Vue基础语法

1.挂载点，模板和实例之间的关系

> 挂载点：Vue实例中 **el** 属性所对应的节点

> 模板：挂载点内部的内容就是模板，也可以写在Vue实例的 **template** 属性里面

> 实例：js脚本中 **new Vue** 创建出来的为一个vue实例，其中包含挂载点，模板，数据，方法等一系列元素

```
<div id="root">
    <h1>{{content}}</h1>
</div>

<script>
    new Vue({
        el: "#root",
        // template: '<h1>Hello Vue</h1>',
        data: {
            content:"<h1>hello vue</h1>"
        }
    });
</script>
```

2.数据和事件

> 对一个**DOM**赋值可以使用一下三种方式

- **{{变量名}}** 插值表达式
- **v-text** : 原样输出 变量里面的内容， 和插值表达式一样
- **v-html** : 会尝试将变量内容解析为html元素, 与使用template相似

```
<div id="root">
    <!-- 引入数据的三种方式 -->
    <h1>{{number}}</h1>
    <h1 v-text="number"></h1>
    <h1 v-html="number"></h1>
</div>

<script>
    new Vue({
        el: "#root",
        data: {
            number: <h1>hello vue</h1>
        }
```

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

> **v-on** 和 **@** 都可以绑定事件，对应时间处理写在 **method** 中

```
    <div id="root">
        <div v-on:click="handleClick">{{content2}}</div>
        <div @click="handleClick2">{{content2}}</div>
    </div>
    <script>
        new Vue({
            el: "#root",
            data: {
                content2:"hello"
            },
            methods: {
                handleClick: function(){
                    this.content2 = "world";
                },
                handleClick2: function(){
                    this.content2 = "hello";
                }
            }
        });
    </script>
```

3.属性绑定和双向数据绑定

> **v-bind** 可以实现属性和数据间的单向绑定，即数据可以单向传递给属性,可以简写为冒号加属性 **:title**

> **v-model** 可以实现value属性和数据间的双向绑定，任一方的值改变都可以传给另一方

```
    <div id="root">
        <!-- 直接对应title，无法获取data中的DOMtitle值 -->
        <div title="DOMtitle">hello world</div>
        <!-- 使用 v-bind 标签可以将属性和data值绑定，DOMtitle 会对应到 data.DOMtitle
-->
        <div v-bind:title="DOMtitle">hello vue</div>
        <!-- 使用 v-bind 标签可以简写为冒号 : 代替  -->
        <div :title="DOMtitle">hello vue</div>

        <input v-model="content" />
        <div>{{content}}</div>
    </div>

    <script>
        new Vue({
            el: "#root",
```

```
            data: {
                DOMtitle: "this is hello vue",
                content: "this is content"
            }
        });
    </script>
```

4.计算属性与侦听器

> **计算属性**：一个属性的值是根据其他数据项计算得来的结果，写在 **computed** 中，使用函数来进行计算赋值

> **侦听器**：监测某一个数据或者计算属性的变化，一旦发生变化，就可以触发侦听器中对应的业务逻辑，写在 **watch** 中

```
    <div id="root">
        姓：<input v-model="firstName" />
        名：<input v-model="lastName" />
        <div>{{fullName}}</div>
        <div>{{count}}</div>
    </div>
    <script>
        new Vue({
            el: "#root",
            data: {
                firstName:"",
                lastName:"",
                // fullName:firstName+" "+lastName
                count: 0
            },
            //一个属性通过其他属性计算而来，如果firstName 和 lastName 都没改变，则会使用上一次计算的值
            computed: {
                fullName: function(){
                    return this.firstName + " " + this.lastName;
                }
            },
            watch: {
                // firstName: function(){
                //     this.count ++;
                // },
                // lastName: function(){
                //     this.count ++;
                // }
                fullName: function(){
                    this.count ++;
                }
            }
        });
    </script>
```

5.v-if, v-show 和 v-for 标签

> **[v-if:变量名]** :根据变量名对应的布尔值来确定当前元素是否显示，值为true时显示当前元素，为false时不显示，其底层是通过创建和销毁**DOM**节点来实现的

> **[v-show:变量名]** :与 **v-if** 使用方法相同，其底层是通过添加和移除元素的样式**"display:none"**来实现元素的展示和隐藏，对于需要频繁改变现实状态的DOM，使用 **v-show** 效率高一点

```
    <div id="root">
        <!-- 直接移除DOM节点俩实现隐藏 -->
        <div v-if="show">hello v-if</div>
        <!-- 通过给节点添加 display:none 样式来隐藏节点 -->
        <div v-show="show2">hello v-show</div>
        <button @click="handleVif">v-if</button>
        <button @click="handleVshow">v-if</button>
    </div>
    <script>
        new Vue({
            el: "#root",
            data: {
               show: true,
               show2:true
            },
            methods: {
                handleVif: function(){
                    this.show = !this.show;
                },
                handleVshow:function(){
                    this.show2 = !this.show2;
                }
            }
        });
    </script>
```

> **v-for** :对于需要遍历的数据内容，可以使用 **v-for="item of list"**的形式来遍历的数据，其中**item**为每个数据项，**list**是**data**中定义的数据集合,DOM中可以使用{{item}}来获取数据项的值

```
    <div id="root">
       <ul>
           <li v-for="item of list">{{item}}</li>
            <!-- 添加key元素可以提高遍历效率，key必须唯一，item唯一时也可以当做key -->
           <li v-for="item of list" key="item">{{item}}</li>
           <!-- 如果暂时没有合适的值来作为key，可以item后附加index作为key，index是自动生成的唯一序列 -->
           <li v-for="(item, index) of list" key="index">{{item}}</li>
       </ul>
    </div>
```

```
    <script>
        new Vue({
            el: "#root",
            data: {
               list:[1,2,3]
            }
        });
    </script>
```