vue中的 ref 和 $refs

vm.$refs

- **类型**： `Object`
- **只读**
- **详细**：

  一个对象，持有已注册过 `ref` 的所有子组件。

ref

- **预期**： `string`

  `ref` 被用来给元素或子组件注册引用信息。引用信息将会注册在父组件的 `$refs` 对象上。如果在普通的 DOM 元素上使用，引用指向的就是 DOM 元素；如果用在子组件上，引用就指向组件实例：

  ```HTML
  <!-- vm.$refs.p will be the DOM node -->
  <p ref="p">hello</p>

  <!-- vm.$refs.child will be the child comp instance -->
  <child-comp ref="child"></child-comp>
  ```

  当 `v-for` 用于元素或组件的时候，引用信息将是包含 DOM 节点或组件实例的数组。

  关于 ref 注册时间的重要说明：因为 ref 本身是作为渲染结果被创建的，在初始渲染的时候你不能访问它们 - 它们还不存在！ `$refs` 也不是响应式的，因此你不应该试图用它在模板中做数据绑定。

- **参考**：子组件 Refs
- **源代码**

```
<div id="app">
    <input type="text" ref="input1" id='input1' />
    <button @click="add">添加</button>
</div>
<script src="../js/vue.js"></script>
<script type="text/javascript">
var Vuepro = new Vue({
                el:'#app',
                methods:{
                    add:function(){
                        this.$refs.input1.value ="22"; //this.$refs.input1 减少获取dom节点的消耗
                        console.log(this.$refs.input1)//<input type="text" id="input1">
                        console.log(document.getElementById('input1'))//<input type="text" id="input1">
                        }
                    }
            })
</script>
```

如图，`ref` 被用来给元素或子组件注册引用信息。引用信息将会注册在父组件的 `$refs` 对象上。如果在普通的 DOM 元素上使用，引用指向的就是 DOM 元素；如果用在子组件上，引用就指向组件实例：

在上面的例子中，input的引用信息为input1 ， $refs 是所有注册过的ref的一个集合，

`console.log(this.$refs.input1)//<input type="text" id="input1">`

`console.log(document.getElementById('input1'))//<input type="text" id="input1">`

这两种方法获得的都是Dom节点，而$refs相对document.getElementById的方法，会减少获取dom节点的消耗。

子组件引用

尽管有 prop 和事件，但是有时仍然需要在 JavaScript 中直接访问子组件。为此可以使用 `ref` 为子组件指定一个引用 ID。例如：

```HTML
<div id="parent">
  <user-profile ref="profile"></user-profile>
</div>
```

```JS
var parent = new Vue({ el: '#parent' })
// 访问子组件实例
var child = parent.$refs.profile
```

当 `ref` 和 `v-for` 一起使用时，获取到的引用会是一个数组，包含和循环数据源对应的子组件。

> `$refs` 只在组件渲染完成后才填充，并且它是非响应式的。它仅仅是一个直接操作子组件的应急方案——应当避免在模板或计算属性中使用 `$refs`。

```
<body>
    <div id="app">
        <input type="text" ref="input1" id='input1' />
        <button @click="add">添加</button>
        <my-component ref="com"></my-component>
    </div>
    <script src="../js/vue.js"></script>
    <script type="text/javascript">
    Vue.component('my-component',{
            template:"<div>我是一个全局组件</div>"
                    }
                )
    var vuepro = new Vue({
                    el:'#app',
                    methods:{
                        add:function(){
                            this.$refs.input1.value ="22"; //this.$refs.input1 减少获取dom节点的消耗
                            console.log(this.$refs)
                            console.log(this.$refs.input1)//<input type="text" id="input1">

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

组件

```
▼ Object {input1: input#input1, com: VueComponent}
  ▶ com: VueComponent
  ▶ input1: input#input1
  ▶ __proto__: Object
```

ref和v-for在一起的情况

```
<div id="app">
    <input type="text" ref="input1" id='input1' />
    <button @click="add">添加</button>
    <ul v-for="item in list">
        <li ref="item">姓名：{{item.name}}；年龄：{{item.age}}</li>
    </ul>
    <my-component ref="com"></my-component>
</div>
<script src="../js/vue.js"></script>
<script type="text/javascript">
Vue.component('my-component',{
        template:"<div class='inner'>我是一个全局组件</div>"
                }
            )
var Vuepro = new Vue({
                el:'#app',
                data:{
                    list:[
                    {name:'lily',age:14},
                    {name:'lucy',age:24},
                    {name:'lulu',age:32},
                    ]
                },
                mounted:function(){
                    console.log(this.$refs)
                },
```

li里的ref的无法读取item里面的值，即item.name或被直接读取为字符串“item.name”，

此时的$refs

整个用下来就是比较方便取DOM，方便操作DOM