ddd")  }  }  }).$mount("#box")      </script>    </body>  </html> |

过滤器

Capitalize uppercase currency

Debounce 配合事件，延迟执行

数据配合使用过滤器

LimitBy 现在几个

LimitBy 参数(取几个)

LimitBy 取几个 从哪开始

filterBy 过滤数据

filterBy ‘谁’ （可以用变量代替）

orderBy 排序

orderBy 1/-1 正序/倒序

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  <ul>  <li v-for="val in arr | limitBy 2 1">{{val}}</li>  </ul>  </div>  <script>  var vm=new Vue({  data:{  arr:[1,2,3,4,5]  },  methods:{    }  }).$mount("#box")      </script>    </body>  </html> |

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  <input type="text" @keyup="show | debounce 2000">  </div>  <script>  var vm=new Vue({  data:{    },  methods:{  show:function(){  alert(1)  }  }  }).$mount("#box")      </script>    </body>  </html> |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  <input type="text" v-model="a">  <ul>  <!-- <li v-for="val in arr | filterBy '1'">{{val}}</li> -->  <li v-for="val in arr | filterBy a">{{val}}</li>  </ul>  </div>  <script>  var vm=new Vue({  data:{  a:"",  arr:["11","23","13","34"]  },  methods:{    }  }).$mount("#box")      </script>    </body>  </html> |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  <input type="text" v-model="a">  <ul>  <!-- <li v-for="val in arr | filterBy '1'">{{val}}</li> -->  <li v-for="val in arr | orderBy -1">{{val}}</li>  </ul>  </div>  <script>  var vm=new Vue({  data:{  a:"",  arr:["11","23","13","34"]  },  methods:{    }  }).$mount("#box")      </script>    </body>  </html> |

自定义过滤器

Vue.filter('toDou',function(input){

return input<10?'0'+input:''+input;

})

时间过滤器

Vue.filter('date',function(input){

var oDate = new Date(input);

return oDate.getFullYear()+"-"

+(oDate.getMonth()+1)+"-"

+oDate.getDate()+" "

+oDate.getHours()+":"

+oDate.getMinutes()+":"

+oDate.getSeconds();

})

Html过滤器

双向过滤器

Vue.filter('filterHtml',{

read:function(input){

return input.replace(/<[^<+]>/g,'')

},//model->view;

write:function(val){

console.log(val)

}//view->model

})

数据——>视图

Model 🡪view

View 🡪model

v-text

v-for

v-html

指令

属性指令

Vue.directive(指令名称(‘red’)，function(参数){

This.el🡪原生Dom元素

})

<div v-red=”参数”></div>

指令名称：v-red 🡪red

注意：必须以v-开头

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>  </head>  <body>  <div id="box">  <span v-red>aadf</span>  </div>  <script>  Vue.directive('red',function(){  this.el.style.background='red';  })  var vm=new Vue({  data:{  msg:'a',    },  methods:{    }  }).$mount("#box")      </script>    </body>  </html> |

自定义指令---拖拽

|  |
| --- |
| <!DOCTYPE html>  <html>  <head>  <meta charset="utf-8" />  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <title>Page Title</title>  <meta name="viewport" content="width=device-width, initial-scale=1">  <script src="vue.js"></script>    </head>  <body>  <div id="box">  <div v-drag :style="{width:'100px',height:'100px',background:'blue',position:'absolute',right:0,top:0}">aadf</div>  <div v-drag :style="{width:'100px',height:'100px',background:'red',position:'absolute',left:0,top:0}">aadf</div>    </div>  <script>  Vue.directive('drag',function(){  var oDiv=this.el;  oDiv.onmousedown=function(el){  var disX=el.clientX-oDiv.offsetLeft;  var disY=el.clientY-oDiv.offsetTop;  document.onmousemove=function(ev){  var l=ev.clientX-disX;  var t=ev.clientY-disY;  oDiv.style.left=l+'px';  oDiv.style.top=t+'px';  }  document.onmouseup=function(){  document.onmousemove=null;  document.onmouseup=null;  }  }  })  var vm=new Vue({  data:{  msg:'a',    },  methods:{    }  }).$mount("#box")      </script>    </body>  </html> |

自定义元素指令(用处不大)

Vue.elementDirective('zns-red',{

bind:function(){

this.el.style.background='red';

}

})

@keydown.up

@keydown.enter

自定义键盘信息

@keydown.a/b/c…..

<input type="text" @keydown.ctrl='show'>

<input type="text" @keydown.myenter='show | debounce 2000'>

Vue.directive('on').keyCodes.ctrl=17;

Vue.directive('on').keyCodes.myenter=13;

监听数据变化

Vm.$el/$mount/$options/…..

Vm.$watch(name,fnCn);//浅度

Vm.$watch(name,fnCn,{deep:true});//深度

Vue组件：

Slot

Vue-loader

Vue-router

1.0

2.0

引入vue.js

Bower 前端 -🡪npm 后端

Npm install bower –g

验证 bower –version

Bower install <包名>

Bower uninstall <包名>

Bower info <包名> 查看包版本信息

# [ENOGIT git is not installed or not in the PATH](https://segmentfault.com/q/1010000010538990)

是因为你没有把git配置好，bower无法找到git，而bower是基于git去获取远程仓库中的包

Bower install [vue#1.0.28](mailto:vue@1.0.28)

Npm install [vue@1.0.28](mailto:vue@1.0.28)

<script src="bower\_components/vue/dist/vue.js"></script>

Vue🡪(过度)动画

本质走的css3:transition,animate

动画：

.fade-transition{

transition:1s all ease;

}

进入

.fade-enter{

opacity: 0;

}

离开

.fade-leave{

opacity: 0;

transform: translateX(200px)

}

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  <style>  #div1{  width: 100px;  height: 100px;  background: red;  }  .fade-transition{  transition:1s all ease;  }  .fade-enter{  opacity: 0;  }  .fade-leave{  opacity: 0;  transform: translateX(200px)  }  </style>  </head>  <body>  <div id="box">  <input type="button" value="按钮" @click="toggle">  <div id="div1" v-show="bSign" transition="fade"></div>  </div>  <script>  new Vue({  el:"#box",  data:{  bSign:true,  },  methods:{  // toggle:function(){  // alert(1)  // }  toggle(){  this.bSign=!this.bSign;  }  }  })  </script>  </body>  </html> |

Animate.css

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  <link rel="stylesheet" href="bower\_components/animate.css/animate.css">  <style>  #box{  width:400px;          margin: 0 auto;  }  #div1{  width: 100px;  height: 100px;  background: red;  }    </style>  </head>  <body>  <div id="box">  <input type="button" value="按钮" @click="toggle">  <div id="div1" class="animated" v-show="bSign" transition="bounce"></div>  </div>  <script>  new Vue({  el:"#box",  data:{  bSign:true,  },  methods:{  // toggle:function(){  // alert(1)  // }  toggle(){  this.bSign=!this.bSign;  }  },  transitions:{  bounce:{  enterClass:'zoomInLeft',  leaveClass:'zoomOutRight',  },  }  })  </script>  </body>  </html> |

Vue组件

组件：一个大的对象

注册或定义一个组件的方式有很多种

1. 全局组件

Var Aaa=Vue.extend({

Template:})

Vue.component(‘aaa’,Aaa)

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  <style>      </style>  </head>  <body>  <div id="box">  <aaa></aaa>  </div>  <script>    var Aaa= Vue.extend({  template:'<h3>我是标题h3</h3>'  })  Vue.component('aaa',Aaa)  var vm= new Vue({  el:"#box",  data:{  bSign:true,  },  methods:{    },    })  </script>  </body>  </html> |

组件里面放数据

Data必须是函数的形式，函数必须返回一个对象（json）

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  </head>  <body>  <div id="box">  <aaa></aaa>  </div>  <script>  var Aaa= Vue.extend({  template:'<h3 @click="change">{{msg}}</h3>',  data(){  return {msg:'我是标题'}  },  methods:{  change(){  this.msg="change"  }  }  })  Vue.component('aaa',Aaa)  var vm= new Vue({  el:"#box",  data:{  bSign:true,  },  methods:{    },    })  </script>  </body>  </html> |

1. 局部组件

放到某个组件内部

|  |
| --- |
| var Aaa=Vue.extend({  template:'<h3>{{msg}}</h3>',  data(){return {msg:'ddd'}}  })  var vm= new Vue({  el:"#box",  data:{  bSign:true,  },  methods:{  },  components:{  ‘aaa’:Aaa,  }  }) |

另一种编写方式：

|  |
| --- |
| Vue.component('my-aaa',{  template:'<strong>我是strong</strong>'  }) |

或

|  |
| --- |
| var vm= new Vue({  el:"#box",  components:{  'my-aaa':{  data(){  return {  msg:'welcome'  }  },  methods:{  change(){  this.msg="changed"  }  },  template:'<h2 @click="change">我是标题h2--{{msg}}</h2>'  }  },  }) |

配合模板：

1.

|  |
| --- |
| template:'<h2 @click="change">我是标题h2--{{msg}}</h2>' |

2.单独放到某个地方

a.

|  |
| --- |
| <div id="box">  <my-aaa></my-aaa>  </div>  <script type="x-template" id='aaa'>  <h2 @click="change">我是标题h2--{{msg}}</h2>  <ul>  <li>111</li>  <li>222</li>  <li>333</li>  </ul>  </script>  <script>  var vm= new Vue({  el:"#box",  components:{  'my-aaa':{  data(){  return {  msg:'welcome'  }  },  methods:{  change(){  this.msg="changed"  }  },  template:'#aaa',  }  },  })  </script> |

b.

|  |
| --- |
| <div id="box">  <my-aaa></my-aaa>  </div>  <template id="aaa">  <h1>标题1</h1>  <ul>  <li v-for='val in arr'>  {{val}}  </li>  </ul>  </template>  <script>  var vm= new Vue({  el:"#box",  components:{  'my-aaa':{  data(){  return {  msg:'welcome',  arr:['apple','banana','orange']  }  },  methods:{  change(){  this.msg="changed"  }  },  template:'#aaa',  }  },  })  </script> |

动态组件：

<component :is="a"></component>

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  </head>  <body>  <div id="box">  <input type="button" @click="a='aaa'" value="aaa组件">  <input type="button" @click="a='bbb'" value="bbb组件">  <component :is="a"></component>  </div>  <script>  var vm= new Vue({  el:"#box",  data:{  a:'aaa',  },  components:{  'aaa':{  template:'<h2>我是组件aaa</h2>'  },  'bbb':{  template:'<h2>我是组件bbb</h2>'  }  }      })  </script>  </body>  </html> |

Vue-devtools –调试工具

Vue 默认情况下，父组件也没法去访问父组件的数据

组件数据传递：

1. 子组件就想获取父组件data

在调用子组件：

<bbb :m=”数据”></bbb>

子组件之内：

Props:[‘m’,’myMsg’](注意：- 驼峰命名 )

Props:{

‘m’:String,

‘myMsg’:Number,}

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  </head>  <body>  <div id="box">  <aaa></aaa>  </div>  <template id="aaa">  <h1>11111</h1>  <bbb :m="msg" :my-msg="mgs1"></bbb>  </template>  <script>  var vm= new Vue({  el:"#box",  data:{  a:'aaa',  },  components:{  'aaa':{  data(){  return {  msg:'我是父组件的数据',  mgs1:"111",  }  },  template:'#aaa',  components:{  'bbb':{  props:['m','myMsg'],  template:'<h2>我是组件bbb-->{{m}}---{{myMsg}}</h2>'  }  }  },    }      })  </script>  </body>  </html> |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  </head>  <body>  <div id="box">  <aaa></aaa>  </div>  <template id="aaa">  <h1>11111</h1>  <bbb :m="msg" :my-msg="mgs1"></bbb>  </template>  <script>  var vm= new Vue({  el:"#box",  data:{  a:'aaa',  },  components:{  'aaa':{  data(){  return {  msg:'我是父组件的数据',  mgs1:111,  }  },  template:'#aaa',  components:{  'bbb':{  props:{'m':String,'myMsg':Number},  template:'<h2>我是组件bbb-->{{m}}---{{myMsg}}</h2>'  }  }  },    }      })  </script>  </body>  </html> |

1. 父组件获取子组件数据

子组件把自己的数据，发送给父组件

Vm.$emit(事件名称，数据)

V-on:接受数据 @事件名称

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  </head>  <body>  <div id="box">  <aaa></aaa>  </div>  <template id="aaa">  <span>我是父组件---{{msg}}</span>  <bbb @children-msg="get"></bbb>  </template>  <template id="bbb">  <h3>子组件</h3>  <input type="button" value="send" @click="send">  </template>  <script>  var vm= new Vue({  el:"#box",  data:{  a:'aaa',  },  components:{  'aaa':{  data(){  return {  msg:'我是父组件的数据',  mgs1:111,  }  },  template:'#aaa',  methods:{  get(msg){  this.msg=msg;  }  },  components:{  'bbb':{  data(){  return {a:'我是子组件数据'}  },  template:'#bbb',  methods:{  send(){  this.$emit('children-msg',this.a)  },  }  }  }  },    }      })  </script>  </body>  </html> |

方法二：

Vm.dispatch(事件名，数据) 子级向父级发送数据

Vm.$broadcast(事件名，数据) 父级向子级广播数据

配合event:{}

在vue2.0里面已经报废了

Slot

位置、槽口

作用：占位置

类型ng里面transclude(指令)

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  </head>  <body>  <div id="box">  <aaa>  <ul slot="ul-slot">  <li>11111111111</li>  <li>22222</li>  <li>3333</li>  </ul>  <ol slot="ol-slot">  <li>234</li>  <li>123</li>  <li>345</li>  </ol>  </aaa>  <hr>  <aaa></aaa>  </div>  <template id="aaa">  <h2>xxx</h2>  <slot name="ul-slot">这是默认情况</slot>  <p>welcome vue</p>  <slot name="ol-slot">这是默认情况</slot>  </template>  <script>  var vm= new Vue({  el:"#box",  data:{  a:'aaa',  },  components:{  aaa:{  template:"#aaa",  }  }  })  </script>  </body>  </html> |

Vue-spa应用，单页面应用

Vue-resource 交互

Vue-router 路由

根据不同url地址，出现不同效果

Bower info vue-router

0.7.13

Bower install vue-router#0.7.13

主页 home

新闻页 news

View

链接

<a v-link="{paht:'/home'}">新闻页</a>

展示内容

<router-view></router-view>

|  |
| --- |
| //1.准备一个根组件  var App=Vue.extend();  //2.Home News组件都准备  var Home=Vue.extend({  template:'<h3>我是主页</h3>'  })  var News=Vue.extend({  template:'<h3>我是新闻</h3>'  })  //3.准备路由  var router=new VueRouter();  //4.关联  router.map({  'home':{  component:Home  },  'news':{  component:News  }  })  //5.启动路由  router.start(App,'#box') |

跳转

Router.redirect({

‘/’:’home’

})

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  <script src="bower\_components/vue-router/dist/vue-router.js"></script>  </head>  <body>  <div id="box">  <ul>  <li>  <a v-link="{path:'/home'}">主页</a>  </li>  <li>  <a v-link="{path:'/news'}">新闻页</a>  </li>  </ul>  <div>  <router-view></router-view>  </div>  </div>  <script>  //1.准备一个根组件  var App=Vue.extend();  //2.Home News组件都准备  var Home=Vue.extend({  template:'<h3>我是主页</h3>'  })  var News=Vue.extend({  template:'<h3>我是新闻</h3>'  })  //3.准备路由  var router=new VueRouter();  //4.关联  router.map({  'home':{  component:Home  },  'news':{  component:News  }  })  //5.启动路由  router.start(App,'#box')  //6.跳转  router.redirect({  '/':'/home'  })  </script>  </body>  </html> |

路由嵌套（多层路由）

主页 home

登录 home/login

注册 home/regist

新闻页 news

|  |
| --- |
| subRoutes:{  'login':{  component:{  template:'<strong>我是登录信息</strong>'  }  },  'reg':{  component:{  template:'<strong>我是注册信息</strong>'  }  }  } |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  <script src="bower\_components/vue-router/dist/vue-router.js"></script>  </head>  <style>  .v-link-active{  font-size: :20px;  color:#f60;  }  </style>  <body>  <div id="box">  <ul>  <li>  <a v-link="{path:'/home'}">主页</a>  </li>  <li>  <a v-link="{path:'/news'}">新闻页</a>  </li>  </ul>  <div>  <router-view></router-view>  </div>  </div>  <template id="home">  <h3>我是主页</h3>  <div>  <a v-link="{path:'/home/login'}">登录</a>  <a v-link="{path:'/home/reg'}">注册</a>  </div>  <div>  <router-view></router-view>  </div>  </template>  <template id="news">  <h3>我是新闻</h3>  </template>  <script>  //1.准备一个根组件  var App=Vue.extend();  //2.Home News组件都准备  var Home=Vue.extend({  template:'#home'  })  var News=Vue.extend({  template:'#news'  })  //3.准备路由  var router=new VueRouter();  //4.关联  router.map({  'home':{  component:Home,  subRoutes:{  'login':{  component:{  template:'<strong>我是登录信息</strong>'  }  },  'reg':{  component:{  template:'<strong>我是注册信息</strong>'  }  }  }  },  'news':{  component:News  }  })  //5.启动路由  router.start(App,'#box')  //6.跳转  router.redirect({  '/':'/home'  })  </script>  </body>  </html> |

路由其他信息

/detail/:id/age/:age

{{$route.params |json}} -🡪当前参数

{{$route.path}} -🡪当前路径

{{$route.query|json}} –>当前数据

|  |
| --- |
| <!DOCTYPE html>  <html lang="en">  <head>  <meta charset="UTF-8">  <meta name="viewport" content="width=device-width, initial-scale=1.0">  <meta http-equiv="X-UA-Compatible" content="ie=edge">  <title>Document</title>  <script src="bower\_components/vue/dist/vue.js"></script>  <script src="bower\_components/vue-router/dist/vue-router.js"></script>  </head>  <style>  .v-link-active{  font-size: :20px;  color:#f60;  }  </style>  <body>  <div id="box">  <ul>  <li>  <a v-link="{path:'/home'}">主页</a>  </li>  <li>  <a v-link="{path:'/news'}">新闻页</a>  </li>  </ul>  <div>  <router-view></router-view>  </div>  </div>  <template id="home">  <h3>我是主页</h3>  <div>  <a v-link="{path:'/home/login'}">登录</a>  <a v-link="{path:'/home/reg'}">注册</a>  </div>  <div>  <router-view></router-view>  </div>  </template>  <template id="news">  <h3>我是新闻</h3>  <div>  <a v-link="{path:'/news/detail/001'}">新闻001</a>  <a v-link="{path:'/news/detail/002'}">新闻002</a>  </div>  <router-view></router-view>  </template>  <template id="detail">  {{$route.params |json}}  <br>  {{$route.path}}  <br>  {{$route.query|json}}  </template>  <script>  //1.准备一个根组件  var App=Vue.extend();  //2.Home News组件都准备  var Home=Vue.extend({  template:'#home'  })  var News=Vue.extend({  template:'#news'  })  var Detail=Vue.extend({  template:'#detail'  })  //3.准备路由  var router=new VueRouter();  //4.关联  router.map({  'home':{  component:Home,  subRoutes:{  'login':{  component:{  template:'<strong>我是登录信息</strong>'  }  },  'reg':{  component:{  template:'<strong>我是注册信息</strong>'  }  }  }  },  'news':{  component:News,  subRoutes:{  'detail/:id':{  component:Detail,  }  }  }  })  //5.启动路由  router.start(App,'#box')  //6.跳转  router.redirect({  '/':'/home'  })  </script>  </body>  </html> |

Vue-loader:

其他loader->css-loader、url-loader、html-loader

NodeJs-🡪require exports

Borserify 模块加载js

Webpack 模块加载器，一切东西都是模块，最后打包到一块

Require(‘style.css’) 🡪css-loader、 style-loader

Vue-loader基于webpack

.vue文件

放置的是vue组件代码

<template>html</template>

<style>css</style>

<script>js平时代码 es6 <script> babel-loader es6🡪es5

Vue-cli 脚手架

简单的目录结构

|-index.html

|-main.js 入口文件

|-App.vue vue文件 官方推荐命名法 大写

|-package.json 工程文件（项目依赖、名称、配置）

Npm init –yes 生成

|-webpack.config.js webpack配置文件

ES6:模块化开发

导出模块：export default{}

引入模块：import 模块名 from 地址

Webpack准备工作：

Npm install webpack 不带服务器

Npm install webpack-dev-server 带服务器

Webpack –p 压缩

Webpack -dev–server –inline –hot –port

npm install webpack-dev-server@ 3.1.14 --save-dev  
注意：

1.--save-dev 会将使用的版本添加到package.json 文

件中

2.Webpack 与webpack 的版本的兼容性

3.npm view vue-loader versions 查看版本

App.vue 🡪变成正常代码 🡪vue-loader@9.1.0

vue-html-loader、css-loader、vue-style-loader、[vue-hot-reload-api@1.3.2](mailto:vue-hot-reload-api@1.3.2)

babel-loader

babel-core

babel-plugin-transform-runtime

babel-preset-es2015

babel-runtime

最最核心

Vue

各种配置版本

"dependencies": {

"babel-core": "^6.26.3",

"babel-loader": "^6.3.0",

"babel-plugin-transform-runtime": "^6.23.0",

"babel-preset-es2015": "^6.24.1",

"babel-runtime": "^6.26.0",

"vue-hot-reload-api": "^1.3.2",

"vue-html-loader": "^1.2.4"

},

"devDependencies": {

"css-loader": "^0.25.0",

"vue": "^1.0.28",

"vue-loader": "^8.5.4",

"vue-style-loader": "^1.0.0",

"webpack": "^1.15.0",

"webpack-dev-server": "^1.6.0"

}

04

手动配置自己：

Webpack+vue-loader

Webpack 加载模块

如何运行此项目？

1. Npm install 或者cnpm install
2. Npm run dev

🡪package.json

“script”:{

“dev”:”webpack-dev-server –inline –hot –port 8082”

}

下载模块:npm install <package-name> --save-dev

EADDRINUSE 端口被占用

"devDependencies": {

"babel-core": "^6.17.0",

"babel-loader": "^6.2.5",

"babel-plugin-transform-runtime": "^6.15.0",

"babel-preset-es2015": "^6.16.0",

"babel-runtime": "^6.11.6",

"css-loader": "^0.25.0",

"vue-hot-reload-api": "^1.3.2",

"vue-html-loader": "^1.2.3",

"vue-loader": "^8.5.4",

"vue-style-loader": "^1.0.0",

"webpack": "^1.13.3",

"webpack-dev-server": "^1.16.2"

},

"dependencies": {

"vue": "^1.0.28"

}

.vue结尾，之后称呼组件

路由；

Vue-router

🡪如何查看版本

Bower info vue-router

路由使用版本：0.7.13

使用：

配合vue-loader使用

1. 下载vue-router模块 cnpm install [vue-router@0.7.13](mailto:vue-router@0.7.13) –save
2. import VueRouter from 'vue-router'
3. Vue.use(VueRouter);//
4. 配置路由

Var router=new VueRouter();

Router.map({

路由规则

})

1. 开启 router.start(App,’#app’);

注意：

之前 Index.html 🡪 <app></app>

现在 index.html 🡪 <div id=”app”></div>

App.vue 🡪需要一个 <div id=”app”></div> 根元素

路由嵌套：

和之前一模一样

上线：

Npm run build

-webpack –p

脚手架：

Vue-cli ---vue脚手架

帮你提供好基本的项目结构

本身集成很多项目模板

Simple 个人觉得没用

Webpack 可以使用 （大型项目使用）

Eslint检查代码规范

单元测试

Webpack-simple

个人推荐使用，没用代码检查

Browserify

Browserify-simple

基本使用流程

1. npm install vue-cli –g 安装vue命令环境

验证安装ok?

Vue --version

1. 生成项目模板

Vue init <模板名> 本地文件夹名称

$ vue init simple#1.0 vue-simple-demo

1. 进入到生成目录里面

Npm install

1. npm run dev

——————————————  
05

Vue2.0变化

Bower info vue

<http://vuejs.org/>

到了2.0以后有哪些变化

1. 在每个组件模板，不支持片段代码

组件中模板：

之前：

<template id="aaa">

<h4>这是组件</h4><strong>加粗</strong>

</template>

现在：必须有一个根元素包裹住所有的代码

<template id="aaa">

<div>

<h4>这是组件</h4><strong>加粗</strong>

</div>

</template>

1. 关于组件定义

Vue.extend 这种方式，在2.0里面有，但是有一些改动，这种写法，即使能用，也不用,废弃。

Vue.component(组件名称，{

Data(){},

Methods:{},

Template:

})

2.0推出一个组件，简洁定义方式：

Var Home={

Template: ->vue.extend()

}

1. 生命周期

之前：

Init

Created

beforeCompile

compiled

ready 使用较多 🡪mounted

beforeDestroy

destroyed

现在；

BeforeCreate 组件实例刚刚被创建，属性都没有

Created 实例已经创建完成，属性已经绑定

BeforeMount 模板编译之前

Mounted 模板编译之后，代替之前ready \*

beforeUpdate 组件更新之前

updated 组件更新之后 \*

beforeDestroy 组件销毁前

destroyed 组件销毁后

4.循环

2.0里面默认可以添加重复数据

Arr.forEach(function(item,index){

})

去掉了隐式变量

${{index}} ${{key}}

之前：v-for=”(index,val) in list”

现在：v-for=”(val,index) in list”

5.之前：track-by=”id”

现在：:key=”index”

<li v-for="(val,index) in list" track-by="index">{{val}}{{index}}</li>

6.自定义键盘指令

之前：

Vue.directive(‘on’).keyCodes.ctrl=17;

现在：

Vue.config.keyCodes.ctrl=17;

7.过滤器

之前：

系统就自带很多过滤器：

{{msg|currency}} {{msg|json}}…limitBy filterByll 。。。。。

现在：到了2.0，内置过滤器，全部删除了。

一些简单的功能，自己通过js实现

Lodash 工具库 \_.debouce(fn,200)

自定义过滤器——还有，但是自定义过滤器传参

之前：{{msg|toDou ‘12’ ‘5’}}

现在：{{msg|toDou (‘12’,’5’)}}

组件通信：

Vm.$emit()

Vm.$on()

父组件和子组件：

子组件想要拿到父组件的数据：

通过props

之前：子组件可以更改父组件信息，可以是同步的sync

现在：不允许之间改父级的数据，做赋值操作

问题，就想更改

A).父组件每次传一个对象给子组件,对象之间是引用的

B). 只是不报错，mounted中转

————————————  
可以单一事件管理组件通信：vuex

Var Event=new Vue();

Event.$emit(事件名称、数据)

Event.$on(事件名称，function(数据){

data

}。Bind(this))

Debounce 废弃

🡪lodash

--🡪\_debounce(fn,时间)

Vue动画

Vue路由

Transition 之前 属性

<p transition=”fade”></p>

.fade-transition{}

.fade-enter{}

.fade-leave{}

——————————————————————

到了2.0以后transition 组件

<transition name="fade">

运动东西（元素、属性、路由）

</transition>

Class定义：

.fade-enter{} //初始状态

.fade-enter-active{} //变化成什么样 –>当元素出来(消失)

.fade-leave{}

.fade-leave-active{}//变化成什么样 –>当元素离开(消失)

如何animate.css配合使用？

|  |
| --- |
| <transition enter-active-class="zoomInLeft"  leave-active-class="zoomOutRight" >  <p v-show="show" class="animated"></p>  </transition> |
| <transition enter-active-class="animated zoomInLeft"  leave-active-class="animated zoomOutRight" >  <p v-show="show" ></p>  </transition> |

多个元素运动

|  |
| --- |
| <transition-group enter-active-class="zoomInLeft"  leave-active-class="zoomOutRight" >  <p v-show="show" class="animated" :key="1"></p>  <p v-show="show" class="animated" :key="2"></p>  </transition-group> |
| <!DOCTYPE html>  <html lang="">  <head>  <meta charset="utf-8">  <meta http-equiv="X-UA-Compatible" content="IE=edge">  <meta name="viewport" content="width=device-width, initial-scale=1">  <title>Title Page</title>  <link href="./animate.css" rel="stylesheet">  <style>  p{  width: 100px;  height: 100px;  background: red;  margin:10px auto;  }  </style>  <script src="./vue.js"></script>  <script>  window.onload=function(){  var vm=new Vue({  el:'#box',  data:{  show:'',  list:['apple','orange','banana']  },  methods:{  },  computed:{  lists:function(){  var arr=[];  this.list.forEach(function(val){  if(val.indexOf(this.show)!=-1){  arr.push(val)  }  }.bind(this));  return arr;  }  }  })  }  </script>  </head>  <body>  <div id="box">  <input type="text" v-model="show">  <transition-group enter-active-class="animated zoomInLeft"  leave-active-class="animated zoomOutRight" >  <p v-show="show" v-for="(val,index) in lists" :key="index">{{val}}</p>  </transition-group>  </div>  </body>  </html> |

Vue2.0路由

1.基本使用

|  |
| --- |
| <div id="box">  <div>  <router-link to="/home">主页</router-link>  <router-link to="/news">新闻</router-link>  </div>  <div>  <router-view></router-view>  </div>  </div> |

1. 路由具体写法

//组件

var Home={

template:'<h3>我是主页</h3>'

}

var News={

template:'<h3>我是新闻</h3>'

}

//配置路由

var routes=[

{path:'/home',component:Home},

{path:'/news',component:News},

]

//生成路由实例

var router=new VueRouter({

routes

})

//最后挂到vue上

var vm=new Vue({

el:'#box',

data:{

},

methods:{

},

router

})

1. 重定向：

{path:'\*',redirect:'/home'},

路由嵌套

/user/username

var routes=[

{path:'/home',component:Home},

{path:'/user',

component:User,

children:[

{path:'username',component:UserDeatail,}

]},

{path:'\*',redirect:'/home'},

]

/user/strive/age/10

:id

:username

:age

路由实例方法：

Router.push({path:’home’})//直接添加路由，表现切换路由，本质往历史记录里面添加一个

Router.replace({path:’user’});//替换路由，不会往历史记录里面添加

-----------------------------

路由配合运动

<transition enter-active-class="animated bounceInLeft" leave-active-class="animated bounceOutRight">

<router-view></router-view>

</transition>

Vue-cli:

脚手架：vue-loader

1.0->

new Vue({

el: '#app',

components:{App}

})

* 1. ->

new Vue({

el: '#app',

render: h => h(App)

})

Vue2.0

Vue-loader和vue-router配合

Style-loader css-loader

Style!css

UI组件：

别人提供好一堆东西

目的：为了提高开发效率

原则：拿过来直接使用

Vue-cli vue-loader

-------------------------------

Bootstrap

基于jquery

栅格化系统+响应式工具（移动端、pad、pc）

按钮

Bower 前端包管理器 jquery#1.11.1

Npm node 包管理器 [jquery@1.11.1](mailto:jquery@1.11.1)

饿了么团队开源一个基于vue组件库

ElementUI PC

MintUI 移动端

……

Vue智能社论坛：<http://bbs.zhinengshe.com>

ElementUI

如何使用：官网：<http://element.eleme.io/>

使用

1. 安装element-ui

Npm I element-ui-D

等同于：npm install element-ui –save-dev

// I ->install 简写

// D ->--save-dev

// S ->--save

1. 引入 main.js 入口文件

Import ElementUI from ‘element-ui’

Emport ‘element-ui/lib/theme-default/index.css’

全部引入

1. 使用组件

Vue.use(ElementUI)

Css-loader引入css

字体图标 file-loader

Less:less less-loader

Vue-init webpack-simple element-demo

Cd element-demo

Npm install

Npm run dev

Npm I element-ui –D

Npm install style-loader

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

按需加载相应的组件：推荐

就需要按钮

1.babel-plugin-component

Npm install babel-plugin-component –D

2..babelrc文件里面新增一个配置

"plugins": [["component",[

{

"libraryName":"element-ui",

"styleLibraryName":"theme-default"

}

]]]

3.哪个组件就用哪个组件

引入：import {Button,Radio} from 'element-ui'

使用

a).Vue.component(Button.name,Button)

b).Vue.use(Button)

发送请求

vue-resourse 交互

axios

-------------------------

Mintui 移动端ui

https://mint-ui.github.io/#!/zh-cn

<http://mint-ui.github.io/>

1. 下载 npm install mint-ui –S

A).全部引入：

import Vue from 'vue'

import Mint from 'mint-ui';

import ‘mint-ui/lib/style.css’

Vue.use(Mint);

B).按需引入：

import { Cell, Checklist } from 'mint-ui';

Vue.component(Cell.name, Cell);

Vue.component(Checklist.name, Checklist);

<https://mint-ui.github.io/docs/#/zh-cn2/indicator>

mint-ui-demo

https://github.com/itstrive/striveCode/tree/js/vue2.0-Mint-ui-demo