Vue.js_09--Vue组件

🌟Vue组件的三种创建方式
🌟Vue父组件向子组件传值

全局组件

1.
//注册组件的样式

```
var t1 = Vue.extend({
    template:'<h1>组件的第一种创建方式</h1>'
})
```

//初始化组件（组件名,组件内容）

```
Vue.component('test1',t1)
```

//或第二种方式将组件直接定义出来

```
Vue.component('test2',Vue.extend({
    template:'<h1>组件的第一种创建方式的拓展</h1>'
}))
```

2.使用自变量形式创建

```
Vue.component('test3',{
    template:'<h2>组件的第二种创建方式</h2>'
})
```

> 第一二种创建方式在template中是以字符串的方式展示，写代码时无法提示，效率不高，容易出错。

3.🌟
//创建时给一个选择器

```
Vue.component('test4',{
    template:'#content'
})
```

//在选择器中定义组件

```
<template id="content">
    <!-- 注意：组件中的模版，只能有一个根元素 -->
        <div>
            <h4>这是组件的第三种创建方式</h4>
        </div>
</template>
```

1、2、3以上三种定义全局组件的方式，最后显示方式为：

```
<!-- 使用该组件 -->
<div id="main">
    <test1></test1>
    <test2></test2>
    <test3></test3>
    <test4></test4>
</div>
```

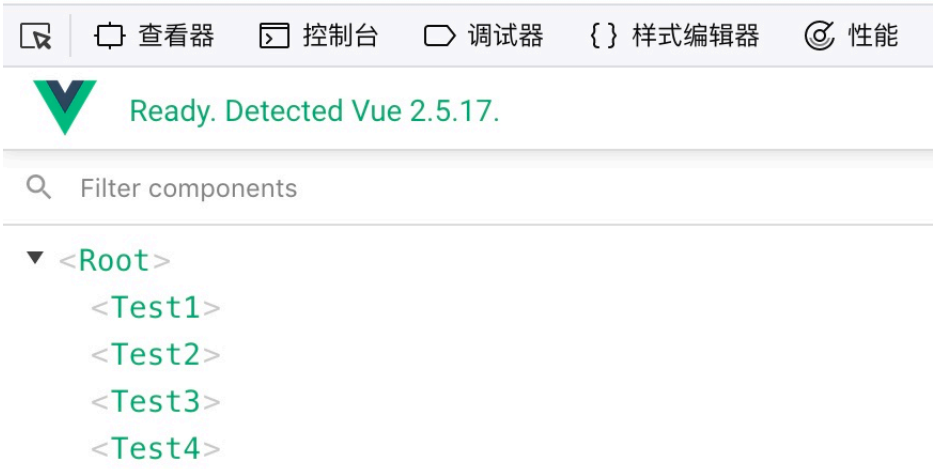
//运行效果如下：

组件的第一种创建方式

组件的第一种创建方式的拓展

组件的第二种创建方式

这是组件的第三种创建方式



注意:组件中的模版，只能有一个根元素,否则会报错。

私有组件

//将组件定义在一个新的Vue对象中

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

    },
    methods:{

    },
    //私有组件
    components:{
        //名字
        test1:{
            template:'#con'
        }
    }

})
```

//初始化

```
<template id="con">
    <div>
        <h1>私有组件</h1>
    </div>
</template>
```

//使用

```
<div id="app">
    <test1></test1>
</div>
```

组件中的data和methods

1.data

注意:对象中的data定义为数据；组件data定义为方法，并且其中需要return一个对象，如：

```
//私有组件
components:{
    //名字
    test1:{
        template:'#con',
        data:function () {
            return{

            }
        }
    }
}
```

//使用当中的msg可以直接在组件的内部调用

```
//私有组件
components:{
    //名字
    test1:{
        template:'#con',
        data:function () {
            return{
                msg:'我是组件test1中的data数据'
            }
        }
    }
}
```

调用：

```
<template id="con">
    <div>
        <h1>私有组件----{{msg}}</h1>
    </div>
</template>
```

```
<div id="app">
    <test1></test1>
</div>
```

私有组件----我是组件test1中的data数据

2.methods

注意:组件中的methods对象后面也是方法。

组件的切换

//首先简单的定义三个组件:

```
<script>
    Vue.component('test1',{
        template:'<h1>这是第一个组件</h1>'
    })

    Vue.component('test2',{
        template:'<h1>这是第二个组件</h1>'
    })

    Vue.component('test3',{
        template:'<h1>这是第三个组件</h1>'
    })

    var vm = new Vue({
        el:'#main',
        data:{
            name:'test1'
        },
        methods:{

        },
    })


</script>
```

//绑定切换事件、（设置切换动画）

```
<div id="main">
    <a href="" @click.prevent="name='test1'">组件1</a>
    <a href="" @click.prevent="name='test2'">组件2</a>
    <a href="" @click.prevent="name='test3'">组件3</a>
    <transition name="t1" mode="out-in">
        <component :is="name"></component>
    </transition>
</div>
```

Vue单向数据流---父组件向子组件传值

单向数据流

所有的 prop 都使得其父子 prop 之间形成了一个**单向下行绑定**：父级 prop 的更新会向下流动到子组件中，但是反过来则不行。这样会防止从子组件意外改变父级组件的状态，从而导致你的应用的数据流向难以理解。

额外的，每次父级组件发生更新时，子组件中所有的 prop 都将会刷新为最新的值。这意味着你**不**应该在一个子组件内部改变 prop。如果你这样做了，Vue 会在浏览器的控制台中发出警告。

来自Vue官方文档定义

```
<div id="main">
    <test v-bind:fathermsg="msg1"></test>
</div>

<template id="con">
    <div>
        <h1>这是一个私有组件---{{msg}}---{{fathermsg}}</h1>
    </div>
</template>
```