基于 Webpack & Vue & Vue-Router 的 SPA 初体验

• 发布于 1 年前 • 作者 Bugly_Tony (/user/115290) • 331548 次浏览 • 来自 技术

最近这几年的前端圈子，由于戏台一般精彩纷呈，从 MVC 到 MVVM，你刚唱罢我登场。backbone，angularjs 已成昨日黄花，reactjs 如日中天，同时另一更轻量的 vue 发展势头更猛，尤其是即将 release 的2.0版本，号称兼具了 angularjs 和 reactjs 的两者优点。不过现在的官方版本还是1.0 ， 下面就是基于1.0版本的初体验。

1. 为什么要 SPA?

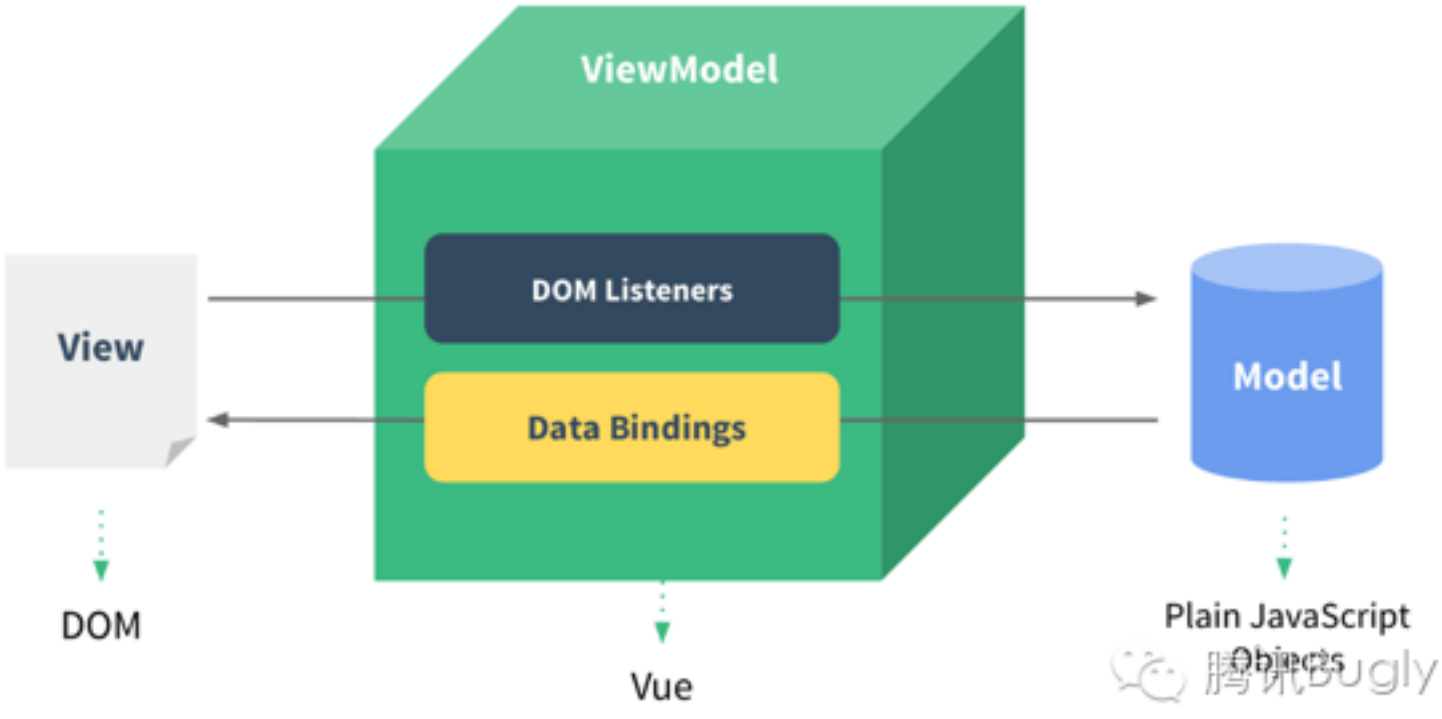
SPA： 就是俗称的单页应用（Single Page Web Application）。

在移动端，特别是 hybrid 方式的H5应用中，性能问题一直是痛点。 使用 SPA，没有页面切换，就没有白屏阻塞，可以大大提高 H5 的性能，达到接近原生的流畅体验。

2. 为什么选择 vue?

在选择 vue 之前，使用 reactjs 也做过一个小 Demo，虽然两者都是面向组件的开发思路，但是 reactjs 的全家桶方式，实在太过强势，而自己定义的 JSX 规范，揉和在 JS 的组件框架里，导致如果后期发生页面改版工作，工作量将会巨大。

vue 相对来说，就轻量的多，他的view层，还是原来的 dom 结构，除了一些自定义的 vue 指令作为自定义标签以外，只要学会写组件就可以了，学习成本也比较低。



3. 环境配置

初始化工程，需要 node 环境使用 npm 安装相应的依赖包。

先创建一个测试目录，在里面依次输入以下命令。

```
//初始化package.json
npm init

//安装vue的依赖
npm install vue --save
npm install vue-router --save

//安装webpack的开发依赖
npm install webpack --save-dev

//安装babel的ES6 Loader 的开发依赖
npm install babel --save-dev
npm install babel-core --save-dev
npm install babel-loader --save-dev
npm install babel-preset-es2015 --save-dev

//安装html loacer 的开发依赖
npm install html-loader --save-dev
```

4. 目录结构

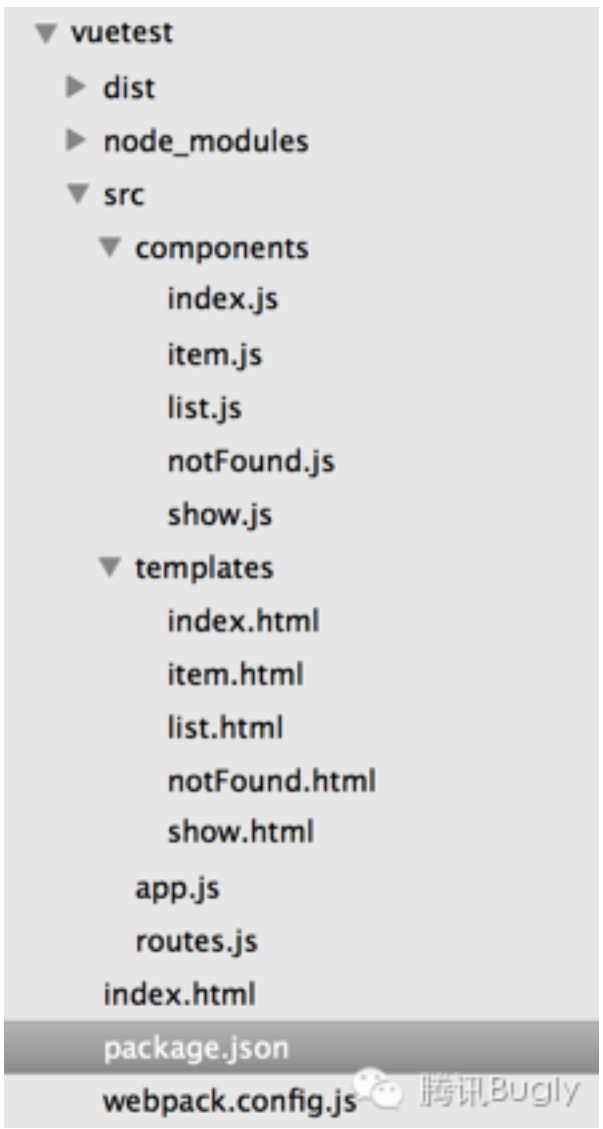
src 为开发目录，其中 components 为组件子目录，templates 为模板子目录。

dist 为构建出的文件目录。

index.html 为入口文件。

package.json 为项目描述文件，是刚才 npm init 所建立。

webpack.config.js 是 webpack 的构建配置文件



5. Webpack 配置

下面是 webpack 的配置文件，如何使用 webpack，请移步 webpack 的官网。

```
var webpack= require("webpack");

module.exports={
        entry:{
                bundle:[ "./src/app.js"]
        },
        output:{
                path:__dirname,
                publicPath:"/",
                filename:"dist/[name].js"
        },
        module:{
                loaders:[
                    {test: /\.html$/, loaders: ['html']},
                        {test: /(\.js)$/, loader:["babel"] ,exclude:/node_modules/,
                         query:{
                                        presets:["es2015"]
                         }
                    }
                ]
        },
        resolve:{
        },
        plugins:[
                 /*
                 new webpack.optimize.UglifyJsPlugin({
                        compress: {
                                warnings: false
                        }
                 })
                */
        ]
}
```

6. 入口文件

index.html

```
<!doctype html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Vue Router Demo</title>
</head>
<body>
   <div id="app">
     <router-view></router-view>
    </div>
    <script src="dist/bundle.js"></script>
</body>
</html>
```

其中 id 为 app 的 div 是页面容器，其中的 router-view 会由 vue-router 去渲染组件，讲结果挂载到这个 div 上。

app.js

```
var Vue = require('vue');
var VueRouter = require('vue-router');

Vue.use(VueRouter);
Vue.config.debug = true;
Vue.config.delimiters = ['${', '}']; // 把默认的{{ }} 改成ES6的模板字符串 ${ }
Vue.config.devtools = true;

var App = Vue.extend({});
var router = new VueRouter({});

router.map(require('./routes'));
router.start(App, '#app');
router.go({"path":"/"});
```

这是 vue 路由的配置。 其中由于习惯问题，我把 vue 默认的{{ }} 改成了的 ${ } ,总感觉这样看模板，才顺眼一些。

routes.js

```
module.exports = {
  '/': {
    component: require('./components/index')
  },
   '/list': {
    component: require('./components/list')
  },
  '*': {
    component: require('./components/notFound')
  }
}
```

7. 第一个组件

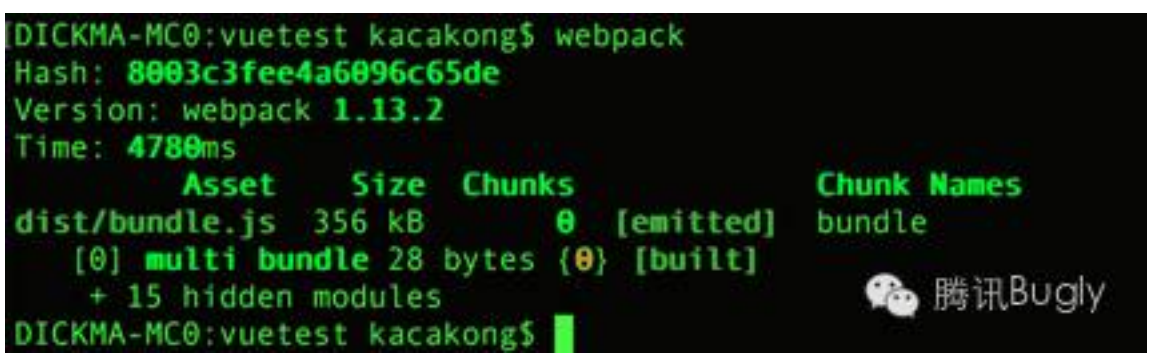
components/index.js

```
module.exports = {
  template: require('../templates/index.html'),

  ready: function () {
  }
};
```

templates/index.html

```
<h1>Index</h1>
<hr/>
<p>Hello World Index!</p>
```

执行 webpack 构建命令



浏览器中访问：

查看 bundle 源码：

发现 template 模板文件，已经被 webpack 打成字符串了。这其中，其实是 webpack 的 html-loader 起的作用

8. 组件之间跳转

修改刚才的 index 组件，增加一个跳转链接，不用 href 了，要用 vue 的指令 v-link。

```
<h1>Index</h1>
<hr/>
<p>Hello World Index!</p>
<p><a v-link="{path:'/list'}" >List Page</a></p>
```

添加 list 组件

components/list.js

```
module.exports = {
  template: require('../templates/list.html'),

  data:function(){
        return {items:[{"id":1,"name":"hello11"},{"id":2,"name":"hello22"}]};
  },
  ready: function () {
  }
};
```

templates/list.html

```
<h1>List</h1>
<hr/>

<p>Hello List Page!</p>
<ul>
        <li v-for="(index,item) in items">
             ${item.id} : ${item.name}
        </li>
</ul>
```

v-for 也是 vue 的默认指令，是用来循环数据列表的。

现在开始执行 webpack --watch 命令进行监听，这样就不用每次敲 webpack 命令了。只要开发者每次修改 js 点了保存，webpack 都会自动构建最新的 bundle 文件。

```
DICKMA-MC0:vuetest kacakong$ webpack --watch
Hash: c06e8ca3532d87e8e4a2
Version: webpack 1.13.2
Time: 3345ms
         Asset     Size  Chunks             Chunk Names
dist/bundle.js  356 kB       0  [emitted]  bundle
   [0] multi bundle 28 bytes {0} [built]
    + 15 hidden modules
```


浏览器里试试看：

index 页

点击 List Page 跳转到 list 页

Bingo！ 单页面两个组件之间跳转切换成功！

9. 组件生命周期

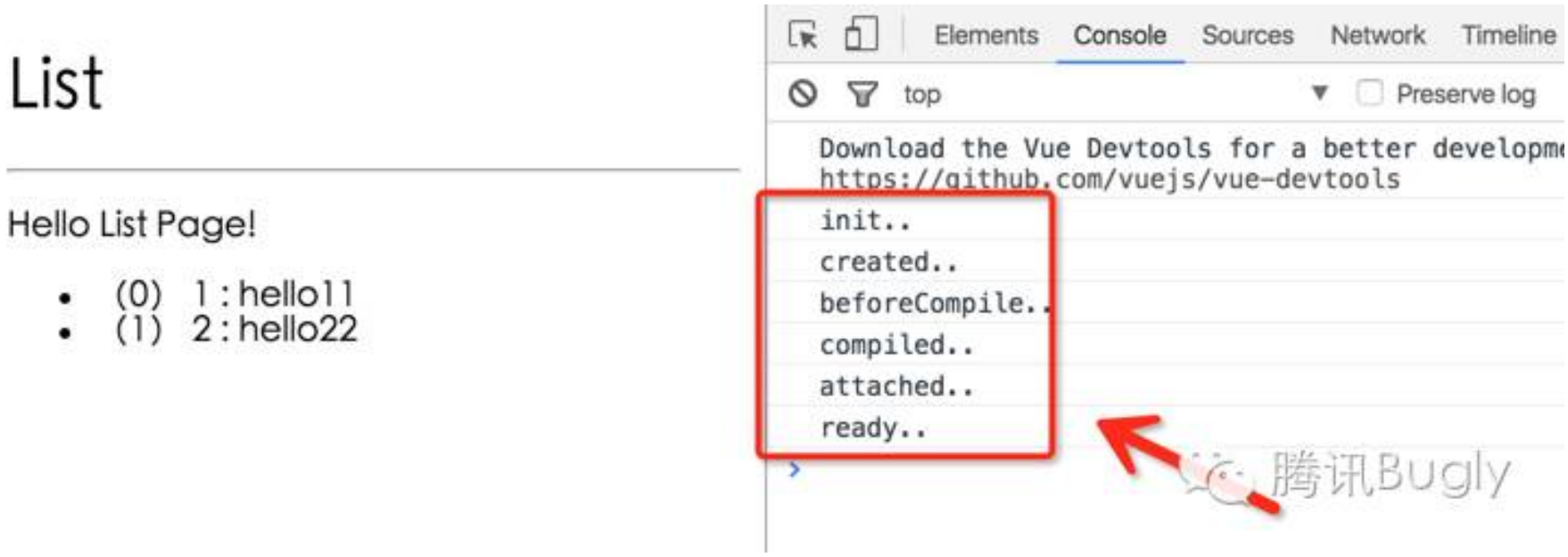
修改 **componets/list.js **

```
module.exports = {
  template: require('../templates/list.html'),

  data:function(){
        return {items:[{"id":1,"name":"hello11"},{"id":2,"name":"hello22"}]};
  },

  //在实例开始初始化时同步调用。此时数据观测、事件和 watcher 都尚未初始化
  init:function(){
        console.log("init..");
  },

  //在实例创建之后同步调用。此时实例已经结束解析选项，这意味着已建立：数据绑定，计算属性，方法，watcher/事件回调。但是还没有开始 DOM 编译，$el 还不存在。
  created:function(){
        console.log("created..");
```

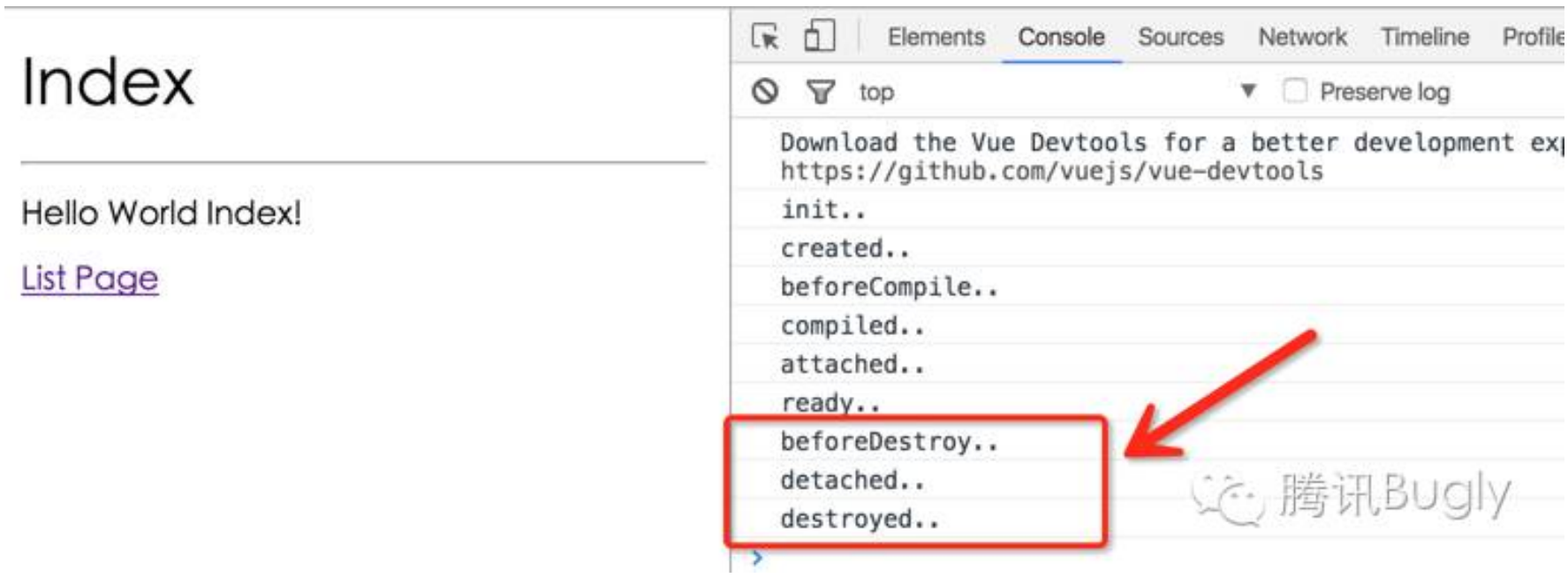
```
  },

  //在编译开始前调用。
  beforeCompile:function(){
         console.log("beforeCompile..");
  },

  //在编译结束后调用。此时所有的指令已生效，因而数据的变化将触发 DOM 更新。但是不担保 $el 已插入文档。
  compiled:function(){
         console.log("compiled..");
  },

   //在编译结束和 $el 第一次插入文档之后调用，如在第一次 attached 钩子之后调用。注意必须是由 Vue 插入（如 vm.$appendTo() 等方法或指令更新）才触发 ready 钩子。
  ready: function () {
    console.log("ready..");

  },

  //在 vm.$el 插入 DOM 时调用。必须是由指令或实例方法（如 $appendTo()）插入，直接操作 vm.$el 不会 触发这个钩子。
  attached:function(){
         console.log("attached..");
  },

  //在 vm.$el 从 DOM 中删除时调用。必须是由指令或实例方法删除，直接操作 vm.$el 不会 触发这个钩子。
  detached:function(){
         console.log("detached..");
  },

  //在开始销毁实例时调用。此时实例仍然有功能。
  beforeDestroy:function(){
         console.log("beforeDestroy..");
  },

  //在实例被销毁之后调用。此时所有的绑定和实例的指令已经解绑，所有的子实例也已经被销毁。如果有离开过渡，destroyed 钩子在过渡完成之后调用。
  destroyed:function(){
         console.log("destroyed..");
  }

};
```

在浏览器里执行了看看：

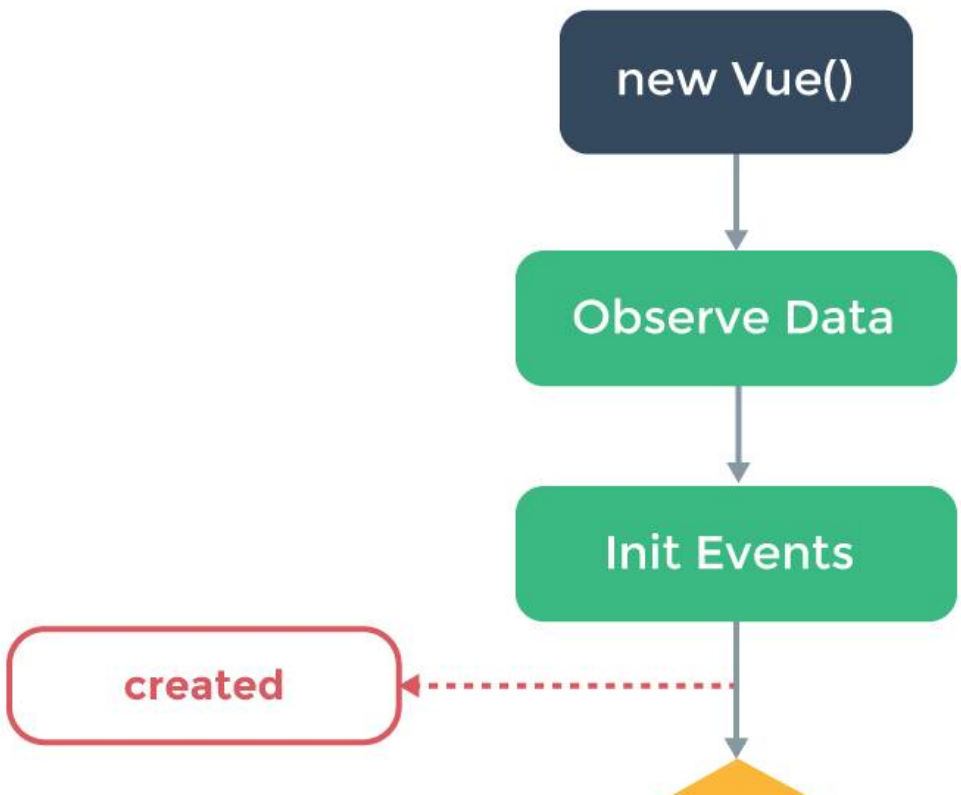
首次进入 List 页面的执行顺序如下：

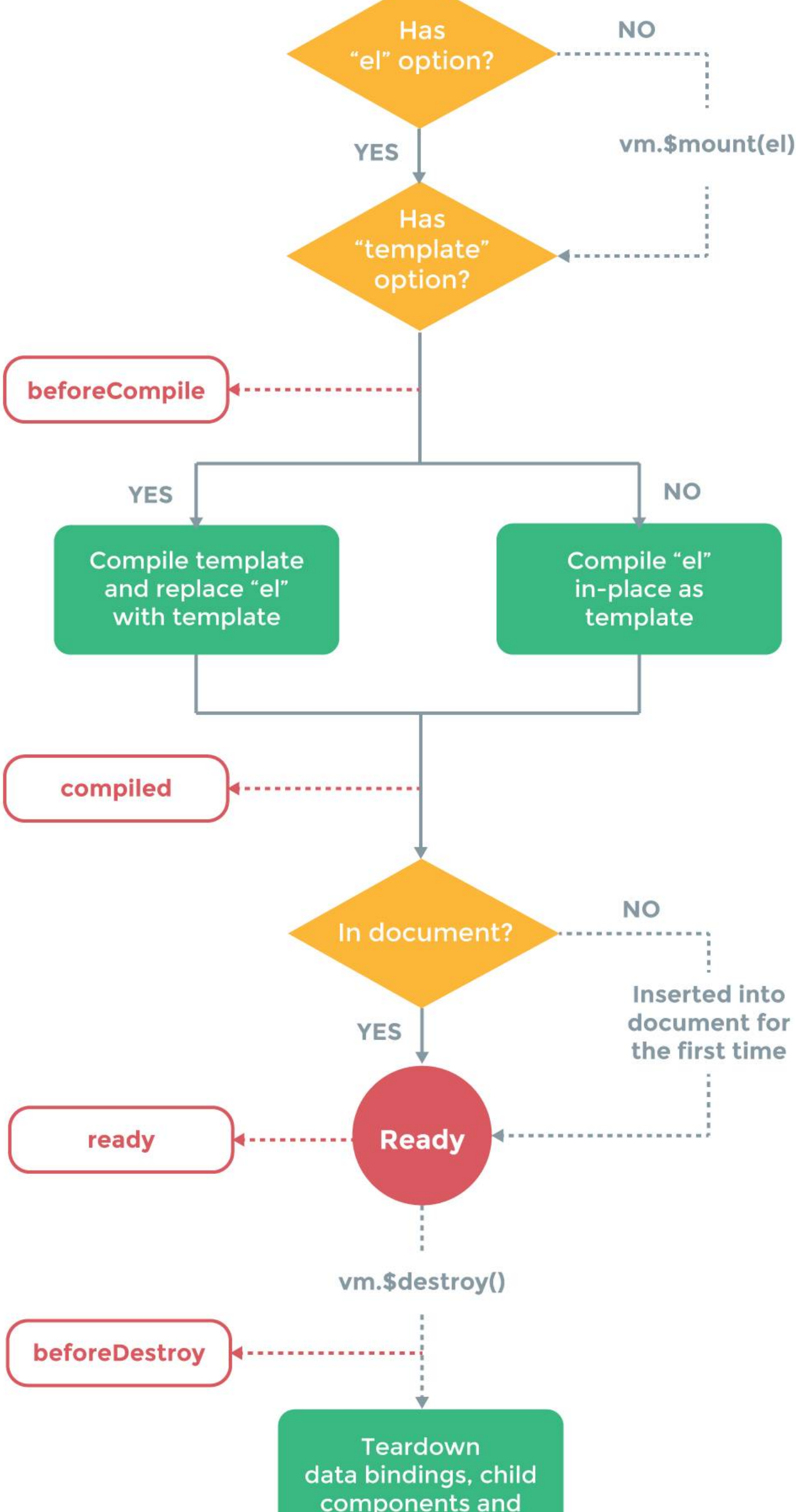


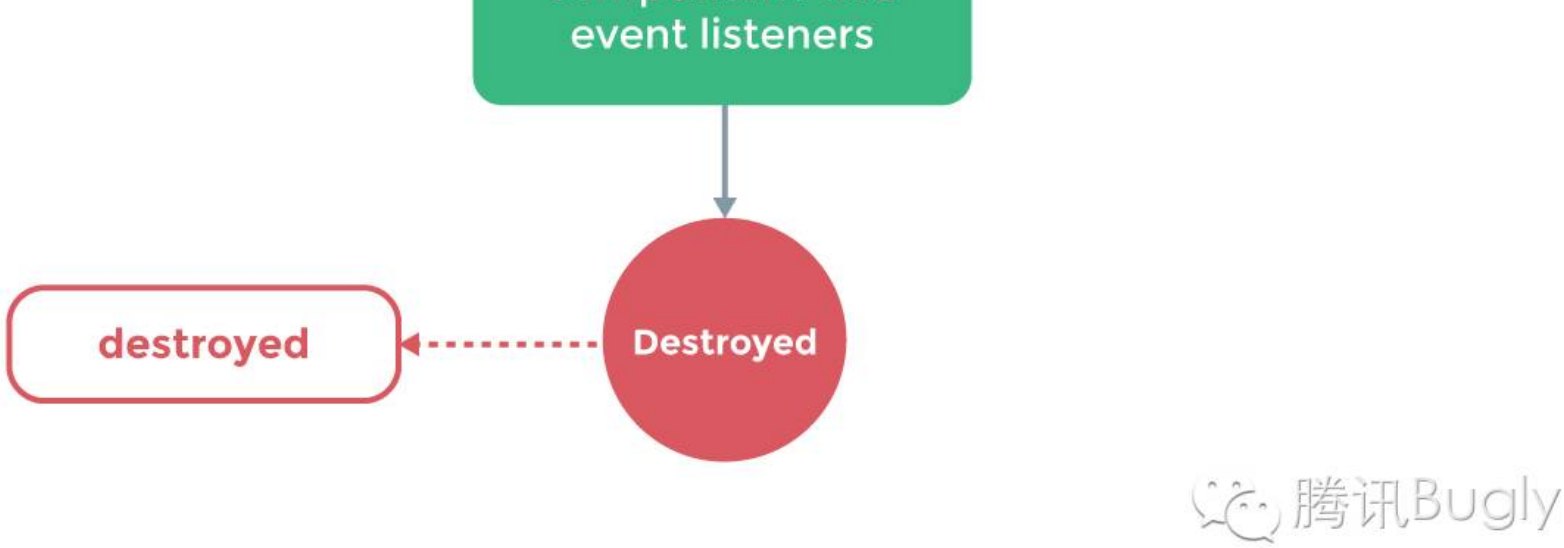
此时点一下浏览器的后退，List Component 会被销毁，执行顺序如下：



这是官方的生命周期的图：




Has
“el” option?
NO
YES
vm.$mount(el)
Has
“template”
option?
beforeCompile
YES
NO
Compile template
and replace “el”
with template
Compile “el”
in-place as
template
compiled
In document?
NO
YES
Inserted into
document for
the first time
ready
Ready
vm.$destroy()
beforeDestroy
Teardown
data bindings, child
components and



10. 父组件与子组件

在很多情况下，组件是有父子关系的，比如 list 列表组件有个子组件 item

components/item.js

```
module.exports = {
  template: require('../templates/item.html'),

  props:["id","name"],

  ready: function () {

  }
};
```

templates/item.html

```
<p>我是subitem:  ${id} - ${name}</p>
```

修改 list 组件，添加 item 的引用

components/list.js

```
//引用item组件
import item from "./item";

module.exports = {
  template: require('../templates/list.html'),

  data:function(){
        return {items:[{"id":1,"name":"hello11"},{"id":2,"name":"hello22"}]};
  },

 //定义item组件为子组件
  components:{
     "item":item
  },

  ready: function () {
  }

};
```

templates/list.html

```
<h1>List</h1>
<hr/>
<p>Hello List Page!</p>
<ul>
        <li v-for="(index,item) in items">
            <!--使用item子组件，同时把id,name使用props传值给item子组件-->
            <item v-bind:id="item.id" v-bind:name="item.name"></item>
        </li>
</ul>
```

浏览器里试试看：

子组件成功被调用了

11. 组件跳转传参

组件之间的跳转传参，也是一种非常常见的情况。下面为列表页，增加跳转到详情页的跳转，并传参 id 给详情页

修改路由 routes.js

```
module.exports = {

  '/': {
    component: require('./components/index')
  },
   '/list': {
    component: require('./components/list')
  },

  //增加详情页的跳转路由，并在路径上加上id传参，具名为name：show
   '/show/:id': {
      name:"show",
      component: require('./components/show')
    },
  '*': {
    component: require('./components/notFound')
  }
}
```

添加组件 show

components/show.js

```
module.exports = {
  template: require('../templates/show.html'),

  data:function(){
        return {};
  },

  created:function(){
        //获取params的参数ID
        var id=this.$route.params.id;

        //根据获取的参数ID，返回不同的data对象（真实业务中，这里应该是Ajax获取数据）
        if (id==1){
                this.$data={"id":id,"name":"hello111","age":24};
        }else{
       this.$data={"id":id,"name":"hello222","age":28};
        }
  },

  ready: function () {
        console.log(this.$data);
  }
};
```

templates/show.html

```
<h1>Show</h1>
<hr/>

<p>Hello show page!</p>

<p>id:${id}</p>
<p>name:${name}</p>
<p>age:${age}</p>
```

修改 templates/item.html

```
<p>我是subitem:  <a v-link="{name:'show',params: { 'id': id } }"> ${id} : ${name}</a> </p
>
```

这里 name:‘show’ 表示具名路由路径，params 就是传参。

继续浏览器里点到详情页试试：

点击“hello11”，跳转到详情页：

传参逻辑成功。

12. 嵌套路由

仅有路由跳转是远远不够的，很多情况下，我们还有同一个页面上，多标签页的切换，在 vue 中，用嵌套路由，也可以非常方便的实现。

添加两个小组件

components/tab1.js

```
module.exports = {
  template: "<p>Tab1 content</p>"
};
```

components/tab2.js

```
module.exports = {
  template: "<p>Tab2 content</p>"
};
```

修改 components/index.js 组件，挂载这两个子组件

```
import tab1 from "./tab1";
import tab2 from "./tab2";

module.exports = {
  template: require('../templates/index.html'),

  components:{
     "tab1":tab1,
     "tab2":tab2
  },

  ready: function () {

  }
};
```

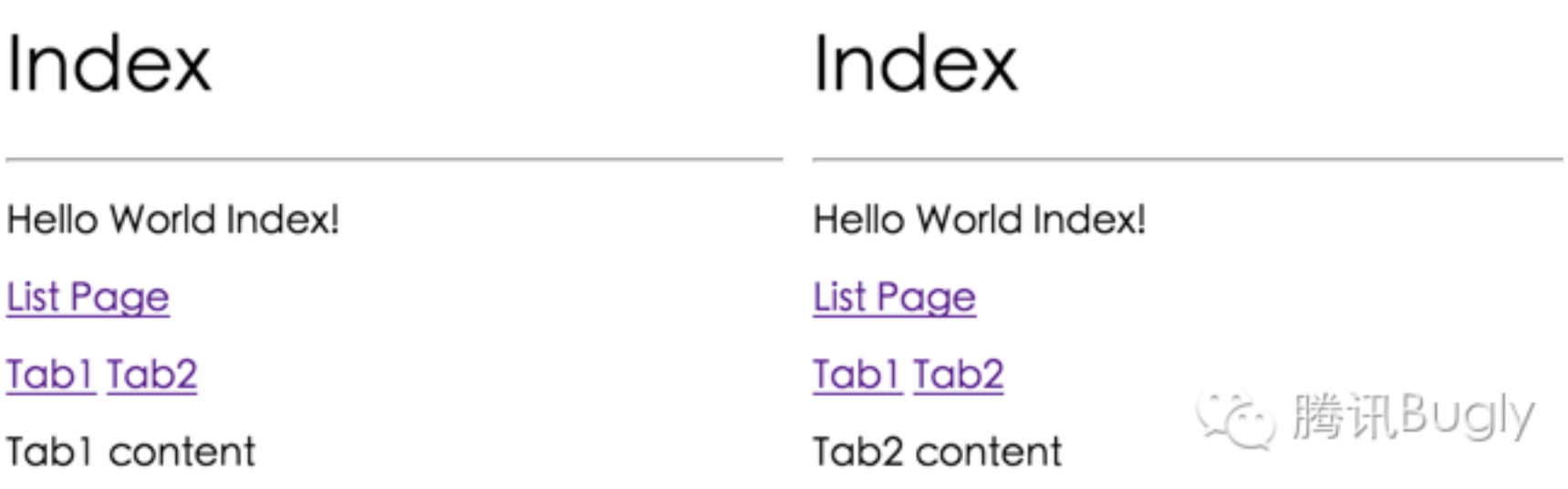
在路由里加上子路由

```
module.exports = {

  '/': {
    component: require('./components/index'),

    //子路由
    subRoutes:{
      "/tab1":{
          component:require('./components/tab1')
      },
      "/tab2":{
          component:require('./components/tab2')
      }
    }
  },

   '/list': {
    component: require('./components/list')
  },

   '/show/:id': {
      name:"show",
      component: require('./components/show')
    },

  '*': {
    component: require('./components/notFound')
  }

}
```

好了，在浏览器里试一下：

初始状态：

点了 tab1，tab2：



Tab 切换没问题，可是，初始状态显示是空的，能不能默认显示 Tab1 Content 呢？很简单，调整下路由就可以了：

```
module.exports = {

  '/': {
    component: require('./components/index'),

    //子路由
    subRoutes:{
     //默认显示Tab1
      "/":{
          component:require('./components/tab1')
      },
      "/tab1":{
          component:require('./components/tab1')
      },
      "/tab2":{
          component:require('./components/tab2')
      }
    }
  }
}
```

13. 业界 vue 使用案例

小米移动官网：http://m.mi.com/1/#/index (http://m.mi.com/1/#/index)

饿了吗招聘：https://jobs-mobile.ele.me/#!/ (https://jobs-mobile.ele.me/#!/)

4 回复


(/user/778206) **deiphi (/user/778206)** 1楼•1 年前

请问大神，最后嵌套路由这里能不能详细点，首页index.html 里面的tab1和tab1的链接要怎么写呢？ 我写成Tab1Tab2并没有测试成功。 感谢!

原来还要在tempates/index.html 里面写 <router-view></router-view>，懂了，谢大神。


(/user/780596) **EndaYuan (/user/780596)** 2楼•1 年前

当我执行　webpack 时，得到 Error

/usr/local/lib/node_modules/webpack/lib/RuleSet.js:143
　　throw new Error(“options/query cannot be used with loaders”);
　　^

Error: options/query cannot be used with loaders

麻烦指点一下，谢谢


(/user/865223) **lucine (/user/865223)** 3楼•1 年前

图片都裂了，看不了啊


(/user/1076082) **Leon_ (/user/1076082)** 4楼•1 年前

请问小米移动官网商品详情里面的为您推荐的商品， 点击的时候刷新页面是用router.push跳转还是另外写个ajax请求去修改页面内容?