

# Vue2.0第一节课

2.0官网: <https://v2.cn.vuejs.org/>

## 一、什么是Vue?

官方: 是一套用于构建用户界面的**渐进式框架**

也是前端开发工程师必学的一个框架, 和Angular、React 并称为前端三大主流框架。

## 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。

### 1. 通过包的形式引入

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

### 2. 查看是否引入成功

打开网页 在控制台查看是否有vue的标识

 image-20230518164606952

## 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。Vue会递归将data对象的属性转换为getter/setter, 使得数据的改变能够触发视图的重新渲染。
- 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">
<script src="https://cdn.jsdelivr.net/npm/vue@2/dist/vue.js"></script>
<!-- 引入Vue -->
<script src="./vue.js"></script>
<title>Document</title>
</head>

<body>

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

  <script>
    // 实例化
    let app = new Vue({
      // el:挂载点
      el: '#app',
      // data:数据源(存键值对)
      data: {
        message: 'Hello vue!'
      } //提问：如何将Hello vue!渲染到div ?
    })
  </script>
</body>

</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。

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

我们需要多少个事件就可以在methods中写多少个函数 如果需要使用就使用@click进行一个绑定.

练习: 结合setInterval()定时器, 制作一个VIP小木鱼, 要求页面显示一个初始值, 且有一个用于点击的图标或者文字或者按钮, 点击之后, 页面上的数值+1。

### 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。

```
<div id="qiao">
  <button v-if="boolean">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.boolean = !this.boolean
      }
    }
  })
</script>
```

o **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="boolean">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.boolean = !this.boolean
      }
    }
  })
</script>
```

o 总结 **v-show** 和 **v-if** 的区别：

- **v-show** 是通过css `display: 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

### o 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。

- `key`: 唯一标识符 不加上不会报错 加上会提升性能，一般用 `index` 作为值。

```
<body>
  <div id = "div3">
    <li v-for="(x, index) in list" :key="x.index">
      {{x.value}}
    </li>
  </div>

  <script>
    // 实例化
    let div3 = new Vue({
      el: "#div3",
      data: {
        list: [
          {key: 1, value: 1},
          {key: 2, value: 2},
          {key: 3, value: 3},
          {key: 4, value: 4},
        ]
      }
    })
  </script>
</body>
```

拓展:

## 拓展:

### 1、`v-for` 实现列表新增和删除 【删除存在bug，请找出并修正】

`.splice(索引, 删几次)` 从数组中移除元素。

`.push(数据)` 向数组中添加新的对象

```
<div id = "div3">
  <li v-for="x in list" :key="x.key">
    {{x.value}}
    <button @click="del(key)">删除</button>
  </li>
  <p>插入新数据: </p>
```

```

    <input type="text" v-model="val" placeholder="请输入密码">
    <button @click="add">add</button>
  </div>
</script>
<script>
  let div3 = new Vue({
    el:"#div3",
    data:{
      list:[
        {key:1,value:1},
        {key:2,value:2},
        {key:3,value:3},
        {key:4,value:4},
      ],
      val : ''
    },
    methods:{
      del(id){
        this.list.splice(id,1)
      },
      add(){
        this.list.push({key:this.list.length+1,value:this.val})
        this.val = ''
      }
    }
  })
</script>

```

v-for 实现列表新增和删除代码bug纠正

```

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

<head>
  <meta charset="UTF-8">
  <title>Document</title>
  <script src="./vue.js"></script>
</head>

<body>
  <!-- `v-for` 实现列表新增和删除 -->
  <div id="div1">
    <!-- key="i.id" 我们设置渲染出来的每条数据的 key 是列表的 id值 -->
    <!-- 规定 key 应该是唯一 -->
    <!-- (i,index) in list : 其中i是遍历的数值值, index是当前值的下标-->
    <li v-for="(i,index) in list" :key="index">
      {{i.val}}
      <button @click="del(index)" >删除</button>
    </li>
    <input v-model="newVal" type="text" placeholder="请输入新增值">
    <button @click="add">新增</button>
  </div>

  <script>
    // 实例一个Vue对象

```

```

new Vue({
  // 选项
  el: "#div1",    // 挂载，将本Vue对象与上面的html进行关联
  data: {         // 数据源对象
    list: [
      { id: 1, val: "狮子" },    // 0
      { id: 2, val: "老虎" },    // 1
      { id: 3, val: "斑马" },
      { id: 4, val: "猴子" },
    ],
    newVal : ''
  },
  methods: {      // 方法区
    // .splice(下标,个数)
    del(index){
      // 本方法我们需要的数据是 下标 ，现在检测一下，下标有没有拿到
      alert(`当前数据下标是: ${index}`) // 格式化输出打印 当前值
      this.list.splice(index,1)
    },add(){
      // .push(新增的数据)
      this.list.push({id:this.list.length+1, val:this.newVal})
    }
  }
})
</script>
</body>

</html>

```

## 2、clearInterval 停止计时器

要停止使用JavaScript中的 `setInterval` 函数创建的计时器，可以使用 `clearInterval` 函数来取消该计时器。`clearInterval` 函数接受一个参数，即要取消的计时器的标识符（也称为计时器ID）。

在调用 `setInterval` 函数时，它将返回一个计时器ID。可以将该ID存储在一个变量中，并在需要停止计时器时将其传递给 `clearInterval` 函数。

以下是一个示例，展示如何使用 `setInterval` 创建一个简单的计时器，并使用 `clearInterval` 停止它：

```

// 创建计时器，并存储计时器ID
var timerId = setInterval(function() {
  console.log("计时器正在运行...");
}, 1000);

// 在一段时间后停止计时器
setTimeout(function() {
  clearInterval(timerId);
  console.log("计时器已停止。");
}, 5000);

```

在上述示例中，`setInterval` 函数每秒钟输出一条消息，持续5秒钟。然后，通过调用 `clearInterval` 函数并传递计时器ID，我们停止了计时器的执行。

请注意，确保传递给 `clearInterval` 函数的计时器ID是正确的，并且在调用 `clearInterval` 之前，计时器已经启动。

### 3、Vue实现简单登录

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Vue实现登录案例</title>
  <!-- 引入Vue -->
  <script src="./vue.js"></script>
</head>
<body>
  <div id = "login">
    <h1>登录</h1>
    <!-- 用户名输入框 -->
    <div>
      账户:
      <input type="text" v-model="userName" placeholder="请输入账户">
    </div>
    <!-- 用户密码输入框 -->
    <div>
      密码:
      <input type="password" v-model="userPass" placeholder="请输入密码">
    </div>
    <button @click="send">登录提交</button>
  </div>

  <script>
    // 创建Vue对象
    new Vue({
      el : "#login",
      data : {
        userName : '',
        userPass : ''
      },
      methods : {
        send(){
          // 验证密码是否为8位数
          if(this.userPass.length!=8){
            alert("密码必须为8位数")
          }else{
            alert("登录成功! ")
          }
          // 这里可以加登录的验证
        }
      }
    })

    // != 运算符（不相等）：它用于检查两个值是否不相等，但在比较时会进行类型转换。
    // !== 运算符（不全等）：它用于检查两个值是否不仅不相等，还不具有相同的类型。
  </script>
</body>
</html>
```



# 课后作业（基础作业和进阶作业只需要完成一个）：

---

基础作业：将课堂案例吃透

进阶作业：使用今天学的Vue知识完成上节课 登录作业和木鱼作业的需求

## 课后拓展：

---

v-for的一个小案例，有时间可以看看