Vue进阶

一、vue实例

1.一个基本的vue的实例

```
<head>
    <meta charset="UTF-8">
    <title></title>
</head>
<body>
    <div id="app">
    <h1>
        {{title}}
    </h1>
    <button id="btn" @click="btnclick">show</button>
    <p v-if="showFlag">显示段落</p>
    {{lowercasetitle}}
    </div>
</body>
<script src="https://cdn.bootcss.com/vue/2.5.17-beta.0/vue.min.js"></script>
<script>
    var v1 = new Vue(
        {
            el:"#app",
            data:{
                title:"This is Title",
                showFlag:false
            },
            methods:{
                btnclick:function(){
                    this.showFlag=true;
                    this.updateTitle("this is new title");
                },
                updateTitle:function(d){
                    this.title = d;

                }
            },
            computed:{
                lowercasetitle:function(){
                    return this.title.toLowerCase();
                }
            },
            watch:{
                title:function(newVal,oldVal){
                    console.log(newVal)
                }
            }
        }
```

```
        );
    </script>
```

2.新的实例

```
new Vue({
    el:"#app2"
})
```

3.一个实例改变另一个实例中的内容，通过给实例命名

在一个实例中，通过调用另一个实例的属性，来操作其属性

```
var v1 =  new Vue();
var v2 = new Vue({
        el:"#app2",
        data:{},
        methods:{
            changeTitle:function(){
                v1.title = "changed title";
            }
        }
});
```

4.在vue外面操作vue实例——操作属性

```
setTimeout(function(){
        v1.title="st title";
},2000);
```

5.调用vue实例中的方法——操作方法

```
setTimeout(function(){
            v1.btnclick();
},2000);
```

6.为vue实例设置属性

- vue中的实例属性

  ```
  v1.$data
  v1.$el
  ```

- 设置实例属性

```
        var data = {
            arg1:"arg1 value"
        };
        var v2 = new Vue({
            el:"#app2",
            data:data,
```

- 实例属性还能这么调用： v1.$data.title 相当于 v1.title

7.实例属性ref的用法：相当于是id

```
<div id="app">
    <button ref="mybtn1" id="btn" @click="btnclick">show</button>
    <button ref="mybtn2" id="btn" @click="btnclick">show</button>
</div>

        methods:{
                btnclick:function(){
                    this.$refs.mybtn1.innerText = "tttttbbbbnnnn";
                }
            }
```

8.动态绑定vue实例到页面中

mount 加载的意思

```
<div id="app3"></div>
<script>
    var v3 = new Vue({

        template:"<h1>hello</h1>"
    });
    v3.$mount("#app3");
</script>
```

二、Vue组件

万事万物皆对象

万物皆组件

1.vue组件

Vue.componet 实现全局注册

```
        <div id="app1">
            <hello></hello>
        </div>
        Vue.component("hello",{
            template:"<h1>hello</h1>"
        })
        new Vue({
            el:"#app1"
        })
```

注意： div得是vue的实例，组件创建好后，只要被vue注册过的div里面，都能使用该组件

2.vue的生命周期函数

在控制台查看各函数的调用顺序，并调用destroy，再点改变title按钮，发现改变不了，因为已被销毁

```
<html>
    <head>
        <meta charset="UTF-8">
        <title>生命周期</title>
    </head>
    <body>
        <div id="app1">
            {{title}}
            <button type="button" @click="changeTitle">change title</button>
            <button type="button" @click="destroy">destroy</button>
        </div>
    </body>
    <script src="https://cdn.bootcss.com/vue/2.5.17-beta.0/vue.min.js"></script>
    <script>
        new Vue({
            el:"#app1",
            data:{
                title:"this is title"
            },
            methods:{
                changeTitle:function(){
                    this.title= "new title";
                },
                destroy:function(){
                    this.$destroy();
                }
            },
            beforeCreate(){
                console.log("beforeCreate")
            },
            created(){
                console.log("created")
            },
            beforeMount(){
                console.log("beforeMount")
```

```
            },
            mounted(){
                console.log("mounted")
            },
            beforeUpdate(){
                console.log("beforeUpdate")
            },
            updated(){
                console.log("updated")
            },
            beforeDestroy(){
                console.log("beforeDestory")
            },
            destroyed(){
                console.log("destory")
            }
        })
    </script>
</html>
```

3.一个页面中只有一个div，其他的都是vue组件

vue组件里的data必须使用function的形式对{}对象进行封装，防止对其他数据的修改。 注意，template里必须有一个根节点。

```
    <head>
        <meta charset="UTF-8">
        <title>component</title>
    </head>
    <body>
        <div id="app1">
            <my-cap></my-cap>
        </div>
        <div id="app2">

        </div>
    </body>
    <script src="https://cdn.bootcss.com/vue/2.5.17-beta.0/vue.js"></script>
    <script>
        new Vue({
            el:"#app2",
            template:"<div>aaa</div>"
        });

        Vue.component("my-cap",{
            data(){
                return {
                    status:"hellllllllo"
                }
            },

            template:'<p>{{status}}<button @click="changebtn">btn</button></p>',
```

```
            methods:{
                changebtn:function(){
                    this.status = "enennnnnnn"
                }
            }

        });
        new Vue({
            el:"#app1"
        })
    </script>
```

改进版：提高了代码的重用性——万物皆组件

```
    var cmp = {
            data(){
                return {
                    status:"hellllllllo"
                }
            },
            template:'<p>{{status}}<button @click="changebtn">btn</button></p>',
            methods:{
                changebtn:function(){
                    this.status = "enennnnnnn"
                }
            }
        }
        new Vue({
            el:"#app1",
            components:{
                "my-cap":cmp
            }
        })
        new Vue({
            el:"#app2",
            components:{
                "cap":cmp
            }

        })
```

三、vue开发模式

1.vue-cli骨架

CLI（command line interfaces ）命令行接口。在进行Vue项目开发时，可以选择不同的Vue模板进行项目的搭建，比如simple、webpack-simple、webpack、browserify/browserify-simple等；vue-cli是官方提供的一个脚手架（预先定义好的目录结构及基础代码，咱们在创建 Maven 项目时可以选择创建一个骨架项目，这个骨架项目就是脚手架），用于快速生成一个 vue 的项目模板。

2.详细步骤

- 下载安装node.js https://nodejs.org/en/download/
- 在node.js的cmd组件（command prompt）中安装vue-cli

```
npm install vue-cli -g
```

- 创建vue项目文件夹并打开

```
mkdir d:/vuedemo

cd d:/vuedemo
```

- 使用vue list命令查看当前可用的vue骨架
- 使用vue命令创建基于vue-webpack-simple骨架的项目,vue-cli-demo是项目名，过程中需要输入一些参数，回车是使用提示的值

```
vue init webpack-simple vue-cli-demo
```

- 创建基于webpack骨架的项目

```
vue init webpack vue-cli-demo
```

- 查阅readme.md文档

```
npm install  #安装依赖环境

npm run dev  #进入开发模式

npm run build #发布项目
```

3.webpack-simple模板初体验

- 1)进入到项目文件中，安装依赖环境

```
npm install
```

- 2)进入开发模式

```
npm run dev
```

如果提示缺少依赖东西，尝试重新 npm install，如果缺少node-sass，执行以下内容：

```
npm install node-sass
```

如果npm命令无法使用，可以使用淘宝镜像http://npm.taobao.org/

操作步骤如下：

```
# 1. 安装cnpm
npm install -g cnpm --registry=https://registry.npm.taobao.org
# 2. 使用cnpm代替npm
cnpm install
cnpm run dev
cnpm run build
```

- 开始组件开发 修改主组件App.vue

```
<template>
  <div>
    {{title}}
  </div>
</template>

<script>
export default {
  data(){
    return {
        title:"hellooooooooo!"
    }
  }
}
</script>
```

四、编写app.vue

1.注意template必须有一个根节点

这样会报错

```
<template>
  <div>
    {{title}}
  </div>
  <span>
  </span>
</template>
```

2.在App.vue组件中使用另一个vue组件

- 新建Home.vue

```
<template>
    <div>
    label: {{label}}
    <button @click="changeLable">bttttttttn</button>
    </div>
</template>
<script>
export default {
    data(){
        return {
            label:"i am a label"
        }
    },
    methods:{
        changeLable(){
            this.label = "yes a label!"
        }
    }
}
</script>
<style>
</style>
```

- 修改main.js

定义Home.vue为组件，并设置其标签名字

```
import Vue from 'vue'
import App from './App.vue'
import Home from './Home.vue'

Vue.component("app-server-home",Home)

new Vue({
  el: '#app',
  render: h => h(App)
})
```

- 修改App.vue

直接使用在main.js中定义的组件的标签名

```
<template>
  <app-server-home></app-server-home>
</template>
```

3.组件中嵌套组件

app内使用home，home内使用subcontent。并在home中本地注册subcontent组件。

```
<template>
    <div>
        <app-server-subcontent v-for="(sc,i) in 5":key="i">
        </app-server-subcontent>
    </div>
</template>
<script>
//本地注册
import SubContent from "./SubContent.vue"
export default {
    components:{
        "app-server-subcontent":SubContent
    }
}
</script>
```

五、在项目中使用bootstrap

1.在index.html中引入bootstrap.css

```
<head>
    <meta charset="utf-8">
    <title>vuedemo2</title>
    <link rel="stylesheet"
href="https://cdn.jsdelivr.net/npm/bootstrap@3.3.7/dist/css/bootstrap.min.css" />
  </head>
```

2.编写组件，组件中使用bootstrap

编写header、footer、content组件

- header组件

```
<template>
    <div class="col-xs-12">
        <header>
            This is Header!!!
        </header>
    </div>
</template>
```

- footer组件

```
<template>
    <div class="col-xs-12">
        <footer>
        All Right Copy
        </footer>
    </div>
</template>
```

- content组件

```
<template>
    <div class="col-xs-12 col-sm-6">
        <ul class="list-group">
            <li class="list-group-item" v-for="(s,i) in 5" :key="i">
                {{s}}
            </li>
        </ul>
    </div>
</template>
```

3.在app.vue中本地注册这些组件并使用

```
<template>
  <div id="app">
    <app-header></app-header>
    <app-content></app-content>
    <app-footer></app-footer>
  </div>
</template>

<script>
    //内部注册
import Footer from  "Footer.vue"
import Header from "Header.vue"
import Content from "Content.vue"
export default {
    components:{
        "app-footer":Footer,
        "app-header":Header,
        "app-content":Content
    }
}
</script>
```

六、组件的全局注册

在main.js中注册组件，该组件能够在项目中的每个Vue中被使用

```
import Home from './Home.vue';
Vue.component("appHome",Home);
```

七、在组件中使用样式

在组件中定义样式，组件里的样式会影响所有组件，如何设定，让样式只作用于当前组件?

content组件中的style将会影响整个页面的元素，因此在组件的style标签里加入scoped关键字，让该样式只作用于当前组件。

```
<template>
    <div class="col-xs-12 col-sm-6">
        <ul class="list-group">
            <li class="list-group-item" v-for="(s,i) in 5" :key="i">
                {{s}}
            </li>
        </ul>
    </div>
</template>

<script>
</script>

<style scoped>
    div{
        border: 1px solid red;
    }
</style>
```

八、各组件中的参数传递

1.父传子

通过父vue的标签属性传递参数:

- App.vue

```
<template>
  <div id="app">
    <sub-app :myName="name"></sub-app>
  </div>
</template>

<script>
    //内部注册
import Sub1 from "@/subapp/sub1.vue"
export default {
    data(){
        return {
            name:"xiaoyu"
        }
```

```
    },
    components:{
        subApp:Sub1
    }
}
</script>
```

- sub1.vue

```
<template>
    <div>
        <h1>{{myName}}</h1>
    </div>
</template>

<script>
    export default{
        props:['myName']
    }
</script>
```

props后存放数组，表示可以拿多个参数。也可以改写成一个对象：

```
<template>
    <div>
        <h1>{{myName}}</h1>
    </div>
</template>

<script>
    export default{
        //props:['myName']
        props:{
            myName:{
                type:String,
                required:true,
                default:'xx'
            }
        }
    }
</script>
```

2.子组件调用父组件中的函数并传递参数

- App.vue

```

<template>
```

```
  <div id="app">
    <sub-app :myName="name" :ffn="changeName"></sub-app>
  </div>
</template>

<script>
    //内部注册
import Sub1 from "@/subapp/sub1.vue"
export default {
    data(){
        return {
            name:"xiaoyu"
        }
    },
    methods:{
        changeName:function(name){
            this.name = name
        }
    },
    components:{
        subApp:Sub1
    }
}
</script>
```

- sub1.vue

```
<template>
    <div>
        <h1>{{myName}}</h1>
        <button type="button" @click="doClick" >点我</button>
    </div>
</template>

<script>
    export default{
        props:{
            myName:{
                type:String,
                required:true,
                default:'xx'
            },
            ffn:{
                type:Function
            }
        },
        methods:{
            doClick:function(){
                this.ffn("xiaohe");
            }
        }

    }
```

```
</script>
```

3.改进版的参数传递(常用)

从下往上的事件发射

- sub1.vue

```
doClick:function(){
        //this.ffn("xiaohe");
        this.$emit('newName','xiaoliu');
}
```

- App.vue

```
 <sub-app :myName="name" @newName="name=$event"></sub-app>
```