前端框架的三国时代-Vue.js(三)

1.组件化开发

我们可以很直观的将一个复杂的页面分割成若干个独立组件,每个组件包含自己的逻辑和样式 再将这些独立组件组合完成一个复杂的页面。 这样既减少了逻辑复杂度，又实现了代码的重用。 页面是组件的容器，组件自由组合形成完整的界面，当不需要某个组件时，或者想要替换某个组件时，可以随时进行替换和删除，而不影响整个应用的运行。

组件化开发的好处

- 提高开发效率
- 方便重复使用
- 便于协同开发
- 更容易被管理和维护

2.Vue中的组件

vue中的组件是自定义的标签，可以扩展原生的html元素，封装可重用的代码

组件的组成

- 样式结构
- 逻辑行为
- 数据绑定

3.定义组件

全局定义

- 注册后在任意实例的模版中均可使用。

```
<div id="app">
  <zf-component></zf-component>
</div>
Vue.component('zf-component',{
  template:`<div>hello vuejs</div>`
});
```

局部定义

- 在组件实例中通过选项注册，只有在所注册的作用域中使用

```
var vm = new Vue({
  el:'#app',
  components:{
      'zf-component':{
          template:`<div>hello vuejs</div>`
      }
  }
})
```

> 在html中使用组件需要使用短横线隔开(kebab-case)命名法

4.组件中的数据

- 每个组件都是相互独立的，如果共用一个对象，在更改某个组件时会影响其他组件，如果是函数返回自己独立的数据，就可以实现互不影响。所以vue中组件的数据都是函数形式

```
var vm = new Vue({
  el:'#app',
  components:{
      'zf-component':{
          template:`<div>hello {{language}}</div>`,
          data(){
              return {language:'vuejs'}
          }
      },
  }
})
```

5.组件间的通信

父组件要将数据传递给子组件，子组件要将内部发生的事情通知父组件

父组件->子组件

不能在子组件中直接调用父组件中的数据，需要通过父组件的属性传递，并且子组件要显示的通过props声明传递过来的属性。

```
<zf-component :language="language"></zf-component>
components:{
    'zf-component':{
        props:['language'],
        template:`<div>hello {{language}}</div>`,
    },
}
```

props可以进行属性校验

```
props:{
    msg:{
        type:[Number,String],
        default:100,
        required:true,
        validator(value){
            return value>100
        }
    }
},
```

> 可以校验String,Number,Boolean,Function,Object,Array

子组件->父组件

子组件需要通过事件发射($emit)触发父组件的自定义事件

```
<zf-component @custom-event="receive"></zf-component>
var vm = new Vue({
    el:'#app',
    methods:{
        receive(data){alert(data);}
    },
    components:{
        'zf-component':{
            template:`<button @click="run">发射</button>`,
            methods:{
                run(){
                    this.$emit('custom-event','我好饿')
                }
            }
        }
    }
})
```

平级组件间的通信

- 通过事件的方式传递数据

```
var bus = new Vue();
var vm = new Vue({
  el:'#app',
  created(){
      bus.$on('child',function (data) {
          console.log(data);
```

```
        })
    },
    components:{
        child:{
            data(){
                return {child:'vuejs'}
            },
            template:'<div><button @click="up">ok</button></div>',
            methods:{
                up(){
                    bus.$emit('child',this.child);
                }
            }
        }
    }
})
```

> 模版中的html元素的根节点只能有一个。

6.单向数据流

数据从父组件流向子组件, 只能单向绑定，在子组件内不应该直接修改父组件传递的数据

```
{{language}}  <!--在这里你会发现子组件改变了数据父组件不会更新-->
<zf-component :language="language"></zf-component>
components:{
    'zf-component':{
        props:['language'],
        template:`<div>hello {{language}}
```

```
            <button @click="language='angularjs'">点我变ng</button>
        </div>`,
    },
}
```

子组件更改数据方式

- 可以作为数据的初始值使用

```
data(){
return {msg:this.language + 'hello'}
},
```

- 作为子组件的computed属性

```
computed: {
  msg(){
     return this.language + 'hello'
  }
},
```

父子组件相互影响

父组件传递的数据，在子组件中发生变化并且父组件要更新数据。

- 通过事件的方式传递给父组件

```
var vm = new Vue({
    el:'#app',
```

```
    data:{language:'vuejs'},
    methods:{
        change(data){
            this.language = data
        }
    },
    components:{
        'zf-component':{
            props:['language'],
            data(){
              return {msg:this.language}
            },
            template:`<div>hello {{msg}}
                <button @click="changeParent()">点我变ng</button>
            </div>`,
            methods:{
                changeParent(){
                    this.msg = 'angularjs';
                    this.$emit('change',this.msg);
                }
            }
        },
    }
})
```

- 父子组件使用相同data

```
{{language.language}}
<!--在这里传入对象格式，子组件修改这个对象的属性，可以导致父组件更新-->
<zf-component :language="language"></zf-component>
var vm = new Vue({
```

```
  el:'#app',
  data:{language:{language:'vuejs'}},
  components:{
      'zf-component':{
          props:['language'],
          data(){
            return {msg:this.language}
          },
          template:`<div>哈哈 {{language.language}}
              <button @click="language.language='angularjs'">更改</button>
          </div>`
      },
  }
})
```

7.slot插槽

可以在组件内定制模板,在使用组件时会将中间传递的内容替换掉定制的模板。

单个slot

```
<zf-component>
    <!--这里不传入内容时默认会使用模板中定制的内容，传入后则会覆盖掉定制的内容-->
    <div>用这个盒子替换掉slot</div>
</zf-component>
components:{
    'zf-component':{
        template:`<div>
            <slot>
                <h1>这是个小标题</h1>
```

```
                <p>这是内容</p>
            </slot>
        </div>`
    }
}
```

具名slot

```
<zf-component>
    <template slot="header">footer</template>
    <template>这是新内容</template>
    <template slot="footer">header</template>
</zf-component>
components:{
    'zf-component':{
        template:`<div>
            <slot name="header">header</slot>
            <slot>这是默认内容</slot>
            <slot name="footer">footer</slot>
        </div>`
    }
}
```

> 不指定slot名字默认名字为default,可以和具名slot同时使用

8.is模板

可以使用is指定动态模板的渲染，增加keep-alive可以保存组件上一次的状态

```
<input type="radio" v-model="com" value="com1">
<input type="radio" v-model="com" value="com2">
<keep-alive>
<component :is="com"></component>
</keep-alive>
var vm = new Vue({
    el:'#app',
    data:{
      com:'com1'
    },
    components:{
        com1:{
            data(){
                return {msg:'hello'}
            },
            template:'<div @click="changeColor">{{msg}}</div>',
            methods:{
                changeColor(){
                    this.msg = 'hello vue'
                }
            }
        },
        com2:{
            template:'<div>world</div>'
        }
    }
})
```

9.自定义panel组件

应用slot,events,props

- 使用bootstrap样式构建panel组件
- 将html封装到template中
- 在组件内部定制想要的模板
- 可以通过传入标题,模板更改组件的状态

定义模板

```
<template id="panel">
    <div class="panel" :class="[style]">
        <div class="panel-heading">
            {{header}}
        </div>
        <div class="panel-body">
            <slot name="body">这是默认内容</slot>
        </div>
        <div class="panel-footer">
            <slot name="footer">作者:Mrs Jiang</slot>
            <button v-if="flag" @click="say">点我出标题</button>
        </div>
    </div>
</template>
```

> 在模板需要传入的内容有header(标题)、flag(是否显示按钮)、style(panel颜色)

使用组件的理想用法