Vue第一节课

学习目标：

1.理解vue的五大指令

2.理解计算属性compute

3.了解ref

一、Vue概述

Vue.js是前端三大新框架：Angular.js、React.js、Vue.js之一，Vue.js目前的使用和关注程序在三大框架中稍微胜出，并且他的热度还在递增。

Vue.js读音,类似于view

通俗的说：

Vue.js是一个构建数据驱动的Web界面的渐进式框架

Vue.js的目标是通过尽可能简单的API实现响应的数据绑定和组合的视图组件

核心是一个响应的数据绑定系统

Vue官方说明文档：https://cn.vuejs.org/guide/introduction.html

二、如何使用Vue框架?

准备：

Vue.js 文件，这个文件可以从官网下载（很有可能会不成功），我课后也会给大家发的文件。

拿到Vue.js 文件后，将它粘贴到项目中。

通过包的形式引入

```
<script src="vue.js"></script>
```

三、使用vue框架

在Vue.js中，一个Vue实例代表一个Vue应用的根节点。Vue实例是创建Vue应用的起点，它连接了Vue应用与DOM。创建一个Vue实例非常简单，你只需要使用new关键字和Vue构造函数，如下所示：

```
let vm = new Vue({
  // 选项
});
```

在创建Vue实例时，你可以传入一个选项对象。这个选项对象可以包含数据、模板、挂载元素、方法、生命周期钩子等等选项。下面是部分常见的选项：

- el：提供一个在页面上已存在的DOM元素作为Vue实例的挂载目标。它可以是一个CSS选择器，也可以是一个HTMLElement。
- data：Vue实例的数据对象。
- methods：定义属于Vue实例的自定义方法。这些方法可以直接通过Vue实例调用，也可以在指令表达式中使用。

```
<!DOCTYPE html>
<html lang="en">

<head>
  <meta charset="UTF-8">
  <meta http-equiv="X-UA-Compatible" content="IE=edge">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <!-- 引入Vue -->
  <script src="vue.js"></script>
  <title>Document</title>
</head>
<body>
  <div id="app">{{message}}</div>
</body>
<script>
    // 实例化
    let b = new Vue({
      // el:挂载点
      el: '#app',
      // data:数据源(存键值对)
      data: {
        message: 'Hello Vue!'
      }
    })
  </script>
</html>
```

四、vue五大指令 （灵魂）

1.内容渲染指令(插举法)

- {{}}

```
<div id="app">
  {{ message }}
</div>
```

```
    data: {
        message: 'Hello Vue!'
    }

    # 渲染效果
    Hello Vue!
```

2.双向绑定指令

作用：表单

- v-model

```
<div id="div2">
    <input type="text" v-model="user_Name" placeholder="请输入账户">
    <p>{{user_Name}}</p>
</div>
```

```
// js
let div2 = new Vue({
    el:'#div2',
    data:{
        user_Name:'',
    }
})
```

3.事件绑定指令

- @click 绑定

```
<div id="qiao">
    <button @click="clickNum">点击+1</button>
    <h2>功德: {{num}}</h2>
</div>
```

- methods：定义属于Vue实例的自定义方法, 专门写事件的地方。

  在 Vue 实例中，`this` 用于引用当前实例的属性和方法。在 `clickNum` 方法中，`this.num` 引用了 Vue 实例中的 `num` 数据属性，而 `this` 指向了当前的 Vue 实例 `qiao`。

```
<script>
    // 实例化
    let qiao = new Vue({
        el: '#qiao',
        data: {
            num: 0
        },
        methods: {
            clickNum() {
                this.num++
            }
        }
    })
</script>
```

4.条件渲染指令

可以让内容进行一个显示和隐藏

- `v-if`

  `v-if` **根据条件动态地插入或移除元素，元素的存在与否取决于条件的真假。当条件为真时，元素将被渲染到DOM中，当条件为假时，元素将从DOM中移除。**

  v-if='true或者false'

```
<div id="qiao">
    <button v-if="a">点击+1</button>
    <button @click="VIPs">VIPs</button>
</div>
```

```
<script>
    // 实例化
    let qiao = new Vue({
        el: '#qiao',
        data: {
            a:true
        },
        methods: {
            VIPs() {
                this.a = !this.a
            }
        }
    })
</script>
```

- `v-show`

  `v-show`用于根据条件来显示或隐藏元素。它基于CSS的`display`属性进行切换，元素仍然存在于DOM中，只是通过设置`display`属性来控制其可见性。如果条件为真，元素将显示，如果条件为假，元素将隐藏。

```
<div id="qiao">
    <button v-show="a">点击+1</button>
    <button @click="VIPs">VIPs</button>
</div>

<script>
    // 实例化
    let qiao = new Vue({
        el: '#qiao',
        data: {
            boolean:true
        },
        methods: {
            VIPs() {
                this.a = !this.a
            }
        }
    })
</script>
```

- 总结`v-show`和`v-if`的区别：
  - v-show是通过css diaplay：none来显示和隐藏
  - v-if是通过删除和添加dom元素 来显示和隐藏

> v-show 的切换开销较小，但如果元素在初始状态是隐藏的，它仍然会被渲染到 DOM 中。而 v-if 的切换开销较大，因为它涉及到条件判断和 DOM 的插入或移除操作。因此，在选择使用 v-show 还是 v-if 时，应根据具体的使用场景和性能要求进行权衡。

5.列表渲染指令

- v-for

  `v-for`是Vue.js中用于迭代数组或对象，并渲染多个元素的指令。它可以将数据源中的每个元素循环遍历，并根据每个元素生成相应的DOM元素。

```
<body>
    <div id = "div3">
        <ul>
            <!-- 用v-for对items进行遍历 -->
            <li v-for="item in item">{{item}}</li>
            <!-- 对象进行遍历 -->
            <li v-for="(value,key) in object">{{key}}:{{value}}</li>
        </ul>
    </div>

  <script>
    // 实例化
      var vm = new Vue({
            // 挂载点
            el:'#div3',
            data:{
                message:'hello',
                item:[1,2,3,4,5],
                // 创建一个对象
                object:{
                    title:'这是一个标题',
                    author:'张三',
                    time:'2024-01-01'
                }
            }
     })
  </script>
</body>
```

五、计算属性:computed

在Vue中，`computed`（计算属性）是一种特殊的属性，用于对数据进行动态计算和处理，并返回计算结果。

计算属性会根据它依赖的响应式数据自动进行缓存，只有在依赖发生变化时才会重新计算，这样可以提高性能并减少不必要的计算。

作用:

- 响应式计算数据，计算属性可以像普通属性一样在模板中使用。

语法举例：

```
<div id="div1">
    <!-- 通过函数名返回计算后的值 -->
    {{value}}
</div>
```

计算属性可以在`vue`组件的`computed`属性中定义。以下是计算属性的基本语法:

```
<script>
    // 实例化
    let app = new Vue({
        el: '#div1',
        data: {
        },
        computed: {   // 计算属性可以在Vue组件的computed属性中定义。
            value () {
                // 通过return将 计算的结果返回
                return 1024
            }
        }
    })
</script>
```

计算属性可以依赖于其他响应式数据（包括计算属性本身）或者 Vue 实例的数据。当依赖的数据发生变化时，计算属性会重新计算。

实战案例：

```
<div id="div2">
    <h1 >要求如下：
        点击购买按钮，自动计算订单件数、总价（每件商品价格为10元） </h1>
    <button @click="shopping">点击购买商品</button>
    <h3>当前订单件数:{{count}}</h3>
    <h2>当前订单总价:{{jiaGe}}</h2>
</div>
```

`Vue`代码：

```
<script>
    let div2 = new Vue({
        el: "#div2",
        data : {
            count : 0
        },
        methods:{
            shopping(){
                this.count++
            }
        },
        computed: {
            jiaGe(){
                return this.count*10
            }
        }
    })
</script>
```

六、ref 获取 dom

在`Vue`中，可以使用`ref`特殊属性来获取DOM元素的引用。`ref`可以用于在`Vue`组件中给DOM元素添加一个标识符，然后通过`this.$refs`来访问该DOM元素的引用。

```
<div id="div5" >
    <!--通过ref来获取这个dom 取名叫domref -->
    <div ref="domref">
        我是被操作的dom
    </div>
    <button @click="setdom">点击</button>
    (通过操控dom的方式 来使div变成红色)
</div>

<script>
    // 实例化
    let div5 = new Vue({
        el: "#div5",
        methods: {
            setdom() {
                // this.$refs.domref 找到dom domref 然后通过style.color 设置颜色
                this.$refs.domref.style.color = 'red'
                this.$refs.domref.style.height = '100px'
                //在JavaScript中，使用连字符（-）的CSS属性名需要使用驼峰命名法。
                this.$refs.domref.style.backgroundColor = 'pink'
            }
        }
    })
</script>
```

- ref是`vue`里的方法，更加安全，不会依赖class或者id的样式变了而影响布局
- `vue`的主要目的是减少`dom`的操作。减少`dom`节点的消耗