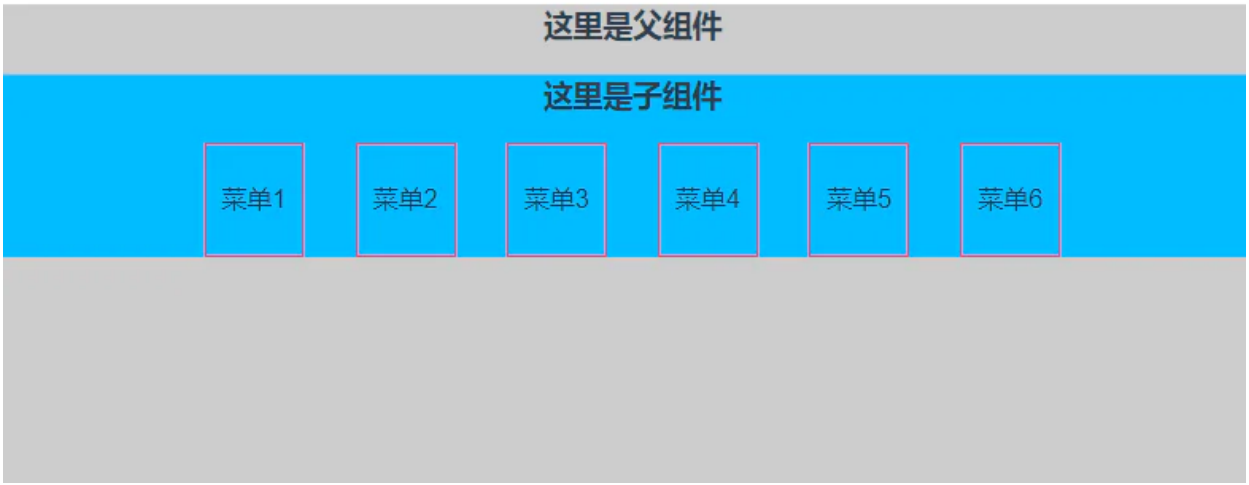


写在前面

vue中关于插槽的文档说明很短，语言又写的很凝练，再加上其和methods，data，computed等常用选项在使用频率、使用先后上的差别，这就有可能造成初次接触插槽的开发者容易产生“算了吧，回头再学，反正已经可以写基础组件了”的想法，于是就关闭了vue的说明文档。

实际上，插槽的概念很简单，下面通过分三部分来讲。这三部分也是按照vue说明文档的顺序来写的。

进入这三部分之前，先让还没接触过插槽的同学对什么是插槽有一个简单的概念：**插槽，也就是slot，是组件的一块HTML模板，这块模板显示不显示、以及怎样显示由父组件来决定。** 实际上，一个slot最核心的两个问题在这里就点出来了，是**显示不显示**和**怎样显示。**

由于插槽是一块模板，所以，**对于任何一个组件**，从**模板种类的角度来分**，其实都可以分为**非插槽模板**和**插槽模板**两大类。 非插槽模板指的是**html模板**，比如‘div、span、ul、table’这些，**非插槽模板的显示与隐藏以及怎样显示由组件自身控制**；插槽模板是slot，它是一个空壳子，**因为它的显示与隐藏以及最后用什么样的html模板显示由父组件控制（即是他的显示隐藏不是由自己控制的，而是父组件控制的）。但是插槽显示的位置却由子组件自身决定，slot写在组件template的什么位置，父组件传过来的模板将来就显示在什么位置（slot的位置，其实就是写在子组件什么位置就在子组件那个位置显示）。**

单个插槽 | 默认插槽 | 匿名插槽

首先是单个插槽，**单个插槽**是vue的官方叫法，但是其实也可以叫它默认插槽，或者与具名插槽相对，我们可以叫它匿名插槽。因为它不用设置name属性。

单个插槽可以放置在组件的任意位置，但是就像它的名字一样，一个组件中只能有一个该类插槽。相对应的，**具名插槽就可以有很多个，只要名字（name属性）不同就可以了。**

下面通过一个例子来展示。

父组件：

```
<template> <div class="father"> <h3>这里是父组件</h3> <child> <div class="tmpl"> <span>菜单1</span> <span>菜单2</span> <span>菜单3</span> <span>菜单4</span> <span>菜单5</span> <span>菜单6</span> </div> </child> </div> </template> 复制代码
```

子组件：

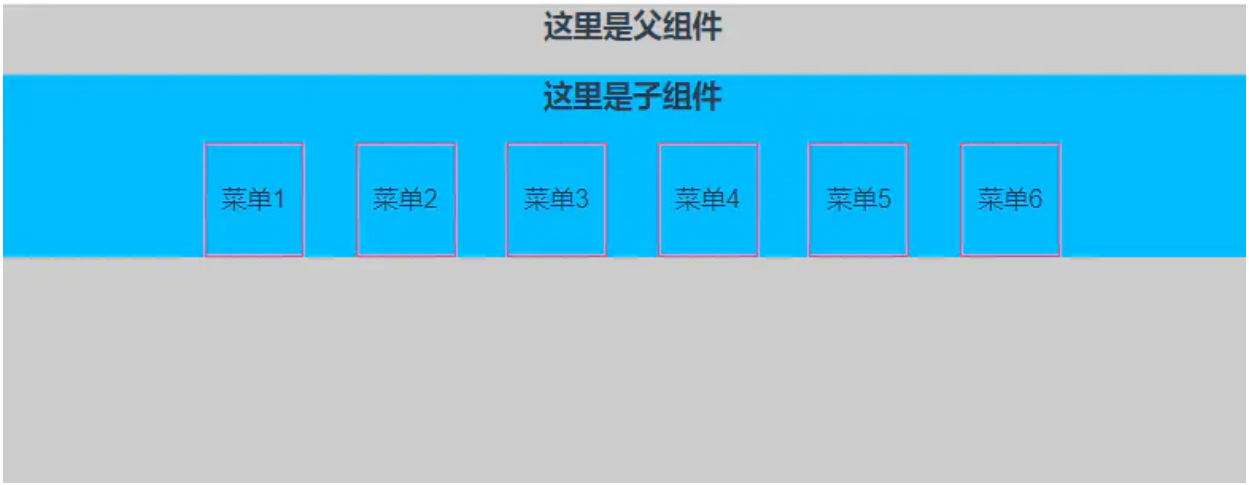
```
<template> <div class="child"> <h3>这里是子组件</h3> <slot></slot> </div> </template> 复制代码
```

在这个例子里，因为父组件在里面写了html模板，那么子组件的**匿名插槽**这块模板就是下面这样。也就是说，子组件的匿名插槽被使用了，是被下面这块模板使用了**（其实这个地方的本质是单个插槽默认就是用的**

父组件里面写在字组件里面的内容，所以这也解释了为什么单个插槽只能有一个的愿意，因为他就是用的父组件里面的东西）。

```
<div class="tmpl"> <span>菜单1</span> <span>菜单2</span> <span>菜单3</span> <span>菜单4</span> <span>菜单5</span> <span>菜单6</span> </div>
```
复制代码

最终的渲染结果如图所示：



注：所有demo都加了样式，以方便观察。其中，父组件以灰色背景填充，子组件都以浅蓝色填充。 复制代码

具名插槽

匿名插槽没有name属性，所以是匿名插槽，那么，插槽加了name属性，就变成了具名插槽。具名插槽可以在一个组件中出现N次，出现在不同的位置。下面的例子，就是一个有两个**具名插槽**和**单个插槽**的组件，这三个插槽被父组件用同一套css样式显示了出来，不同的是内容上略有区别。

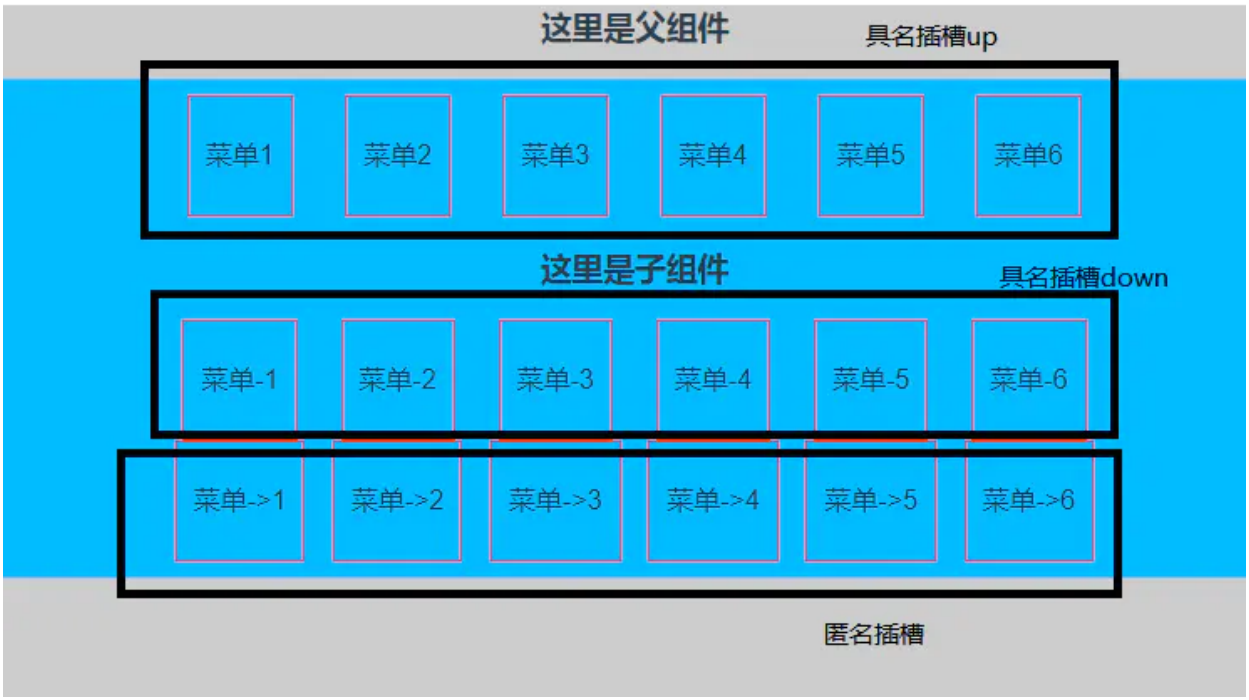
父组件：

```
<template> <div class="father"> <h3>这里是父组件</h3> <child> <div class="tmpl" slot="up"> <span>菜单1</span> <span>菜单2</span> <span>菜单3</span> <span>菜单4</span> <span>菜单5</span> <span>菜单6</span> </div> <div class="tmpl" slot="down"> <span>菜单-1</span> <span>菜单-2</span> <span>菜单-3</span> <span>菜单-4</span> <span>菜单-5</span> <span>菜单-6</span> </div> <div class="tmpl"> <span>菜单->1</span> <span>菜单->2</span> <span>菜单->3</span> <span>菜单->4</span> <span>菜单->5</span> <span>菜单->6</span> </div> </child> </div> </template>
```
复制代码

子组件：

```
<template> <div class="child"> // 具名插槽 <slot name="up"></slot> <h3>这里是子组件</h3> // 具名插槽 <slot name="down"></slot> // 匿名插槽 <slot></slot> </div> </template>
```
复制代码

显示结果如图：



可以看到，父组件通过html模板上的slot属性关联具名插槽。没有slot属性的html模板默认关联匿名插槽。

作用域插槽 | 带数据的插槽

最后，就是我们的作用域插槽。这个稍微难理解一点。官方叫它作用域插槽，实际上，对比前面两种插槽，我们可以叫它带数据的插槽。什么意思呢，就是前面两种，都是在组件的template里面写

```
匿名插槽 <slot></slot> 具名插槽 <slot name="up"></slot> 复制代码
```

但是**作用域插槽要求，在slot上面绑定数据。**也就是你得写成大概下面这个样子。

```
<slot name="up" :data="data"></slot> export default { data: function(){ return { data: ['zhangsan','lisi','wanwu','zhaoliu','tianqi','xiaoba'] } }, } 复制代码
```

我们前面说了，插槽最后显示不显示是看父组件有没有在child下面写模板，像下面那样。

```
<child> html模板 </child> 复制代码
```

写了，插槽就总得在浏览器上显示点东西，东西就是html该有的模样，没写，插槽就是空壳子，啥都没有。

OK，**我们说有html模板的情况，就是父组件会往子组件插模板的情况**，那到底插一套什么样的样式呢，这由父组件的html+css共同决定，但是这套样式里面的内容呢?

正因为作用域插槽绑定了一套数据，父组件可以拿来用。于是，情况就变成了这样：**样式父组件说了算，但内容可以显示子组件插槽绑定的。**

我们再来对比，作用域插槽跟单个插槽和具名插槽的区别，**因为单个插槽和具名插槽不绑定数据，所以父组件提供的模板一般要既包括样式又包括内容**，上面的例子中，你看到的文字，“菜单1”，“菜单2”都是父组件自己提供的内容；而**作用域插槽，父组件只需要提供一套样式（在确实用作用域插槽绑定的数据的前提下）。**

下面的例子，你就能看到，父组件提供了三种样式(分别是flex、ul、直接显示)，都没有提供数据，数据使用的都是子组件插槽自己绑定的那个数组（一堆人名的那个数组）。

父组件：

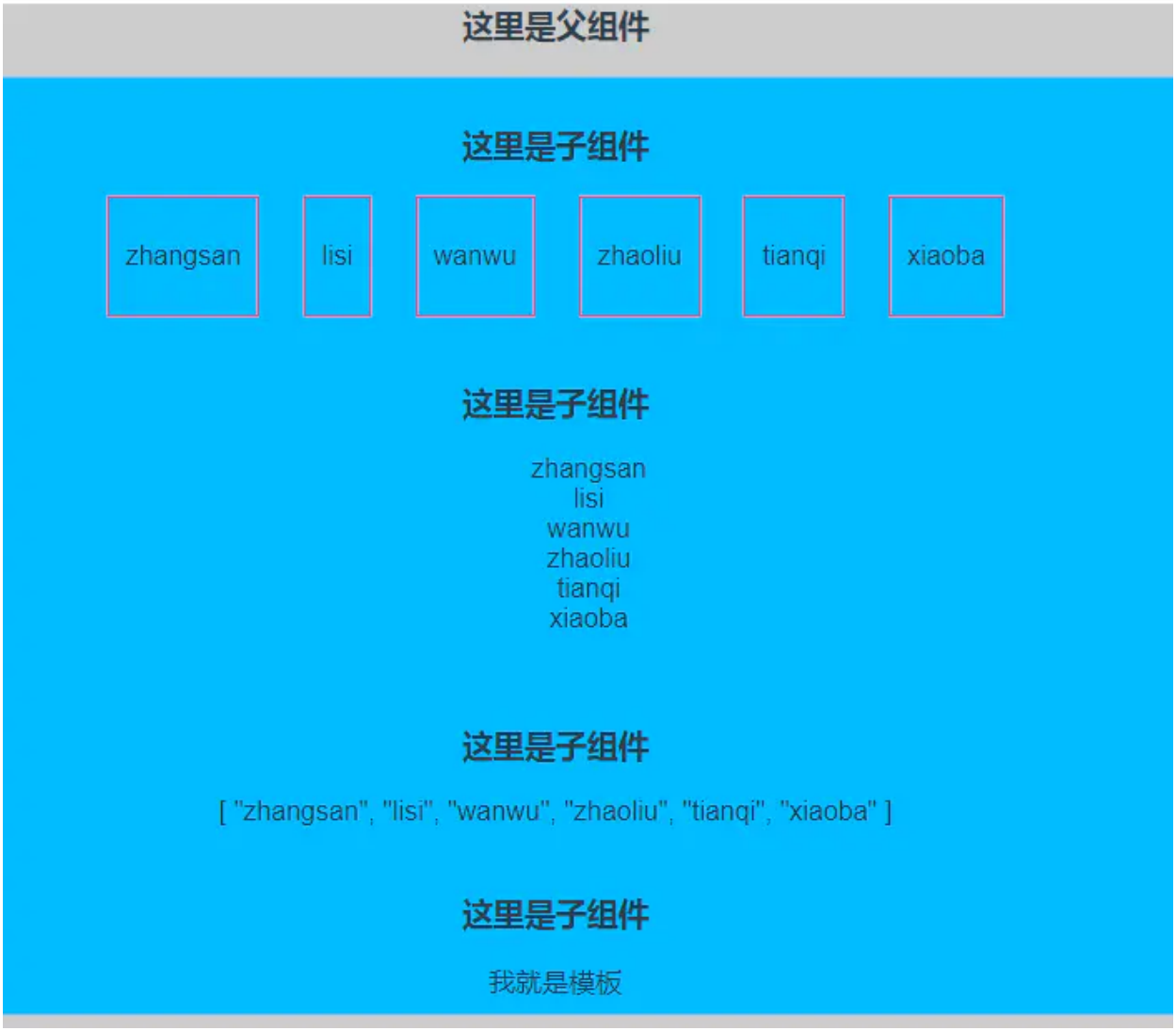
```
<template> <div class="father"> <h3>这里是父组件</h3> <!--第一次使用：用flex展示数据--> <child> <template slot-scope="user"> <div class="tmpl"> <span v-for="item in user.data">{{item}}</span> </div> </template> </child> <!--第二次使用：用列表展示数据--> <child> <template slot-scope="user"> <ul> <li v-for="item in user.data">{{item}}</li> </ul> </template> </child> <!--第三次使用：直接显示数据-->
```

```
<child> <template slot-scope="user"> {{user.data}} </template> </child> <!--第四次使用：不使用其提供的数据，作用域插槽退变成匿名插槽--> <child> 我就是模板 </child> </div> </template>
```
复制代码

子组件：

```
<template> <div class="child"> <h3>这里是子组件</h3> // 作用域插槽 <slot :data="data"></slot> </div> </template> export default { data: function(){ return { data: ['zhangsan','lisi','wanwu','zhaoliu','tianqi','xiaoba'] } } }
```
复制代码

结果如图所示：



github