# vue2.0:

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，不在支持片段代码

组件中模板:

之前:

<template>

<h3>我是组件</h3><strong>我是加粗标签</strong>

</template>

**现在: 必须有根元素，包裹住所有的代码**

<template id="aaa">

<div>

<h3>我是组件</h3>

<strong>我是加粗标签</strong>

</div>

</template>

## 2. 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，咱也不用——废弃

### Vue.component

Vue.component(组件名称,{ 在2.0继续能用

data(){}

methods:{}

template:

});

### 2.0推出一个组件，简洁定义方式：

var Home={

template:'' 相当于-> Vue.extend()

};

## 3. 生命周期

### 之前:

init

created

beforeCompile

compiled

ready √ -> 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 组件更新完毕 \*

beforeDestroy 组件销毁前 vm.$destroy()

destroyed 组件销毁后

## 3. 循环

2.0里面默认就可以添加重复数据

arr.forEach(function(item,index){

});

去掉了隐式一些变量

$index $key

之前:

v-for="(index,val) in array"

现在:

**v-for="(val,index) in array"**

## 4. track-by="id"

**变成 :key=’’**

<li v-for="(val,index) in list" :key="index">

## 5. 自定义键盘指令

之前：Vue.directive('on').keyCodes.f1=17;

现在: Vue.config.keyCodes.ctrl=17

## 6. 过滤器

之前:

系统就自带很多过滤

{{msg | currency}}

{{msg | json}}

....

limitBy

filterBy

.....

一些简单功能，自己通过js实现

到了2.0， 内置过滤器，全部删除了

lodash 工具库 \_.debounce(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数据:

### 通过 props

之前，子组件可以更改父组件信息，可以是同步 sync

现在，不允许直接给父级的数据，做赋值操作

问题，就想更改：

### 命名规范：组件间的通讯使用小写字母

### a). 父组件每次传一个对象给子组件, 对象之间引用 √

### 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(data){

//data

}.bind(this));

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

## debounce 废弃 延时执行

-> 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(显示)

.fade-leave{}

.fade-leave-active{} //变成成什么样 -> 当元素离开(消失)

## 如何animate.css配合用？

<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="" leave-active-class="">

<p :key=""></p>

<p :key=""></p>

</transition-group>

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

# vue2.0 路由:

http://router.vuejs.org/zh-cn/index.html

## 基本使用:

### 1. 布局

<router-link to="/home">主页</router-link>

<router-view></router-view>

### 2. 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

const routes=[

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

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

];

//生成路由实例

const router=new VueRouter({

routes

});

//最后挂到vue上

new Vue({

router,

el:'#box'

});

### 3. 重定向

之前 router.rediect 废弃了

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

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

### 路由嵌套:

/user/username

const routes=[

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

{

path:'/user',

component:User,

children:[ //核心

{path:'username', component:UserDetail}

]

},

{path:'\*', redirect:'/home'} //404

];

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

/user/strive/age/10

:id

:username

:age

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

### 路由实例方法:

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

router.replace({path:'news'}) //替换路由，不会往历史记录里面添加

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

vue-cli

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

npm install

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

## 脚手架: vue-loader

### 1.0 ->

new Vue({

el: '#app',

components:{App}

})

### 2.0->

new Vue({

el: '#app',

render: h => h(App)

})

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

### vue2.0

vue-loader和vue-router配合

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

style-loader css-loader

style!css

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

项目:

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

vue问题:

论坛

http://bbs.zhinengshe.com

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

# UI组件

别人提供好一堆东西

目的:

为了提高开发效率

功能

原则: 拿过来直接使用

vue-cli -> vue-loader

## bootstrap:

twitter 开源

简洁、大方

官网文档

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

自动解决依赖

### npm node包管理器 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

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

## elementUI:

如何使用

官网:http://element.eleme.io/

使用:

1. 安装 element-ui

npm i element-ui -D

npm install element-ui --save-dev

// i -> install

// D -> --save-dev

// S -> --save

2. 引入 main.js 入口文件

import ElementUI from 'element-ui'

import 'element-ui/lib/theme-default/index.css'

全部引入

3. 使用组件

Vue.use(ElementUI)

css-loader 引入css

字体图标 file-loader

less:

less

less-loader

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

### 按需加载相应组件: √

就需要 按钮

1. babel-plugin-component

cnpm 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. 想用哪个组件就用哪个

引入:

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

使用:

a). Vue.component(Button.name, Button); 个人不太喜欢

b). Vue.use(Button); √

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

发送请求:

vue-resourse 交互

## **Axios**

**/\*交互\*/**

**引入：import axios from 'axios'**

**书写方法调用：**

**axios.get('https://api.github.com/users/itstrive')**

**.then(function(res){**

**this.myMessage=res.data;**

**}.bind(this)).catch(function(err){**

**console.log(err);**

**})**

**最新ajax，跨域， promise用法**

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

### element-ui -> pc

## mint-ui

移动端 ui库

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

1. 下载

npm install mint-ui -S

-S

--save

2. 引入

import Vue from 'vue';

**全部引入**

import Mint from 'mint-ui';

import 'mint-ui/lib/style.css'

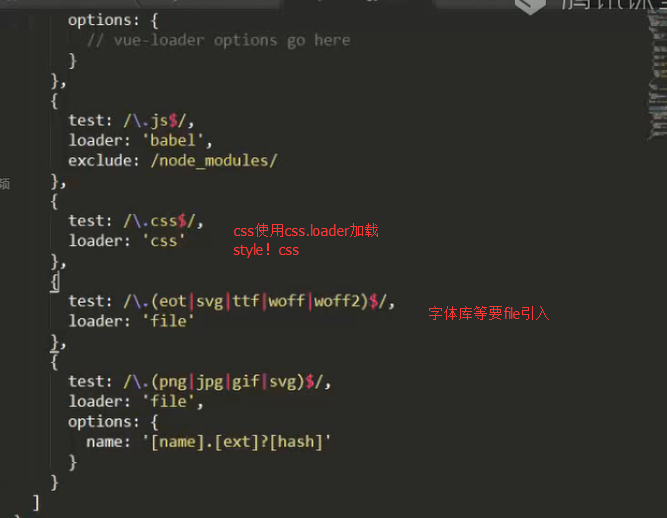
Vue.use(Mint);

**按需引入:**

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

Vue.component(Cell.name, Cell);

Vue.component(Checklist.name, Checklist);



http://mint-ui.github.io/docs/#!/zh-cn2

论坛:

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

## Mint-ui-demo: 看着手册走了一遍

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

# 自定义全局组件

## 使用别人的组件

* Import Vue from ‘Vue’;
* Import XXX from ‘XXX’;
* Vue.use(XXX);

在使用Axios的时候，并不能使用Vue.use(Axios),

**[因为没有写 install]**

## 自定义组件

Import xxx from ‘./components/loading/’ 默认选择的是index.js

### 组件目录结构

--- index.js

---Loading.vue

Index.js

Import LoadingComponent from ‘./loading.vue’;

Const Loading={

Install:function(Vue){

//vue.component(name,component组件)

Vue.component(‘Loading’,LoadingCpmponent)

},

};

Export default Loading;

此时，使用自定义组件即可

1. Import Loading from ‘./components/loading/index.js’ 默认index.js可不写

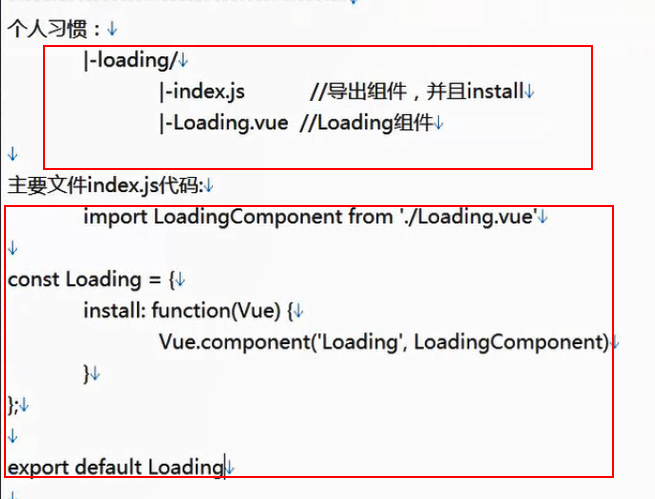
Import Loading from ‘./components/loading’

1. Vue.use(Loading);



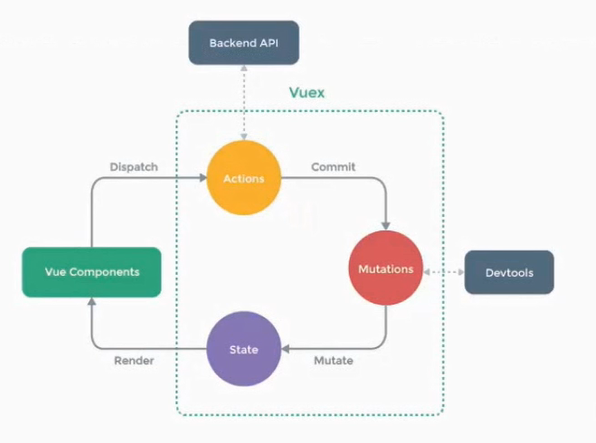
错误分析：

### 自定义组件



# VUEX:集中式管理数据

官网：<http://vuex.vuejs.org/>



1. Vue触发actions，

安装：vuex npm install vuex -D //--save-dev

# 单一结构

Vuex 提供的方法

* mapActions // 管理所有事件
* mapGetters //获取数据

**书写：**

export default{

computed:mapGetters([

'count',

'getOdd'

]),

methods:mapActions([

'increment',

'decrement',

'clickOdd',

'clickAsync'

])

}

**方法：**

Actions = {

clickOdd: ({

commit,

//**传递commit，state对象**

state

}) => {

if (state.mutations.count % 2 == 0) {

**//state对象的mutations包含数据**

commit(types.INCREMENT);

}

}

}

**导出store对象**

export default new Vuex.Store({

modules:{

mutations

},

**actions**

});

**Main.js导入store对象**

new Vue({

store,

el: '#app',

render: h => h(App)

})

可以直接使用 $store 整个对象

* import {mapGetters, mapActions} from 'vuex'

导入vuex提供的方法 mapGetters，mapActions

* 导出Store对象

export default new Vuex.Store({

modules:{

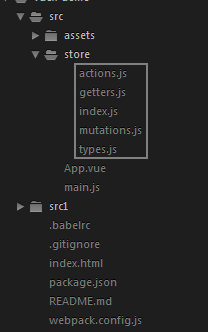
mutations

},

actions

});

## 目录结构



## 流程图：

Index.html——>build.js——>main.js——>[vue.vue..，store.js]

vue——>vuex { mapGetters , mapActions } ----->