第二章：组件入门及相关

一： 什么是组件 以及有什么优势

> a) 组件是可以扩展 HTML 元素，封装可重用的代码
> b) 在较高层面上，组件是自定义元素 （例如在我们的H5 项目中的 UI-XX）
> c) 优势一：组件（Component）是 Vue.js 最强大的功能之一
> d) 优势二：组件（Component）在很多的 大型项目中常常用到，便于代码管理以及维护和 代码复用
> e) 优势三：在 Vue.js 中指令和组件分得更清晰。指令只封装 DOM 操作，而组件代表一个自给自足的独立单元 —— 有自己的视图和数据逻辑。在 Angular 中两者有不少相混的地方。（来啊，互相伤害啊！）

二： 如何编写组件（可扩展性 和 移植性）

> 第一步： 创建一个组件构造器。（用 Vue.extend() 创建一个组件构造器）

```
var MyComponent = Vue.extend({
    // 选项...
})
```

> 第二步： 注册 （用 Vue.component(tag, constructor) 注册 ）

```
// 全局注册组件，tag 为 my-component
Vue.component('my-component', MyComponent)
```

> 第三步：使用（父实例的模块中以自定义元素 的形式使用）

```
    <div id="example">
      <my-component></my-component>
    </div>
```

```
// 定义
var MyComponent = Vue.extend({
  template: '<div>A custom component!</div>'
})

// 注册
Vue.component('my-component', MyComponent)

// 创建根实例
new Vue({
  el: '#example'
})
```

> **以上这种方式，全局注册组件，而且我们也不需要全局注册每一个组件。 下面介绍 文档中给介绍的 也是我们最常用到的 在实例中添加组件**

```
var Child = {
    template: '...',
};

/* 方式一：在实例中使用组件 */
new Vue({
    el: 'body',
    data: {
        idData: "this is data"
    },
    components: {
        'ui-child':  Child
    }
})

/* 方式二：在组件中使用组件 */

var Parent = Vue.extend({
    template: '...',
    components: {
    // <my-component> 只能用在父组件模板内
    'ui-child': Child
  }
})
```

以上两种方式的 用法

```
方式一：
<html>
    <head></head>
    <body>
        <ui-child></ui-child>
    </body>
</html>
```

```
方式二：
var Parent = Vue.extend({
    template: '<ui-child></ui-child>'
})
```

三： vueJS 组件内部属性

```
一、 data (**)
```

```
var Parent = Vue.extend({
    data: function() {
        return {
            isDataA: 'this is components dataA',
            isDataB: 'this is components dataB'
        }
    }
});
```

```
二、 el (**)
```

```
var Parent = Vue.extend({
    el: function() {
        return { '#isParent' }
    }
});
```

> 为什么 1、2 这两个 比较特殊的 给了（ ** ），解释下

```
var data = { a: 1 }
var MyComponent = Vue.extend({
  data: data
})
```

> **这么做的问题是 MyComponent 所有的实例将共享同一个 data 对象!!!!!!（这并不是我们想要的）**
> 所以 我们应当 **使用一个函数作为 *data* 选项**，让这个函数返回一个新对象；（ el 同理 ）
> 除了 data 和 el 属性， 剩下的基本上和 vue 的实例 属性 一样。

```
三、 template
```

```
var MyComponent = Vue.extend({
    template: '<div>123</div>',
    data: function() {
        return {
            isdata: 'this is dataA'
        }
    }
})

// 这里有几点需要注意的地方  (详情见文档)
/*
* a 不能包含其它的交互元素（如按钮，链接）
* ul 和 ol 只能直接包含 li
* select 只能包含 option 和 optgroup
* table 只能直接包含 thead, tbody, tfoot, tr, caption, col, colgroup
* tr 只能直接包含 th 和 td
*/

/*
*自定义标签（包括自定义元素和特殊标签，如 <component>、<template>、 <partial> ）
*不能用在 ul, select, table 等对内部元素有限制的标签内。
*放在这些元素内部的自定义标签将被提到元素的外面，因而渲染不正确
*/
```

> 对于自定义元素，应当使用 is 特性

```
<table>
  <tr is="my-component"></tr>
</table>
```

```
/*
* <template> 不能用在 <table> 内，这时应使用 <tbody>，<table> 可以有多个 <tbody>
*/

<table>
  <tbody v-for="item in items">
    <tr>Even row</tr>
    <tr>Odd row</tr>
  </tbody>
</table>
```

```
    四、 props
```

```
    var MyComponent = Vue.extend({
        template: '<div>{{isDataB}}</div>',
        props: {
            isDataB:{
                default: 'this is default dataB'
            }
        },
        data: function() {
            return {
                isdata: 'this is dataA'
            }
        }
    })
```

```
    <div>
        <my-component isDataB="this is dataB"></my-component>
    </div>
```

```
    <!--render 效果-->
    <div>
        <div>this is dataB</div>
    </div>
```

五、 methods

```
var MyComponent = Vue.extend({
    template: '<div>{{isDataB}}</div>',
    props: {
        isDataB:{
            default: 'this is default dataB'
        }
    },
    data: function() {
        return {
            isdata: 'this is dataA'
        }
    },
    methods: {
        todo: function() {
            // todo something
        },
        todoElse: function(){
            // to do something else
        }
    }
})
```

六、 ready

```
var MyComponent = Vue.extend({
    template: '<div>{{isDataB}}</div>',
    props: {
        isDataB:{
            default: 'this is default dataB'
        }
    },
    data: function() {
        return {
            isdata: 'this is dataA'
        }
    },
    methods: {
        todo: function() {
            // todo something
        },
        todoElse: function(){
            // to do something else
        }
    },
    ready: function() {
        // todo something
    }
})
```

> ready 这状态有什么用，做什么的 详情见 生命周期图

```
七、 等等
```

四： vueJS 兄弟 组件 之间如何通信

五： vueJS 父子 组件 之间如何通信

六： vueJS 组件 的生命周期