1. 课程介绍

* 1. 组件(掌握)
* 2. 路由 (掌握)
* 3. vue-cli-脚手架
* 4. ElementUI（掌握）

1. 组件
   1. 什么是组件呢?

组件 (Component) 是 Vue.js 最强大的功能之一。组件可以扩展 HTML 元素，封装可重用的代码。在较高层面上，组件是自定义元素，Vue.js 的编译器为它添加特殊功能。在有些情况下，组件也可以表现为用 is 特性进行了扩展的原生 HTML 元素。

总结:

组件是用来完成特点功能的一个自定义的HTML标签  
 例如:

1. <body>
2. <mytag></mytag>
3. </body>

注意: mytag就是一个组件, 该组件必须通过Vue为mytag赋予一定的功能

* 1. 组件的作用

组件是对特点功能代码(html,css,js)的封装, 通过组件的名字可以重复利用该组件中的代码.

* 1. 组件分类

全局组件和局部组件

* + 1. 全局组件

1全局组件的语法:  
 Vue.component("自定义标签的名字",{配置对象})  
  
2. 全局组件的特点:  
 2.1 全局组件可以在任何被挂着的标签中使用.  
 2.2 全局组件的配置对象中必须包含template属性  
  
3. 注意事项:  
 template中的html必须在一个标签中. 仅且只能有一个根节点.  
  
4. 全局组件应用场景  
 如果该组件的特定功能需要在任何被挂着的标签中使用. 推荐使用全局组件

1. <div id="app">
2. <mycomponent1></mycomponent1>
3. <mycomponent2></mycomponent2>
4. </div>
5. <div id="app1">
6. <mycomponent1></mycomponent1>
7. <mycomponent2></mycomponent2>
8. </div>
9. <script type="text/javascript" src="../js/vue.js"></script>
10. <script type="text/javascript">
11. //>>1. 定义第一个全局组件
12. Vue.component("mycomponent1",{
13. template : "<h1>这是第一个全局组件</h1>"
14. })
15. //>>2. 定义第二个全局组件
16. var component2Config = {
17. template : "<h1>这是第二个全局组件</h1>"
18. };
19. Vue.component("mycomponent2",component2Config);
20. var app = new Vue({
21. el: "#app",
22. data: {
23. }
24. });
25. var app1 = new Vue({
26. el: "#app1",
27. data: {
28. }
29. });
30. </script>
    * 1. 局部组件

1. 局部语法:  
 var app = new Vue({  
 el: "#app",  
 data: {},  
 components : {  
 "局部组件的名字1" : {组件的配置对象}  
 "局部组件的名字2" : {组件的配置对象}  
 }  
 });  
  
2. 局部组件的特点  
 局部组件只能够在所挂载的标签中使用.

1. <div id="app1">
2. <mycomponent></mycomponent>
3. </div>
4. <div id="app2">
5. <mycomponent></mycomponent>
6. </div>
7. <script type="text/javascript" src="../js/vue.js"></script>
8. <script type="text/javascript">
9. //>>1. 在id=app1挂载的实例的components中定义局部组件
10. var app1 = new Vue({
11. el: "#app1",
12. data: {},
13. components : {
14. "mycomponent" : {
15. template : "<h1>这是一个局部组件</h1>"
16. }
17. }
18. });
19. //>>2. 在id=app2的标签中是不能够使用上面app2对象的局部组件.
20. var app2 = new Vue({
21. el: "#app2",
22. data: {}
23. });
24. </script>
    1. 组件使用两种HTML模板

两种方式  
 1 直接在template属性写上html代码字符串  
 template: "html代码的字符串"  
  
 2 将当前网页中的html标签作为模板代码 (大量的HTML不用拼装HTML字符串)  
 a. 在网页中定义template标签或者script标签包含模板中需要的html模板代码  
 *<!--  
 <template id="mytemplate">  
 <h1>template标签中的html</h1>  
 </template>  
  
 <script type="text/template" id="mytemplate">  
 <h1>template标签中的html</h1>  
 </script>  
  
 -->* b. template属性上使用上门的html模板代码.  
 *<!--  
 "mycomponent2":{  
 template: "#mytemplate" , #代表找到对应的html代码作为当前组件的模板代码  
 }-->* 3.注意事项:  
 如果是script的话建议加上 type="text/template"

1. div id="app">
2. <mycomponent1></mycomponent1>
3. <mycomponent2></mycomponent2>
4. <mycomponent3></mycomponent3>
5. </div>
6. <template id="mytemplate2">
7. <h1>template标签中的html</h1>
8. </template>
9. <script type="text/template" id="mytemplate3">
10. <h1>script标签中的html</h1>
11. </script>
12. <script type="text/javascript" src="../js/vue.js"></script>
13. <script type="text/javascript">
14. var app = new Vue({
15. el: "#app",
16. data: {},
17. components:{
18. "mycomponent1":{
19. template: "<h1>这是html标签代码字符串</h1>"
20. },
21. "mycomponent2":{
22. template: "#mytemplate2"
23. },
24. "mycomponent3":{
25. template: "#mytemplate3"
26. }
27. }
28. });
29. </script>
    1. 组件中的数据必须是函数

1. 组件中数据的定义  
 语法:  
 "组件的名字":{  
 template: "",  
 data : function(){  
 return {  
 键1:值1,  
 键2:值2  
 }  
 }  
 }  
  
  
2.注意事项:  
 2.1 data数据只能够以函数的形式返回,因为函数中可用写其他的业务代码  
 2.2 只能够在各自的组件模板中使用各自的组件中的data数据  
 2.3 Vue对象中的数据不能够在组件中使用. 组件的数据也不能够在挂载的html标签上使用.

1. <div id="app">
2. {{message}}
3. <mycomponent1></mycomponent1>
4. <mycomponent2></mycomponent2>
5. <mycomponent3></mycomponent3>
6. </div>
7. <template id="mytemplate2">
8. <h1>template标签中的html--- {{message}}</h1>
9. </template>
10. <script type="text/template" id="mytemplate3">
11. <h1>script标签中的html--- {{message}}</h1>
12. </script>
13. <script type="text/javascript" src="../js/vue.js"></script>
14. <script type="text/javascript">
15. var app = new Vue({
16. el: "#app",
17. data: {
18. message: "这是app中的data数据"
19. },
20. components:{
21. "mycomponent1":{
22. template: "<h1>这是html标签代码字符串---{{message}}</h1>",
23. data :function () {
24. return {
25. message:"组件mycomponent1中的数据"
26. }
27. }
28. },
29. "mycomponent2":{
30. template: "#mytemplate2",
31. data :function () {
32. return {
33. message:"组件mycomponent2中的数据"
34. }
35. }
36. },
37. "mycomponent3":{
38. template: "#mytemplate3",
39. data :function () {
40. return {
41. message:"组件mycomponent3中的数据"
42. }
43. }
44. }
45. }
46. });
    1. 组件中使用引用数据的问题

1. 注意事项:  
 1.1 当组件返回一个全新的对象时, 组件在每次使用时都会使用各自的数据对象.  
 1.2 当组件返回一个引用的已经存在的对象时, 每个组件都会公用同一个数据对象.

1. <div id="app">
2. <mycomponent></mycomponent>
3. <mycomponent></mycomponent>
4. <mycomponent></mycomponent>
5. <mycomponent></mycomponent>
6. </div>
7. <template id="mytemplate">
8. <button @click="counter++">{{counter}}</button>
9. </template>
10. <script type="text/javascript" src="../js/vue.js"></script>
11. <script type="text/javascript">
12. var data ={ counter: 1};
13. var app = new Vue({
14. el: "#app",
15. components: {
16. "mycomponent": {
17. template: "#mytemplate",
18. /\*
19. 该data函数在组件被每次使用时都返回一个全新的对象.
20. data: function () {
21. return {
22. counter: 1
23. }
24. },\*/
25. //该data函数在组件被每次使用时公用同一个数据对象.
26. data: function () {
27. return data;
28. },
29. }
30. }
31. });
32. </script>
    1. 组件的父子关系

1. 语法:  
 Vue.component("父组件的名字",{  
 template: 'html <**子组件名字**></**子组件名字**> html',  
 components :{  
 子组件的名字: {  
 template : "子组件的模板"  
 }  
 }  
 });  
  
 定义在父组件中的组件叫做子组件  
  
2. 注意事项  
 2.1 父子模板的html都可以是一个template标签或者script标签  
 2.2 子组件只能够在父组件中使用  
 2.3.这种写法也适用于局部组件.  
 4.4.子组件中可以继续定义子组件

1. <div id="app">
2. <parent></parent>
3. </div>
4. <script type="text/javascript" src="../js/vue.js"></script>
5. <script type="text/javascript">
6. Vue.component("parent",{
7. template: "<h1>这是父模板 <child></child></h1>",
8. components:{
9. child:{
10. template: "<h1>这是子模板</h1>"
11. }
12. }
13. });
14. var app = new Vue({
15. el: "#app"
16. });
17. </script>
    1. Vue应用中的数据给顶级组件

因为每个组件都是独立的. 相互不能够默认情况下相互不能够共享数据.

语法:  
 2.1 期望数据  
 Vue.component("mycompenent",{  
 props: ["期望数据的名字1", "期望数据的名字2"],  
 template: "{{期望数据的名字1}} {{期望数据的名字2}}",  
 });  
  
 var app = new Vue({  
 el: "#app",  
 data :{  
 age :29  
 }  
 });  
  
  
 2.2 在Vue挂着的标签中为组件传递数据  
 <**mycompenent 期望数据的名字1="值1" :期望数据的名字1="值2"**></**mycompenent**>

语法解释:  
 1. 子组件要显式地用 props 选项声明它预期的数据.  
 2. 在父组件中使用子组件时, 通过标签属性的形式传递过来. 标签的数据不能够使用功能{{}},只能够使用数据绑定  
 3. 如果属性是带有中划线的, 在props中一定要使用驼峰式的写法.

1. <div id="app">
2. <mycompenent name="小花" :age="age"></mycompenent>
3. </div>
4. <script type="text/javascript" src="../js/vue.js"></script>
5. <script type="text/javascript">
6. Vue.component("mycompenent",{
7. props: ["name", "age"],
8. template: "<h1>姓名: {{name}} 年龄:{{age}}</h1>",
9. });
10. var app = new Vue({
11. el: "#app",
12. data :{
13. age :29
14. }
15. });
16. </script>
    1. 父组件的数据传递给子组件

语法:  
 Vue.component("父组件的名字",{  
 template: 'html <**子组件名字 期望数据的名字1="表达式" 期望数据的名字2="表达式"**></**子组件名字**> html',  
 components :{  
 子组件的名字: {  
 props:["期望数据的名字1","期望数据的名字2"]  
 template : "子组件的模板"  
 }  
 }  
 });  
  
 语法解释:  
 1. 子组件要显式地用 props 选项声明它预期的数据.  
 2. 在父组件中使用子组件时, 通过标签属性的形式传递过来. 标签的数据不能够使用功能{{}},只能够使用数据绑定  
 3. 如果属性是带有中划线的, 在props中一定要使用驼峰式的写法.

1. <div id="app">
2. <parent></parent>
3. </div>
4. <script type="text/javascript" src="../js/vue.js"></script>
5. <script type="text/javascript">
6. Vue.component("parent",{
7. template: "<h1>这是父模板 <child name='张三' :student-age='age'></child></h1>",
8. data : function () {
9. return {
10. age : 29
11. }
12. },
13. components:{
14. child:{
15. props:["name","studentAge"],
16. template: "<h1>这是子模板 姓名:{{name}} 年龄:{{studentAge}}</h1>"
17. }
18. }
19. });
20. var app = new Vue({
21. el: "#app",
22. });
23. console.log(app);
24. </script>



1. 路由
   1. 什么是路由

路由是负责将进入的浏览器请求映射到特定的 组件 代码中。 即决定了由谁(组件)去响应客户端请求。

* 1. 路由入门

1. <div id="app">
2. <router-link to="/product">公司产品</router-link>
3. <router-link to="/about">关于我们</router-link>
4. <hr/>
5. <router-view></router-view>
6. </div>
7. <script type="text/javascript" src="../js/vue.js"></script>
8. <script type="text/javascript" src="../js/vue-router.js"></script>
9. <script type="text/javascript">
10. //>>1.定义首页
11. var index = Vue.component("index",{
12. template :"<h1>首页</h1>"
13. });
14. //>>2.公司产品
15. var product = Vue.component("product",{
16. template :"<h1>公司产品</h1>"
17. });
18. //>>3.关于我们
19. var about = Vue.component("about",{
20. template :"<h1>关于我们</h1>"
21. });
22. var router = new VueRouter({routes:[
23. {path:"/",component:index},
24. {path:"/about",component:about},
25. {path:"/product",component:product},
26. ]});
27. var app = new Vue({
28. el: "#app",
29. router:router
30. });
31. </script>
    1. 路由嵌套

组件中定义的路由叫做嵌套路由

1. <div id="app">
2. <router-link to="/product">公司产品</router-link>
3. <router-link to="/about">关于</router-link>
4. <hr/>
5. <router-view></router-view>
6. </div>
7. <script type="text/javascript" src="../js/vue.js"></script>
8. <script type="text/javascript" src="../js/vue-router.js"></script>
9. <script type="text/javascript">
10. //>>1.定义首页
11. var index = Vue.component("index",{
12. template :"<h1>首页</h1>"
13. });
14. //>>2.公司产品
15. var product = Vue.component("product",{
16. template :"<h1>公司产品</h1>"
17. });
18. //>>3.关于我们
19. var about = Vue.component("about",{
20. template :`<div>
21. <p>
22. <router-link to='/about/'>关于我们</router-link>
23. <router-link to='/about/company'>关于公司</router-link>
24. </p>
25. <p>
26. <router-view></router-view>
27. </p></div>`
28. });
29. var us = Vue.component("us",{
30. template :"<h1>关于我们</h1>"
31. });
32. var company = Vue.component("us",{
33. template :"<h1>关于公司</h1>"
34. });
35. var router = new VueRouter({
36. routes:[
37. {path:"/",component:index},
38. {path:"/product",component:product},
39. {path:"/about",component:about,
40. children: [
41. {path : "/" , component :us},
42. {path : "company" , component :company},
43. ]
44. }
45. ]});
46. var app = new Vue({
47. el: "#app",
48. router:router
49. });
50. vue-cli
    1. 安装

npm install -g vue-cli  
或者  
cnpm install -g vue-cli

* 1. 使用

//>>3.1 创建基于wepack的模板创建的vue项目,该项目的名字为hello-vue  
vue init webpack-simple hello-vue  
//>>3.2 进入项目  
cd hello-vue  
//>>3.3 安装项目依赖的包  
npm install  
//>>3.4 启动项目运行  
npm run dev

1. 常用的Vue组件框架
   1. 饿了么-PC端

<http://element-cn.eleme.io/#/zh-CN>

* 1. 饿了么-移动端

<http://mint-ui.github.io/#!/zh-cn>

1. 作业

作业难度：☆☆☆

1、完成课堂路由部分代码，实现路由切换

2、完成课堂组件代码，实现全局组件、局部组件

3. 完成组件的数据传递（父传子）

4、使用vue-cli搭建项目工程目录

5、elementUI的使用

1. 课程总结
   1. 重点
   2. 难点
   3. 如何掌握？
2. 面试题
3. 扩展知识

常用的框架整理

<http://blog.csdn.net/zgrkaka/article/details/53897202>

单页面组件详解

<http://www.duanliang920.com/learn/web355.html>