Vue第三天

今日内容介绍

- Vue1.0过渡动画的实现
- Vue2.0过渡动画的实现
- 组件定义和注册的各种写法
- 组件的动态切换
- 组件的传值

今日内容学习目标

- 知道vue动画效果的三个css的作用
- 记住vue1.0和2.0利用css的三个类来控制过渡动画的写法
- 记住vue1.0和2.0中利用animate.css来控制过渡动画写法
- 记住vue1.0和2.0中过渡动画的钩子函数使用
- 知道组件的定义和注册写法 Vue.component()
- 知道利用components实现子组件的定义
- 知道利用props实现父组件传值给子组件的写法
- 知道利用component 控制组件的切换
- 知道利用$emit和v-on来实现子组件传值给父组件
- 知道v-el与v-ref的区别和使用

详细内容

1.0 Vue过渡动画

```
    通过 Vue.js 的过渡系统，可以在元素从 DOM 中插入或移除时自动应用过渡效果。Vue.js 会在适当的时机为你触发 CSS 过渡或动画
    常用场景有：
    1、条件渲染 （使用 v-if）
    2、条件展示 （使用 v-show）
    3、动态组件
```

1.0.1 transition的作用

1、在Vue1.0版本中为了应用过渡效果，需要在实现过渡动画的元素上使用 transition 特性，示例：
 ,my-transition 可以有程序员自定义名称

2、 在Vue2.0版本中改变成了由 transition特性的写法变成了 <transition></transition>的写法
<transition name="fade">
hello
</transition>

***transition通常与下面指令结合在一起使用：
- v-if
- v-show

1.0.2 Vue中过渡动画的几种常用写法

- 利用css控制过渡动画
  - Vue1.0写法


```
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <style>
        .show-transition{
            transition:all 0.4s ease;
            padding-left: 0;
        }
        .show-enter,.show-leave{
            padding-left: 100px;
        }
    </style>
    <script src="../../vue1028.js"></script>
</head>
<body>
    <div id="app">
    <button @click="toggle">展示和隐藏</button>
        <div v-show="isshow" transition="show">
            hello vuejs
        </div>
    </div>
    <script>
        new Vue({
            el:'#app',
            data:{
                isshow:false
            },
            methods:{
                toggle:function(){
                    this.isshow = !this.isshow;
                }
            }
        });
    </script>
```

- Vue2.0写法

```
<style>
    .show-enter-active,.show-leave-active{
        transition:all 0.4s ease;
        padding-left: 0px;
    }
    .show-enter,.show-leave-active{
        padding-left: 100px;
    }
</style>
<script src="../../vue2.js"></script>
</head>
<body>
    <div id="app">
    <button @click="toggle">展示和隐藏</button>
    <transition name="show">
        <div v-show="isshow" >
            hello vuejs
        </div>
    </transition>
    </div>
    <script>
        new Vue({
            el:'#app',
            data:{
                isshow:false
            },
            methods:{
                toggle:function(){
                    this.isshow = !this.isshow;
                }
            }
        });
    </script>
```

控制动态的代码

实现动画的方式：
1、动画由100px到0进入
2、动画由0到100px离开

css样式 - 前面的名称必须与这个保持一致

- 利用animate.css控制过渡动画

- Vue1.0写法

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <link rel="stylesheet" href="../../animate.css">
    <script src="../../vue1028.js"></script>
</head>
<body>
    <div id="app">
    <button @click="toggle">展示和隐藏</button>
        <div v-show="isshow" class="animated" transition="show">
            hello vuejs
        </div>
    </div>
    <script>
        new Vue({
            el:'#app',
            data:{
                isshow:false
            },
            methods:{
                toggle:function(){
                    this.isshow = !this.isshow;
                }
            },
            transitions:{
                'show':{
                    enterClass:'fadeInRight',
                    leaveClass:'fadeOutRight'
                }
            }
        });
    </script>
```

在要实现动画的元素上加上amimate.css的类

标记这是一个要实现过渡动画的元素，名称show可以自定义

show的名称必须与目标元素中transition中的值保持一致

进入和离开的动画方式选择animate.css中的对应的动画类

- Vue2.0写法

```
![d3-4.png](imgs/d3-4.png "")
```

- 利用钩子函数控制过渡动画
  - Vue1.0 过渡动画API文档：http://v1-cn.vuejs.org/guide/transitions.html
  - Vue2.0 过渡动画API文档：http://cn.vuejs.org/v2/guide/transitions.html
- Vue1.0钩子函数

```
1、过渡动画进入
    beforeEnter:function(el){}      过渡动画进入之前，一般在这个方法中定义目标元素的初始位置
    enter:function(el,done){}       过渡动画进入中，在这个方法中定义目标元素的结束位置
    afterEnter:function(el){}       过渡动画结束后，通常在这个方法里面重置初始值
    enterCancelled:function(el){}   取消过渡动画时被调用

2、过渡动画离开
    beforeLeave:function(el){}      动画离开之前触发
    leave:function(el){}            过渡动画进入中触发
    afterLeave:function(el){}       过渡动画离开结束后
    leaveCancelled:function(el){}   取消过渡动画时被调用
```

3、使用示例：

```
![d3-5.png](imgs/d3-5.png "")
```

- Vue2.0钩子函数

```
1、过渡动画进入
    before-enter        过渡动画进入之前，一般在这个方法中定义目标元素的初始位置
    enter               过渡动画进入中，在这个方法中定义目标元素的结束位置
    after-enter         过渡动画结束后，通常在这个方法里面重置初始值
    enter-cancelled     取消过渡动画时被调用

2、过渡动画离开
    before-leave        动画离开之前触发
    leave               过渡动画进入中触发
    after-leave         过渡动画离开结束后
    leave-cancelled     取消过渡动画时被调用
```

3、使用示例：

```
    <script src="../../vue2.js"></script>
    <style>      .show{transition:all 0.4s ease;}     </style>
</head>
<body><div id="app">
    <button @click="toggle">展示和隐藏</button>
        <transition @before-enter="beforeEnter" @enter="enter" @after-enter="afterEnter">
            <div v-show="isshow" class="show">hello vuejs</div>
        <transition>
    </div>
    <script>
        new Vue({
            el:'#app',
            data:{isshow:false},
            methods:{
                toggle:function(){this.isshow = !this.isshow;},
                beforeEnter:function(el){//1.0 定义动画的初始位置
                    el.style.transform = "translate(100px,0)";
                },
                enter:function(el){//1.0 设定dom元素的刷新
                    var rf = el.offsetHeight;
                    //2.0 设置动画的结束位置
                    this.$nextTick(function(){
                        el.style.transform = "translate(0,0)";
                    });
                },
                afterEnter:function(el){
                    this.isshow = !this.isshow;/*Vue1.0 没法办修改 isshow的变量*/
                }
            }
        });
    </script>
```

2.0 Vue组件

> 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

2.0.1 组件的定义和注册

- 写法1：使用Vue.extend方法定义组件，使用 Vue.component方法注册组件

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="../vue2.js"></script>
</head>
<body>
    <div id="app">
        <account></account>
    </div>

    <script>

    //写法1:
    //1.0 定义组件
    var account =  Vue.extend({
        template:'<div><a href="#">登录111</a><a href="#">注册</a></div>'
    });
    // 2.0 全局注册组件
    Vue.component('account',account);

    new Vue({
        el:'#app'
    });
    </script>
</body>
</html>
```

二者必须保持一致

- 写法2:使用 Vue.component方法定义注册组件一步到位

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="../vue2.js"></script>
</head>
<body>
    <div id="app">
        <account></account>
    </div>

    <script>


    // 写法2:(定义和注册组件一步到位)
    Vue.component('account',{
        template:'<h3>账户组件内容: </h3><a href="#">登录</a>    <a href="#">注册</a>'
    });

    new Vue({
        el:'#app'
    });
    </script>
</body>
</html>
```

必须保持一致

- 写法3：将组件内容定义到template模板中

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="../vue2.js"></script>
</head>
<body>
    <template id="account">
        <h3>账户组件内容：</h3>
        <a href="#">登录</a>
        <a href="#">注册</a>
    </template>

    <div id="app">
        <account></account>
    </div>
    <script>
    //3.0 写法3：
    Vue.component('account',{
        template:'#account'
    });
    new Vue({
        el:'#app'
    });
    </script>
</body>
</html>
```

二者必须保持同名

- 写法4：将组件内容定义到类型为 x-template的script模板中

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="../vue2.js"></script>
</head>
<body>
    <script type="x-template" id="account">
        <h3>账户组件内容11111: </h3>
        <a href="#">登录</a>
        <a href="#">注册</a>
    </script>
    <div id="app">
        <account></account>
    </div>
    <script>
    //3.0 写法3:
    Vue.component('account',{
        template:'#account'
    });
    new Vue({
        el:'#app'
    });
    </script>
</body>
</html>
```

二者必须保持同名

2.0.2 组件中实现指令以及事件绑定

```
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="../vue1028.js"></script>
</head>
<body>
    <template id="account">
        <h3>账户组件内容：{{msg}}</h3>
        <a href="#" @click="login">登录</a>
        <a href="#">注册</a>
    </template>
    <div id="app">
        <account></account>
    </div>
    <script>
    Vue.component('account',{
        template:'#account',
        //重点注意：原来在new Vue()中定义的data是一个对象
        //但是在组件中定义的data是一个方法,并且在这个方法中一定是要return一个对象
        data:function(){
            return {
                msg:'我是account组件中的msg'
            }
        },
        methods:{
            login:function(){
                alert(this.msg);
            }
        }
    });
    new Vue({
        el:'#app'
    });
    </script>
</body>
```

组件模板

使用组件

组件定义和注册

初始化Vue

2.0.3 组件中注册子组件

```
    <title>Document</title>
    <script src="../vue1028.js"></script>
</head>
<body>
    <template id="account">
        <h3>账户组件内容：</h3>
        <a href="#">登录</a>
        <a href="#">注册</a>
        <!-- 使用login子组件，注意点：login一定是在注册组件时候的那个名字 -->
        <login></login>
        <!-- 使用register子组件 -->
        <register></register>
    </template>
    <div id="app">
        <account></account>
    </div>
    <script>
    Vue.component('account',{
        template:'#account' ,
        components:{
            //在account组件中定义和注册一个login的子组件
            'login':{
                template:'<h2>登录组件内容</h2>'
            }
        }
    });

    new Vue({
        el:'#app'
    });
    </script>
</body>
</html>
```

定义id=account的组件模板

在account组件中使用login子组件

定义和注册account组件

在account组件中定义和注册名为login的子组件

2.0.4 组件中利用component中的is来实现组件切换

```
    <script src="../vue1028.js"></script>
</head>
<body>
    <template id="account">
        <h3>账户组件内容：</h3>
        <a href="#" @click="componentName='login'">登录</a>
        <a href="#" @click="componentName='register'">注册</a>
        <!-- 利用 component 结合其中的is属性来进行动态显示子组件
        其实是通过组件的名字来进行控制这个组件内容的显示的 -->
        <component :is="componentName"></component>
    </template>
    <div id="app">
        <account></account>
    </div>
    <script>
    Vue.component('account',{
        template:'#account' ,
        data:function(){
            return {componentName:'login'  /*代表默认显示的是登录组件*/ }
        },
        components:{
            'login':{//在account组件中定义和注册一个login的子组件
                template:'<h2>登录组件内容</h2>'},
            'register':{//在account组件中定义和注册一个register的子组件
                template:'<h2>注册组件内容</h2>'}
        }
    });
    new Vue({
        el:'#app'
    });
    </script>
</body>
</html>
```

控制组件切换按钮

控制组件显示

2.0.5 实现父组件传值给子组件

```
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="../vue1028.js"></script>
</head>
<body>
    <template id="account">
        <h3>账户组件内容：</h3>
        <register :tip="msg"></register>
    </template>
    <div id="app">
        <account></account>
    </div>
    <script>
    Vue.component('account',{
        template:'#account' ,
        data:function(){
            return {
                msg:'账户管理'
            }
        },
        components:{
            'register':{
                template:'<h2>注册组件内容 --->tip={{tip}}</h2>',
                props:['tip'] //定义接收父组件传入的tip值
            }
        }
    });
    new Vue({
        el:'#app'
    });
    </script>
</body>
</html>
```

将msg值传入到子组件register中的tip参数中

二者必须保持同名

2.0.6 实现子组件传值给父组件

```
    <script src="../vue1028.js"></script>
</head>
<body>
    <template id="subcom">
        <button @click="sendData">发送数据给父组件</button>
    </template>
    <div id="app">
        <subcom v-on:send="getdata"></subcom>
    </div>
    <script>
    new Vue({
        el:'#app',
        methods:{
            getdata:function(input){
                alert(input);
            }
        },
        components:{
            'subcom':{
                template:'#subcom',
                methods:{
                    sendData:function(){
                        //利用this.$emit()将hello传入给父组件
                        this.$emit('send','hello');
                    }
                }
            }
        }

    });
    </script>
</body>
</html>
```

触发方法

二者保持同名

2.0.7 通过v-el获取到dom对象

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Document</title>
    <script src="../vue1028.js"></script>
</head>
<body>
    <div id="app">
        <button @click="getdata">获取dom对象</button>
        <div v-el:mydiv>这是一个div</div>
    </div>
    <script>

    new Vue({
        el:'#app',
        methods:{
            getdata:function(){
                //在这个方法中通过vue形式获取到div的对象
                console.log(this.$els.mydiv);
                this.$els.mydiv.innerText="修改以后的值";
            }
        }
    });
    </script>
</body>
</html>
```

二者保持一致才能获取到dom对象

2.0.8 通过v-ref获取到整个组件的对象

```
    <script src="../vue1028.js"></script>
</head>
<body>
    <div id="app">
        <button @click="getdata">获取dom对象</button>
        <subcom v-ref:mycomp></subcom>
    </div>
    <script>

    //1.0 定义组件
    Vue.component('subcom',{
        template:'<h1>这是subcom子组件内容</h1>',
        data:function(){
            return {
                msg:'hello'
            }
        }
    });

    new Vue({
        el:'#app',
        methods:{
            getdata:function(){
                // 获取到subcom这个组件对象
            console.log(this.$refs.mycomp);
            console.log(this.$refs.mycomp.$data.msg);
            }
        }
    });
    </script>
```

这是一个组件

二者保持同名才能获取到组件对象