

第九章 render函数

9.1 render函数初步了解

template下只允许有一个子节点

```
<template id="hdom">
  <div>
    <h1 v-if="level==1">
      <slot></slot>
    </h1>
    <h2 v-if="level==2">
      <slot></slot>
    </h2>
    <h3 v-if="level==3">
      <slot></slot>
    </h3>
  </div>
</template>
```

```
//是用vue组件定义
// Vue.component('child',{
//   props:['level'],
//   template:'#hdom'
// })
//使用render函数进行定义组件
Vue.component('child',{
  render:function (createElement) {
    return createElement('h'+this.level,
      this.$slots.default);
  },
  props:['level']
})
```

9.2 render函数的第一个参数

在render函数的方法中，参数必须是createElement,createElement的类型是ifunction
render函数的第一个参数可以是 String | Object | Function

```

Vue.component('child',{
// ----第一个参数必选
//String--html标签
//Object---一个含有数据选项的对象
//Function---方法返回含有数据选项的对象
render: function (createElement) {
    alert(typeof createElement)
    // return createElement('h1')
    // return createElement({
    //     template:'<div>锄禾日当午</div>'
    // })
    var domFun = function () {
        return {
            template:'<div>锄禾日当午</div>'
        }
    }
    return createElement(domFun());
}
});

```

9.3 render函数的第二个参数

```

Vue.component('child',{
// ----第二个参数可选,第二个参数是数据对象----只能是Object
render: function (createElement) {
    return createElement({
        template:'<div>我是龙的传人</div>'
    },{
        'class':{
            foo: true,
            baz:false
        },
        style:{
            color:'red',
            fontSize: '16px'
        },
        //正常的html特性
        attrs:{
            id: 'foo',
            src: 'http://baidu.com'
        },
        //用来写原生的Dom属性
        domProps:{
            innerHTML:'<span style="color:blue;font-size: 1

```

```

      8px">我是蓝色</span>'
    }
  })
}
});

```

9.3 render函数的第三个参数

第三个参数也是可选===String | Array—作为我们构建函数的子节点来使用的

```

Vue.component('child',{
  // ----第三个参数是可选的，可以是 String | Array---代表子节点
  render: function (createElement) {
    return createElement('div',[
      createElement('h1','我是h1标题'),
      createElement('h6','我是h6标题')
    ])
  }
});

```

9.4 this.\$slots在render函数中的应用

第三个 参数存的就是VNODE

createElement('header',header), 返回的就是VNODE

var header = this.\$slots.header; //--这返回的内容就是含有=VNODE的数组

```

Vue.component('my-component',{
  render:function (createElement) {
    debugger
    var header = this.$slots.header; //--这返回的内容就是含有=V
    NODE的数组
    var main = this.$slots.default;
    var footer = this.$slots.footer;
    return createElement('div',[
      createElement('header',header),
      createElement('main',main),
      createElement('footer',footer)
    ]);
  }
});

```

9.5 在render函数中使用props传递数据

```
<div id="app" >
  <button @click="switchShow">点击切换美女</button>   {{show}}
  <my-component :show="show">

  </my-component>

</div>
// 需求: 点击按钮切换美女图片
Vue.component('my-component',{
  props:['show'],
  render:function (createElement) {
    var imgsrc;
    if(this.show){
      imgsrc = 'img/001.jpg'
    }else{
      imgsrc = 'img/002.jpg'
    }
    return createElement('img',{
      attrs:{
        src: imgsrc
      },
      style:{
        width:'600px',
        height:'400px'
      }
    });
  }
});
```

9.6 v-model在render函数中的使用

```
<!--<my-component :name="name" @input="showName"></my-component>-->
<my-component :name="name" v-model="name"></my-component>
<br> {{name}}
// 需求:
Vue.component('my-component',{
  render:function (createElement) {
    var self = this; // 指的就是当前的VUE实例
    return createElement('input',{
      domProps:{
```

```

        value: self.name
      },
      on: {
        input: function (event) {
          debugger
          var a = this;
          //此处的this指的是什么？指的就是window
          self.$emit('input', event.target.value)
        }
      }
    })
  },
  props: ['name']
})

```

9.6 作用域插槽在render函数中的使用

```

Vue.component('my-component',{
  render: function (createElement) {
    return createElement('div', this.$scopedSlots.default({
      text: '我是子组件传递过来的数据',
      msg: 'scopetext'
    }))
  }
})

```

9.7 函数化组件的应用

使用context的转变——

```

// this.text----context.props.text
//this.$slots.default-----context.children

```

functional: true,表示该组件无状态无实例