VUEJS渐进式开发

猛叔
okman

修订历史

日期	说明
20171126	创建，html/bootstrap/vue菜单效果
20171127	Vue菜单模板化
20171128	从后台获取json数组并在前台生成菜单
20180109	调整顺序，优化代码
待定	抽取单独文件
	单页应用

目录

需求
Input

入门
design philosophy

1.项目
Project

2.渐进实现
StepbyStep

3.对比
vs Angular/REACT

Vue.js基础
Specification

4.模板
Flat

5.指令
Responsive

进阶
implementation

6.布局
Layout

7.组件
Component

8.公文版式
Document UI

Next

准备工作

下载：

Bootstrap:

https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css
https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.js

jquery

https://code.jquery.com/jquery.js

Vue.js

https://unpkg.com/vue

基本HTML模板

```
<!DOCTYPE html>
<html lang="zh-CN">
  <head>
   <meta charset="utf-8">
    <title>Bootstrap 模板</title>
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <!-- 引入 Bootstrap -->
    <link href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css" rel="stylesheet">

    <!-- HTML5 Shiv 和 Respond.js 用于让 IE8 支持 HTML5元素和媒体查询 -->
    <!-- 注意： 如果通过 file:// 引入 Respond.js 文件，则该文件无法起效果 -->
    <!--[if lt IE 9]>
      <script src="https://oss.maxcdn.com/libs/html5shiv/3.7.0/html5shiv.js"></script>
      <script src="https://oss.maxcdn.com/libs/respond.js/1.3.0/respond.min.js"></script>
    <![endif]-->
  </head>
  <body>
    <h1>Hello, world!</h1>

    <!-- jQuery (Bootstrap 的 JavaScript 插件需要引入 jQuery) -->
    <script src="https://code.jquery.com/jquery.js"></script>
    <!-- 包括所有已编译的插件 -->
    <script src="js/bootstrap.min.js"></script>
                <!--引入vue.js -->
   <script src="vue/vue"></script>

  </body>
</html>
```

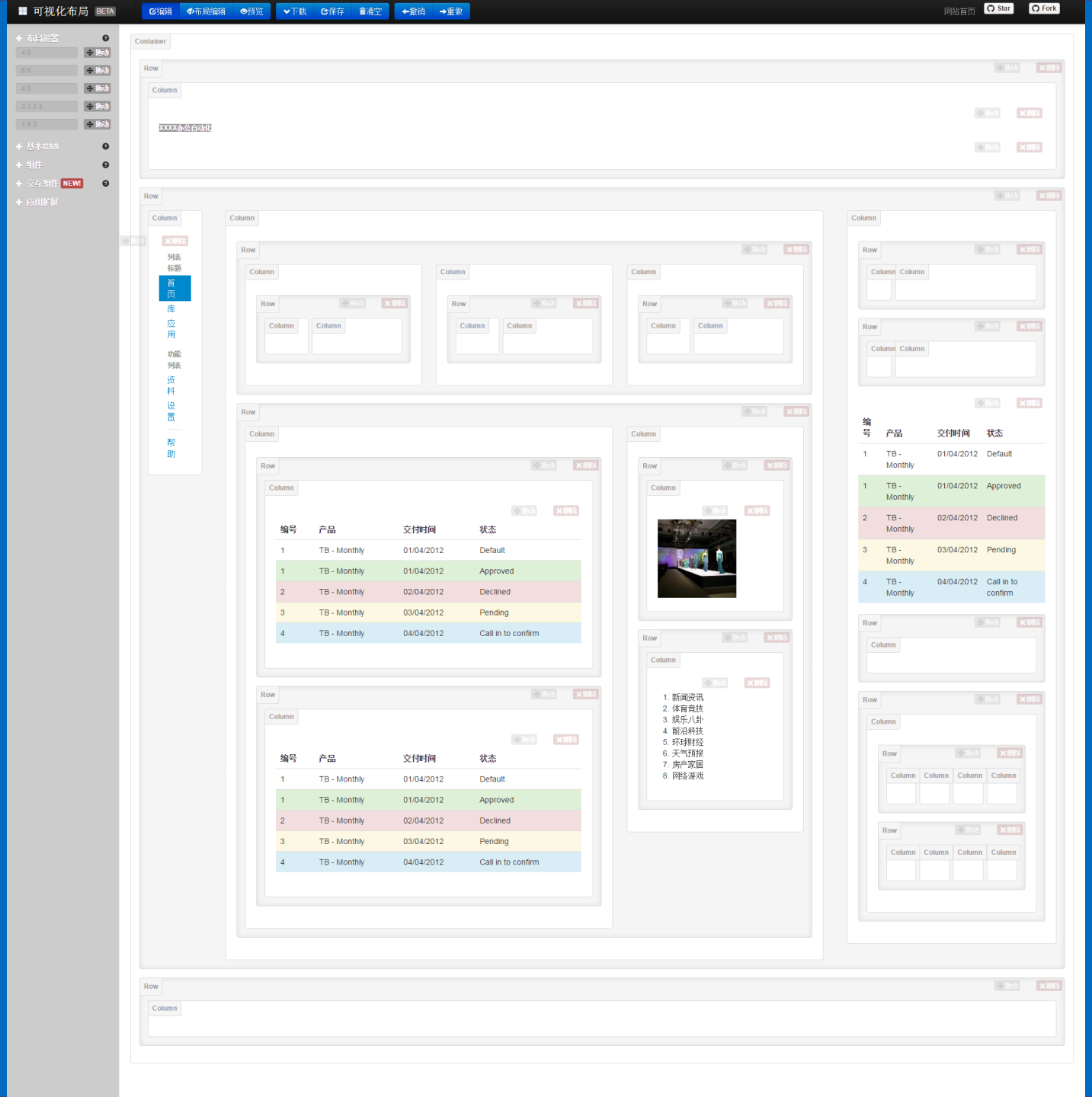
样本程序

12栅格和24栅格两种栅格体系

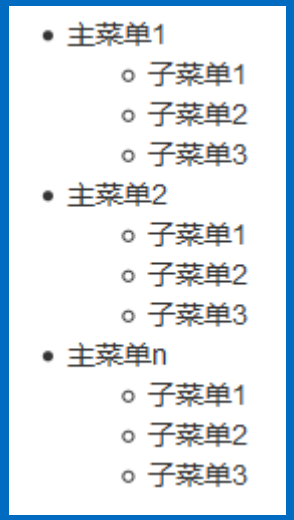
开始设计布局

http://www.ibootstrap.cn/

比例：
2：7：3


可视化布局 BETA
编辑
布局编辑
预览
下载
保存
清空
撤销
重做
网站首页
Star
Fork
布局设置
基本CSS
组件
交互组件 NEW!
应用扩展
Container
Row
Column
XXXX办公自动化
列表标题
首页
库
应用
功能列表
资料
设置
帮助
编号 产品 交付时间 状态
1 TB - Monthly 01/04/2012 Default
1 TB - Monthly 01/04/2012 Approved
2 TB - Monthly 02/04/2012 Declined
3 TB - Monthly 03/04/2012 Pending
4 TB - Monthly 04/04/2012 Call in to confirm
1. 新闻资讯
2. 体育竞技
3. 娱乐八卦
4. 前沿科技
5. 环球财经
6. 天气预报
7. 房产家居
8. 网络游戏

第0步：菜单设计




```
<div class="col-xs-2 visible-md-block visible-lg-block ">
<div id="mainmenu">
  <ul>
     <li>主菜单1
              <ul >
              <li > 子菜单1 </li>
              <li > 子菜单2 </li>
              <li > 子菜单3 </li>
          </ul>
     </li>

 <li>主菜单2
              <ul >
              <li > 子菜单1 </li>
              <li > 子菜单2 </li>
              <li > 子菜单3 </li>
          </ul>
     </li>
 <li>主菜单n
              <ul >
              <li > 子菜单1 </li>
              <li > 子菜单2 </li>
              <li > 子菜单3 </li>
          </ul>
     </li>
</ul>
  </div>
</div>
```

设计要求

1、单击主菜单折叠/显示子菜单

2、主菜单内容/子菜单内容动态化

可以演示了，有一个workable的软件！！！！

第1步：Vue.js迭代生成菜单

```
<div class="col-xs-2 visible-md-block visible-lg-block ">
 <div id="mainmenu">
  <ul>
    <li v-for="item in items">
  <span @click="showToggle(item)">{{item.name}}</span>
        <ul v-show="item.isSubShow">
  <li v-for="subItem in item.subItems">
      <span>{{subItem.name}}</span>
  </li>
      </ul>
    </li>
  </ul>
  </div>
</div>
```

v-for：迭代，自动生成列表
@click：单击主菜单项时的动作绑定showToggle(item)动作处理函数
v-show：根据属性显示或隐藏

```
<script>
new Vue({
  el:"#mainmenu",
  data:{
    items:[
      {
        name:'公文',
        isSubShow:false,
        subItems:[
          { name:'收文' },
          { name:'来文' },
          { name:'起草' }
        ]
      },
      {
        name:'会议管理',
        isSubShow:false,
        subItems:[
          { name:'会议' },
          { name:'会议室' },
          { name:'在线会议' }
        ]
      },
      {
        name:'消息',
        isSubShow:false,
        subItems:[
          { name:'1' },
          { name:'2' },
          { name:'3' }
        ]
      }
    ]
  },
  methods:{
    showToggle:function(item){
      item.isSubShow = !item.isSubShow
    }
  }
})
</script>
```

切换子菜单显示状态

菜单数据

待完善点
1、菜单项从后台传入，根据权限生成，不是静态写在页面中
2、菜单做成VUE组件
3、点击一个主菜单项，展开子菜单同时，其他主菜单的子菜单项关闭
4、菜单美化

依然可以演示，有一个workable的软件！！！！

第2步：从后台获取菜单数据

布局容器及模板不变

菜单数据剥离到后台

```
<div id="mainmenu">
 <ul>
  <li v-for="item in items">
   <span @click="showToggle(item)">{{item.name}}</span>
    <ul v-show="item.isSubShow">
        <li v-for="subItem in item.subItems">
          <span>{{subItem.name}}</span>
        </li>
    </ul>
  </li>
 </ul>
</div>
```

```
<script>

    var url = "./fakedata/left-nav.json";
    $.getJSON(url, function (res) {
       json = eval(res);
      console.log(json);
    })
    vm = new Vue({
        el: "#mainmenu",
        template: "#myMenu",
        data: { items:json.items},
       mounted: function () {
            _self=this;
            $.getJSON(url, function (res) {
               json = eval(res);
              _self.items = json.items;
            })
        },
        methods: {
            showToggle: function (item) {
                item.isSubShow = !item.isSubShow
            }
        }
    })


</script>
```

待完善点

2、组件剥离为独立VUE文件

3、点击一个主菜单项，展开子菜单同时，其他主菜单的子菜单项关闭

4、菜单美化

未解决bug，需要获取2次数据。

未解决点：

1、定义阶段，data中如何指定未生成的json数据

2.Mounted阶段如何将取得的json数据传递给data

始终有一个workable的软件！！！！

第3步：优化从后台获取菜单数据的过程

布局容器及模板不变

```
<div id="mainmenu">
  <ul>
   <li v-for="item in items">
    <span @click="showToggle(item)">{{item.name}}</span>
     <ul v-show="item.isSubShow">
         <li v-for="subItem in item.subItems">
           <span>{{subItem.name}}</span>
         </li>
     </ul>
   </li>
 </ul>
</div>
```

菜单数据剥离到后台

```
<script>

    var url = "./fakedata/left-nav.json";
    vm = new Vue({
        el: "#mainmenu",
        template: "#myMenu",
        data: { items:[]},
       mounted: function () {
            _self=this;
            $.getJSON(url, function (res) {
               json = eval(res);
              _self.items = json.items;
            })
        },
        methods: {
            showToggle: function (item) {
                item.isSubShow = !item.isSubShow
            }
        }
    })


</script>
```

```
var url = "./fakedata/left-nav.json";
vm = new Vue({
    el: "#mainmenu",
    template: "#myMenu",
    data: { items:[]},
   mounted: function () {
        let _self=this;
        $.getJSON(url, function (json) {
           json = eval(res);
          _self.items = json.items;
        })
    },
    methods: {
        showToggle: function (item) {
            item.isSubShow = !item.isSubShow
        }
    }
})
```

始终有一个workable的软件！！！！

第4步：抽出模板

```
<div class="col-xs-2 visible-md-block visible-lg-block ">
 <div id="mainmenu">
  <ul>
    <li v-for="item in items">
  <span @click="showToggle(item)">{{item.name}}</span>
       <ul v-show="item.isSubShow">
  <li v-for="subItem in item.subItems">
     <span>{{subItem.name}}</span>
  </li>
      </ul>
    </li>
  </ul>
 </div>
</div>
```

<template id=“myMenu”>定义模板
其他不变
v-for：迭代，自动生成列表
@click：单击主菜单项时的动作绑定showToggle(item)动作处理函数
v-show：根据属性显示或隐藏

```
<script>
new Vue({
   el:"#mainmenu",
   template:”#myMenu”,
   data:{
```

指定模板

其他不变

```
<div class="col-xs-2 visible-md-block visible-lg-block ">
   <div id="mainmenu">
   <myMenu></myMenu>
    </div>
</div>
```

```
<template id="myMenu">
  <ul>
    <li v-for="item in items">
  <span @click="showToggle(item)">{{item.name}}</span>
       <ul v-show="item.isSubShow">
  <li v-for="subItem in item.subItems">
     <span>{{subItem.name}}</span>
  </li>
      </ul>
    </li>
  </ul>
</template>
```

待完善点
1、菜单项从后台传入，根据权限生成，不是静态写在页面中
2、组件剥离为独立VUE文件
3、点击一个主菜单项，展开子菜单同时，其他主菜单的子菜单项关闭
4、菜单美化

始终有一个workable的软件！！！！

Demo2：项目桌牌！！！

场景

业务：做桌牌

美工做法：一个一个PS的做

挑战：如果要调整字体大小、颜色、背景怎么办？

可选做法：用WORD的

要求：
10分钟编写一个VUE程序
美工去完善设计
美工维护JSON数据

除了生成项目桌牌，还要生成项目清单

做法（10分钟搞定）

模板：扔给美工去美化

```
<div id="projects">
                    <div v-for="item
in items">
        <div >
 <p >{{item.prj_name}}</p>

 <p>项目经理：{{item.prj_manager}}
{{item.prj_manager_mobile}}
{{item.prj_member_number}}</p>

 <p>行方经理：{{item.owner_manager}}
{{item.owner_manager_phone}}
{{item.owner_dept}}</p>

 <p>合同时间：{{item.contract_date}}
合同人数：{{item.contract_member}}
</p>

        </div>
```

js：够用了

```
 <script src="project.js">
  </script>



var url = "./fakedata/projects.json";
vm = new Vue({
    el: "#projects",
    data: {
        items: []
    },
    mounted: function () {
        _self = this;
        $.getJSON(url, function (res) {
            json = eval(res);
            _self.items = json.items;
        })
    }
})
```

JSON数据定义，可以美工维护，也可


```
{
 "items": [
  {
    "prj_name": "统一授权管理系统",
    "prj_manager": "白因",
    "prj_manager_mobile": "13910161532",
    "prj_member_number": "2",
    "owner_manager": "邓炬",
    "owner_manager_phone": "010-88303633",
    "owner_dept": "开发测试中心综合业务处",
    "contract_date": "2017-01-01",
    "contract_member": "5",
      "contract_1member": "5",
       "contract_member2": "5"

  },
  {
    "prj_name": "新OA",
    "prj_manager": "吕建松",
    "prj_manager_mobile": "13910161532",
    "prj_member_number": "2",
    "owner_manager": "邓炬",
    "owner_manager_phone": "010-88303633",
    "owner_dept": "开发测试中心综合业务处",
    "contract_date": "2017-01-01",
    "contract_member": "5"
  },
  {
    "prj_name": "助贷门户网站",
    "prj_manager": "韩佳建",
    "prj_manager_mobile": "13910161532",
    "prj_member_number": "2",
    "owner_manager": "邓炬",
    "owner_manager_phone": "010-88303633",
    "owner_dept": "开发测试中心综合业务处",
    "contract_date": "2017-01-01",
    "contract_member": "5"
  }
 ]
}
```

再生成表格——再做个视图即可，不需要改JS，也不需要改数据

```
<div id="projects">
        <table class="table table-striped table-bordered">
<tr>
   <td>项目名称</td>
   <td>项目经理</td>
   <td>手机</td>
   <td>项目人数</td>
   <td>行方经理</td>
   <td>座机</td>
   <td>部门</td>
   <td>合同时间</td>
   <td>合同人数</td>
</tr>
```

```
 <tr v-for="item in items">
    <td>{{item.prj_name}}</td>
    <td>{{item.prj_manager}}</td>
    <td>{{item.prj_manager_mobile}}</td>
    <td>{{item.prj_member_number}}</td>
    <td>{{item.owner_manager}}</td>
    <td>{{item.owner_manager_phone}}</td>
    <td>{{item.owner_dept}}</td>
    <td>{{item.contract_date}}</td>
    <td>{{item.contract_member}}</td>
 </tr>
</table>
</div>

<script src="project.js">
</script>
```

高级主题

过滤器-自定义

4.过滤器

管道符连接

{{ name | uppercase }} // VUE

多个管道连接多个过滤器

{{ name | filterA | filterB }}

过滤器可以使用参数

{{ name | filterA arg1 arg2}}

Vue.js 内置了10 个过滤器：

① capitalize ：字符串首字符转化成大写
② uppercase ：字符串转化成大写
③ lowercase ：字符串转化成小写
④ currency：参数为{String}[货币符号] ,{Number} [小数位]，将数字转化成货币符号，并且会自动添加数字分节号。
⑤ pluralize：参数为{String} single, [double, triple]，字符串复数化。
⑥ json 参数为{Number}[indent] 空格缩进数，与JSON.stringify() 作用相同，将json对象数据输出成符合json 格式的字符串。
⑦ debounce 传入值必须是函数，参数可选，为{Number}[wait]，即延时时长
⑧ limitBy 传入值必须是数组，参数为{Number}limit，{Number}[offset], limit 为显示个数，offset 为开始显示数组下标。
⑨ filterBy 传入值必须是数组，参数为{String | Function} targetStringOrFunction，即需要匹配的字符串或函数（通过函数返回值为 true 或 false 来判断匹配结果）； “in”（ 可选分隔符）；{String}[…searchKeys]，为检索的属性区域。
⑩ orderBy 传入值必须是数组，参数为{String|Array|Function}sortKeys，即指定排序策略。这里可以使用单个键名，也可以传入包含多个排序键名的数组。也可以像Array.Sort()那样传入自己的排序策略函数。第二个参数为可选参数{String}[order]，即选择升序或降序，order>=0 为升序，order<0 为降序。

VUE-CLI

安装

下载安装NODEJS
http://nodejs.org/dist/latest/

将NODEJS加入path

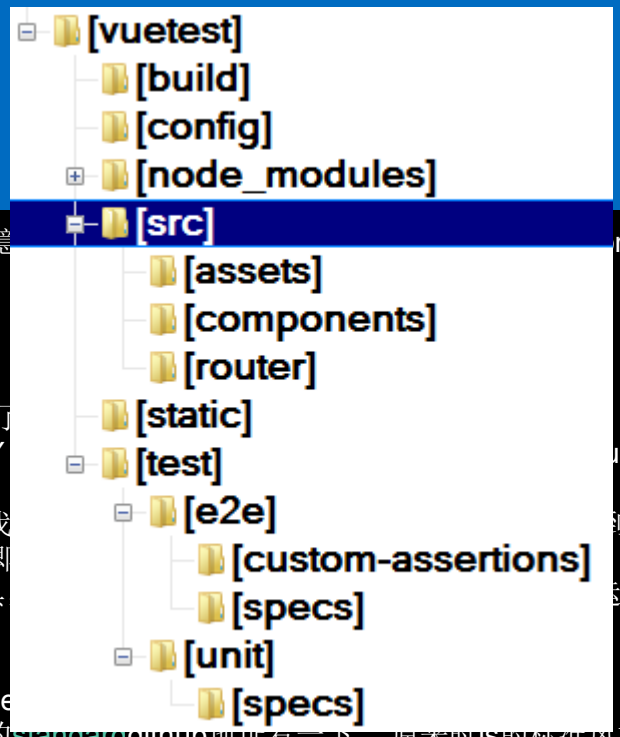
进入命令行：
npm install vue-cli

用vue-cli创建项目

创建一个项目文件夹
在命令行进入项目文件夹

vue-cli init webpack <项目名称>

生成的项目结构

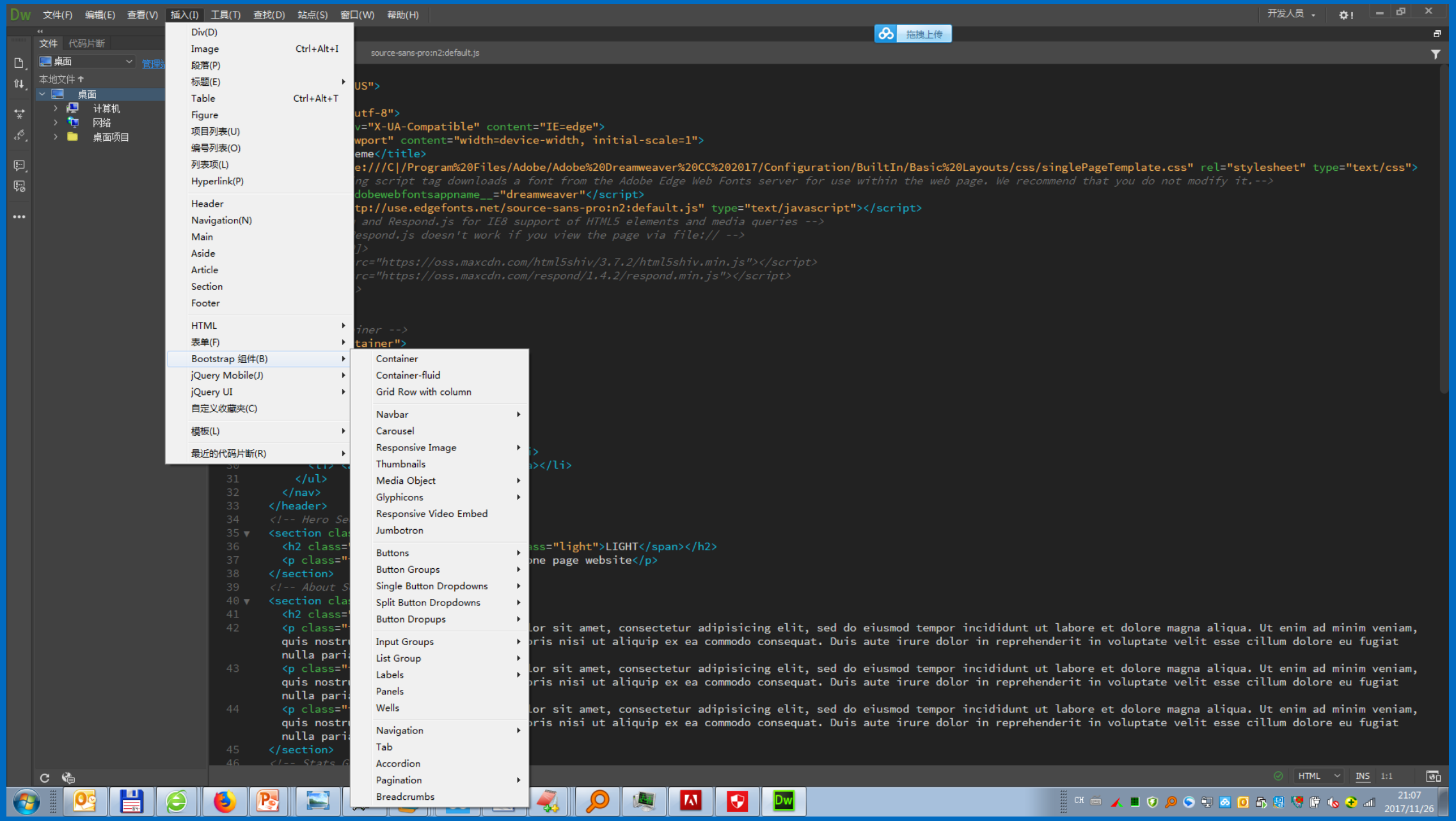


Project name (vuetest)　项目名称，可以自己指定，也可直接回车，按照括号中默认名字（注意rry, name can no longer contain capital letters），阮一峰老师博客为什么文件名要小写，可以参考一下。
Project description (A Vue.js project)　项目描述，也可直接点击回车，使用默认名字
Author (.....)　作者，不用说了，你想输什么就输什么吧
接下来会让用户选择
Runtime + Compiler: recommended for most users　运行加编译，既然已经说了推荐，就选它了
Runtime-only: about 6KB lighter min+gzip, but templates (or any Vue-specificHTML) are ONLY uired elsewhere　仅运行时，已经有推荐了就选择第一个了
Install vue-router? (Y/n)　是否安装vue-router，这是官方的路由，大多数情况下都使用，不过我到官方路由，因为有特殊需求，也因为比较早，官方尚未成熟，vue-router官网。这里就输入“y”后回车即
Use ESLint to lint your code? (Y/n)　是否使用ESLint管理代码，ESLint是个代码风格管理工具行，这也是为了多人协作，新手就不用了，一般项目中都会使用。ESLint官网

接下来也是选择题Pick an ESLint preset (Use arrow keys)　选择一个ESLint预设，编写vue
Standard (https://github.com/feross/standard)　标准，有些看不明白，什么标准呢，去给提示的standardgithub地址看一下，原来的js的标准风格
AirBNB (https://github.com/airbnb/javascript)　JavaScript最合理的方法，这个github地址说的是JavaScript最合理的方法
none (configure it yourself)　这个不用说，自己定义风格
具体选择哪个因人而异吧 ，我选择标准风格
Setup unit tests with Karma + Mocha? (Y/n)　是否安装单元测试，选择安装
Setup e2e tests with Nightwatch(Y/n)?　是否安装e2e测试，选择安装

项目结构说明

```
.
|-- build                  // 项目构建(webpack)相关代码
|   |-- build.js           // 生产环境构建代码
|   |-- check-version.js               // 检查node、npm等版本
|   |-- dev-client.js // 热重载相关
|   |-- dev-server.js // 构建本地服务器
|   |-- utils.js           // 构建工具相关
|   |-- webpack.base.conf.js           // webpack基础配置
|   |-- webpack.dev.conf.js            // webpack开发环境配置
|   |-- webpack.prod.conf.js           // webpack生产环境配置
|-- config                 // 项目开发环境配置
|   |-- dev.env.js         // 开发环境变量
|   |-- index.js           // 项目一些配置变量
|   |-- prod.env.js        // 生产环境变量
|   |-- test.env.js        // 测试环境变量
|-- src// 源码目录
|   |-- components         // vue公共组件
|   |-- store              // vuex的状态管理
|   |-- App.vue            // 页面入口文件
|   |-- main.js            // 程序入口文件，加载各种公共组件
|-- static                 // 静态文件，比如一些图片，json数据等
|   |-- data               // 群聊分析得到的数据用于数据可视化
|-- .babelrc               // ES6语法编译配置
|-- .editorconfig          // 定义代码格式
|-- .gitignore             // git上传需要忽略的文件格式
|-- README.md              // 项目说明
|-- favicon.ico
|-- index.html             // 入口页面
|-- package.json           // 项目基本信息
.
```


文件(F) 编辑(E) 查看(V) 插入(I) 工具(T) 查找(D) 站点(S) 窗口(W) 帮助(H)
开发人员
拖拽上传
文件 代码片断
桌面
本地文件
计算机
网络
桌面项目
source-sans-pro:n2:default.js
Div(D)
Image Ctrl+Alt+I
段落(P)
标题(E)
Table Ctrl+Alt+T
Figure
项目列表(U)
编号列表(O)
列表项(L)
Hyperlink(P)
Header
Navigation(N)
Main
Aside
Article
Section
Footer
HTML
表单(F)
Bootstrap 组件(B)
jQuery Mobile(J)
jQuery UI
自定义收藏夹(C)
模板(L)
最近的代码片断(R)
Container
Container-fluid
Grid Row with column
Navbar
Carousel
Responsive Image
Thumbnails
Media Object
Glyphicons
Responsive Video Embed
Jumbotron
Buttons
Button Groups
Single Button Dropdowns
Split Button Dropdowns
Button Dropups
Input Groups
List Group
Labels
Panels
Wells
Navigation
Tab
Accordion
Pagination
Breadcrumbs
HTML
INS
1:1
21:07
2017/11/26

模板数据绑定

指令
- v-text
- v-html
- v-show
- v-if
- v-else
- v-else-if
- v-for
- v-on
- v-bind
- v-model
- v-pre
- v-cloak
- v-once

组件定义
- data
- props
- propsData
- computed
- methods
- watch
- filters
- 生命周期钩子

选项 / 生命周期钩子
- beforeCreate
- created
- beforeMount
- mounted
- beforeUpdate
- updated
- activated
- deactivated
- beforeDestroy
- destroyed
- errorCaptured

数据绑定

1.文本插值

默认双向绑定

Hello {{ name }} // -> Hello Vue

指定单次绑定

{{name}}

2.HTML属性绑定

v-bind

快捷语法

<div :id="'id-' + id">

3.表达式绑定

<{{ index + 1 }}　// 1

{{ index == 0 ? 'a' : 'b'}} // a

{{ name.split('').join('|') }}　// V|u|e

注意：

1、不支持语句

2、复杂处理还是放在方法里为好

过滤器

4.过滤器

管道符连接

{{ name | uppercase }} // VUE

多个管道连接多个过滤器

{{ name | filterA | filterB }}

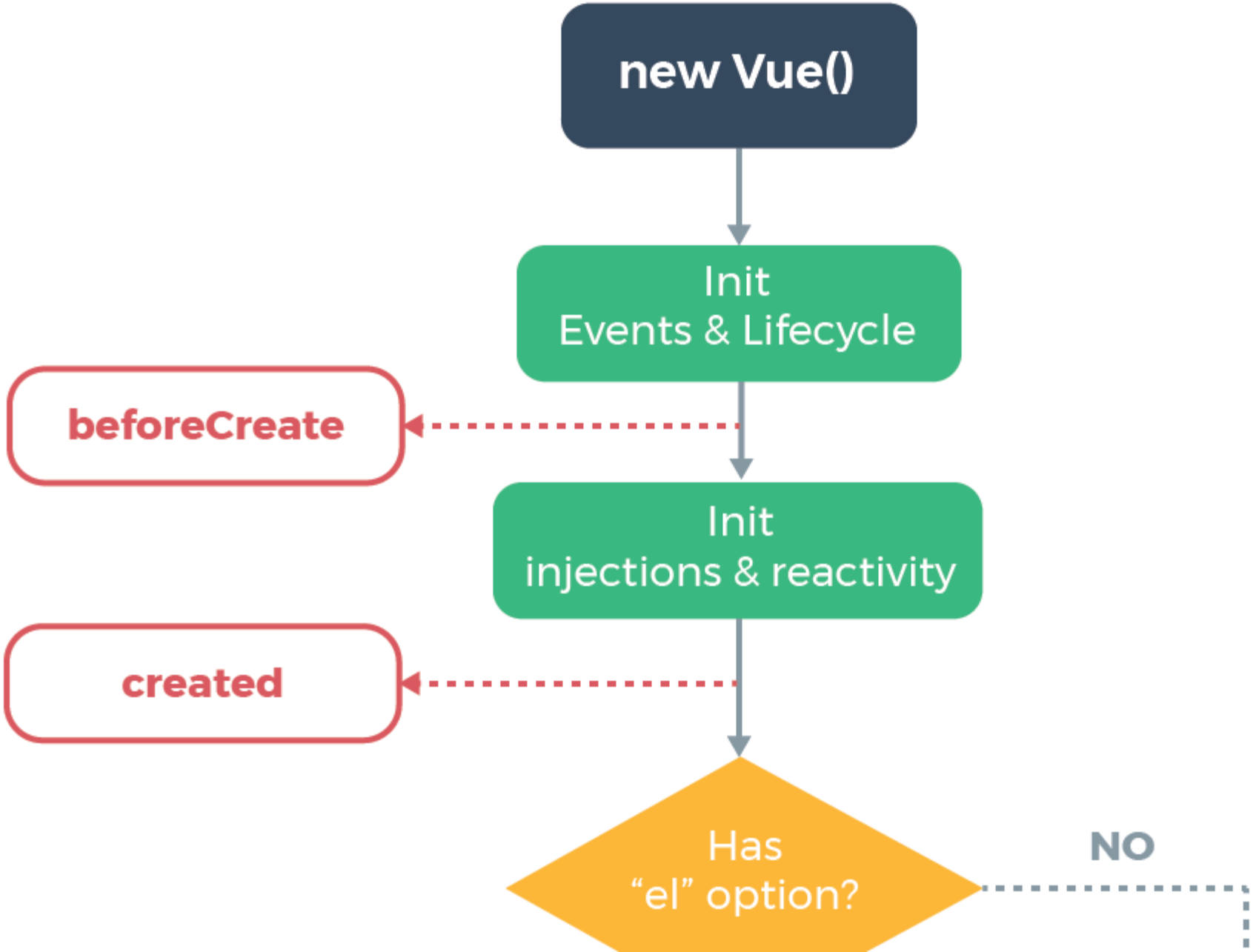
过滤器可以使用参数

{{ name | filterA arg1 arg2}}

Vue.js 内置了10 个过滤器：

① capitalize ：字符串首字符转化成大写
② uppercase ：字符串转化成大写
③ lowercase ：字符串转化成小写
④ currency：参数为{String}[货币符号] ,{Number} [小数位]，将数字转化成货币符号，并且会自动添加数字分节号。
⑤ pluralize：参数为{String} single, [double, triple]，字符串复数化。
⑥ json 参数为{Number}[indent] 空格缩进数，与JSON.stringify() 作用相同，将json对象数据输出成符合json 格式的字符串。
⑦ debounce 传入值必须是函数，参数可选，为{Number}[wait]，即延时时长
⑧ limitBy 传入值必须是数组，参数为{Number}limit，{Number}[offset], limit 为显示个数，offset 为开始显示数组下标。
⑨ filterBy 传入值必须是数组，参数为{String | Function} targetStringOrFunction，即需要匹配的字符串或函数（通过函数返回值为 true 或 false 来判断匹配结果）； “in”（ 可选分隔符）；{String}[…searchKeys]，为检索的属性区域。
⑩ orderBy 传入值必须是数组，参数为{String|Array|Function}sortKeys，即指定排序策略。这里可以使用单个键名，也可以传入包含多个排序键名的数组。也可以像Array.Sort()那样传入自己的排序策略函数。第二个参数为可选参数{String}[order]，即选择升序或降序，order>=0 为升序，order<0 为降序。

始终有一个workable的软件！！！！


new Vue()
Init
Events & Lifecycle
beforeCreate
Init
injections & reactivity
created
Has
“el” option?
NO

1. beforeCreate：在实例初始化之后，数据观测(data observer) 和 event/watcher 事件配置之前被调用。
2. Created：实例已经创建完成之后被调用。在这一步，实例已完成以下的配置：数据观测(data observer)，属性和方法的运算， watch/event 事件回调。然而，挂载阶段还没开始，$el 属性目前不可见。
3. beforeMount：在挂载开始之前被调用：相关的 render 函数首次被调用。
4. Mounted：el 被新创建的 vm.$el 替换，并挂载到实例上去之后调用该钩子。
5. beforeUpdate：数据更新时调用，发生在虚拟 DOM 重新渲染和打补丁之前。 你可以在这个钩子中进一步地更改状态，这不会触发附加的重渲染过程。
6. updated：由于数据更改导致的虚拟 DOM 重新渲染和打补丁，在这之后会调用该钩子。当这个钩子被调用时，组件 DOM 已经更新，所以你现在可以执行依赖于 DOM 的操作。然而在大多数情况下，你应该避免在此期间更改状态，因为这可能会导致更新无限循环。该钩子在服务器端渲染期间不被调用。
7. beforeDestroy:实例销毁之前调用。在这一步，实例仍然完全可用。
8. Destroyed:Vue 实例销毁后调用。调用后，Vue 实例指示的所有东西都会解绑定，所有的事件监听器会被移除，所有的子实例也会被销毁。 该钩子在服务器端渲染期间不被调用。

指令

指令 (Directives) 是带有 v- 前缀的特殊属性。指令属性的值预期是**单个 JavaScript 表达式**(v-for 是例外情况，稍后我们再讨论)。指令的职责是，当表达式的值改变时，将其产生的连带影响，响应式地作用于 DOM。

1. v-if指令：根据表达式的值在DOM中生成或移除一个元素。如果v-if表达式赋值为false，那么对应的元素就会从DOM中移除；否则，对应元素的一个克隆将被重新插入DOM中
2. v-show指令是根据表达式的值来显示或者隐藏HTML元素。当v-show赋值为false时，元素被隐藏。查看DOM时，会发现元素上多了一个内联样式style="display:none"。一般来说，v-if有更高的切换消耗，而v-show有更高的初始渲染消耗。因此，如果需要频繁的切换，则使用v-show较好；如果在运行时条件不大可能改变，则使用v-if较好。
3. v-else就是JavaScript中的else的意思，它必须跟着v-if或者v-show使用。
4. v-model指令用来在input、select、text、checkbox、radio等表单控件元素上创建双向数据绑定的。根据控件类型v-model自动选取正确的方法更新元素
5. v-for