Vue04

> 提前准备并启动项目包

复习

选项

- DOM
  - el: 初始化vue对象时，设定vue对象管理的元素，值 id选择器
  - render: 脚手架中使用， main.js文件. 用于加载 .vue 文件到 vue对象里
- 数据
  - data
  - computed
  - methods

指令: vue提供的一些标签的属性

- v-text: innerText 覆盖标签原内容，不解析直接显示
- v-html: innerHTML 覆盖内容，解析HTML再显示
- v-if: 根据条件来决定是否添加DOM元素. 适合不频繁的切换显示/隐藏. 最好是一次性显示
  - v-else
  - v-else-if
- v-for: 遍历数组 生成元素
  - 语法有3种
    - v-for="值 in/of 数组"
    - v-for="(值,序号) in/of 数组"
    - v-for="值 in/of 数字"
  - key: 给生成的元素添加唯一标识
    - 用途:当 数组有变化 时，提升元素重用的效率； 如果数组不变则没用
- v-on:事件
  - vue2版本 @事件名=""
- v-bind: 属性
  - vue2版本 :属性名=""
- v-model: 双向数据绑定
  - 专为 表单元素而生 -- 下拉框，单选，多选，输入框... 和用户之间能交互
  - 作用: 实时变化绑定的数据
- v-pre
  - 原样显示 {{}}
- v-once: 一次性，初次渲染到页面后，不再变化

作业

```
<template>
  <div>
    <div class="box">
      <span
        v-for="(spec, i) in specs"
        :key="i"
        :class="{ active: now == i }"
        @click="now = i"
        >{{ spec }}</span
      >
```

```
    </div>
  </div>
</template>

<script>
export default {
  data() {
    return {
      now: 0,
      specs: [
        '[流光银] i5-7200u 4G 128 500G',
        '[溢彩金] i5-7200u 8G 128 1T',
        '[元気粉] i5-7200u 4G 128 500G',
      ],
    }
  },
}
</script>

<style lang="scss" scoped>
.box {
  width: 800px;

  span {
    display: inline-block;
    padding: 10px 20px;
    margin: 5px;
    border: 2px solid #555;
    color: #555;
    user-select: none;

    &.active {
      border-color: aquamarine;
      color: aquamarine;
    }
  }
}
</style>
```

下拉

```
<template>
  <div>
    <!-- v-model: 双向绑定 -->
    <!-- 使用范围: 表单元素 - 用户可以操作的元素，单选 多选 输入框 -->
    <!-- 作用: 变量绑定在元素上，实时收集数据，当用户操作时 值发生变化就会实时存储在变量里 --
>

    <!-- 双向的表现 -->
    <!-- 1. 数据tool的默认值是4，页面初始化就显示的老头乐，即4 -->
    <!-- 2. 用户操作表单元素后，数据也会变化 -->

    <h3>壮壮，选择你心仪的座驾: {{ tool }}</h3>
    <!--  -->
    <select v-model="tool">
      <option value="0">雅迪</option>
      <option value="1">小牛</option>
      <option value="2">捷安特</option>
      <option value="3">三蹦子</option>
```

```
      <option value="4">老头乐</option>
    </select>
  </div>
</template>

<script>
export default {
  data() {
    return {
      tool: 4, //默认选项--4
    }
  },
}
</script>

<style lang="scss" scoped></style>
```

单选

```
<template>
  <div>
    <!-- 单选框 -->
    <h2>壮壮的性别是? {{ xb }}</h2>
    <!-- 数组[下标] : 服务器的数据库存选项都是序号，就是方便搭配搭配数组用 -->
    <p>{{ ['boy', 'girl', '???'][xb] }}</p>
    <label>
      <!-- 数据库中，通常用数字代表选项 -->
      <input type="radio" name="sex" value="0" v-model="xb" />
      <span>男</span>
    </label>
    <label>
      <input type="radio" name="sex" value="1" v-model="xb" />
      <span>女</span>
    </label>
    <label>
      <input type="radio" name="sex" value="2" v-model="xb" />
      <span>未知</span>
    </label>
  </div>
</template>

<script>
export default {
  data() {
    return {
      xb: 2,
    }
  },
}
</script>

<style lang="scss" scoped></style>
```

自定义

```
```

```vue
<template>
  <div>
    <!-- 系统指令：vue提供的 能够直接使用的指令 -->
    <!-- 自定义指令：指令都是 v- 开头 -->
    <ul>
      <li v-green>壮壮</li>
      <!-- 指令中的值，是JS代码，所以字符串要写引号包围 -->
      <li v-color="'orange'">浩浩</li>
      <li v-color="'pink'">边边</li>
      <!-- v- 开头，  v-green 希望让元素变绿 -->

      <!-- 仿写系统的v-show，  true显示 false不显示 display:none -->
      <li v-green v-showx="true">陈胜</li>
      <li v-blue v-showx="false">小海</li>
    </ul>
  </div>
</template>

<script>
export default {
  // 指令存储在此处：
  directives: {
    showx(a, b) {
      //写法1
      a.style.display = b.value ? '' : 'none'
      //写法2
      if (b.value) {
        a.style.display = ''
      } else {
        a.style.display = 'none'
      }
    },

    color(el, bindings) {
      // 参数1：代表指令所在的元素
      // 参数2：指令的值，通过=赋值的
      console.log('值:', bindings)
      el.style.color = bindings.value // value属性存储的实际值
    },
    blue(el) {
      el.style.color = 'blue'
    },
    // v-green:  v-是指令的标识，green是指令名
    green(el) {
      // 参数1：指令所在的DOM元素
      console.log(el)
      el.style.color = 'green'
    },
  },
}
</script>

<style lang="scss" scoped></style>
```

自定义指令练习

```vue
<template>
  <div>
```

```
    <div v-for="(emp, i) in emps" :key="i">
      <span>{{ emp.ename }}</span>
      <span>{{ emp.birthday }}</span>
      <!-- 自定义指令: 适合快速给DOM元素增加一些 DOM相关的操作 -->
      <span v-date="emp.birthday"></span>
    </div>
  </div>
</template>

<script>
export default {
  directives: {
    // v-date
    date(el, bindings) {
      // 如何把时间戳 转成 年/月/日
      var d = new Date(bindings.value)
      var year = d.getFullYear()
      var month = d.getMonth() + 1 // 0-11  +1 转1-12
      var day = d.getDate()

      el.innerText = `${year}/${month}/${day}`
    },
  },
  data() {
    return {
      emps: [
        { ename: '壮壮', birthday: 626014629000 },
        { ename: '边边', birthday: 636025629000 },
        { ename: '小海', birthday: 616021629000 },
        { ename: '浩浩', birthday: 606022429000 },
      ],
    }
  },
}
</script>

<style lang="scss" scoped></style>
```

ref

```
<template>
  <div>
    <!-- 简单粗暴: 直接把变量和元素绑在一起, 用 ref 属性 -->
    <!-- 变量会存储在 vue的 $refs 属性里 -->
    <input type="text" ref="suibian" />
    <p ref="zz">壮壮</p>
    <div ref="hh">浩浩</div>
    <br />
    <button @click="doFocus">获得焦点</button>
  </div>
</template>

<script>
export default {
  methods: {
    doFocus() {
      // 要操作输入框, 就要先找到这个元素
      // focus(): DOM元素自带的方法, 用来获得焦点
```

```
      // document.querySelector('input').focus()
      console.log(this) //this是vue对象，查看其中的 $refs 属性

      this.$refs.zz.style.color = 'green'
      this.$refs.suibian.focus()
    },
  },
}
</script>

<style lang="scss" scoped></style>
```

指令的周期

```
<template>
  <div>
    <input type="text" />
    <br />
    <!-- v-focus: 让元素显示时获得焦点 -->
    <input type="text" v-focus />
    <br />
    <input type="text" />
  </div>
</template>

<script>
export default {
  directives: {
    // 概念：生命周期. 一个事物从诞生到销毁的过程
    // 例如人的一生：肚子里->出生->成长=>死了
    // DOM元素：内存中先创建 -> 绘制到页面上 -> 销毁
    // 获得焦点：
    // 指令默认是在 DOM的内存阶段触发的，即元素还没显示到页面上，导致无法获得焦点
    // focus(el) {
    //   el.focus()
    // },
    // 指令有两种语法：
    // 1. 值是函数：则在DOM元素创建时自动触发
    // 2. 值是对象类型：精确配置触发的时机
    focus: {
      // ctrl+i :能看到提示
      // sql:的增语句  insert into 表 value (值),  插入
      // 代表DOM元素 插入到 页面上，展示出来时
      inserted(el) {
        el.focus()
      },
    },
  },
}
</script>

<style lang="scss" scoped></style>
```

axios

关于网络请求：

- 最早起：原生的 XHR
- jQuery：使用回调函数方式，对 Ajax 进行了封装

  $.get(地址，回调函数)

  回调函数使用时存在一个风险： 回调地狱--多个异步操作要同步执行，就需要在回调函数中开启下一次请求
- axios：一款第三方的专门做网络请求的模块
  - 特点：用 Promise 进行封装而来，没有回调地狱风险

> 安装：axios默认没有安装，需要在项目包中 进行安装
>
> 命令： npm i axios vue-axios

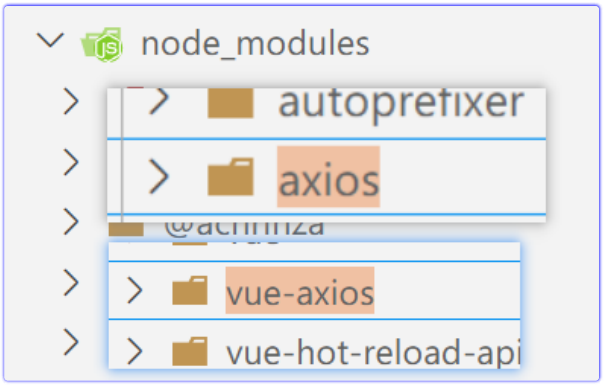
```
D:\WEB2203\14_Vue\Day04\vue-pro>npm i axios vue-axios
npm WARN config global `--global`, `--local` are deprecated. Use `--location=global` instead.

added 6 packages in 2s

D:\WEB2203\14_Vue\Day04\vue-pro>
```

> 百度网盘提供的包，是安装完毕的. 不需要再安装.

检查是否已安装的方式： 在项目包的 node_modules 目录下，能找到 axios 和 vue-axios 就行



axios

```vue
<template>
  <div>
    <button @click="suibian">获取数据</button>
    <!-- <div>data:{{ data }}</div> -->

    <!-- 必须了解：data默认值是null，对null读属性必然报错 -->
    <!-- 需要网络请求数据来展示的DOM元素，在请求未完成前，需要显示吗？ -->
    <!-- 不需要. 应该延后到请求完毕后再创建 -->
    <!-- 好处1：一旦请求未完成，不会浪费资源 -->
    <!-- 好处2：页面初始化的时候，所有性能都执行非请求操作的DOM元素，更快 -->

    <!-- v-if：如果条件是真的，才会加载DOM元素；特别适合请求场景 -->
    <!-- 单纯的提取 v-if ，用div作为容器，会影响css布局 -->
    <!-- vue专门提供了 template 标签：不参与css布局，是虚拟的容器，专门搭配 v-if 用 -->
    <template v-if="data">
      <div>fileTime: {{ data.fileTime }}</div>
      <div>fileName: {{ data.fileName }}</div>
      <div>version: {{ data.version }}</div>
```

```
    </template>
  </div>
</template>

<script>
// axios模块: 需要手动安装  npm i axios vue-axios
// 使用方式分两种:
// 1. 局部: 每个.vue文件都需要手动引入, 生效范围仅限当前文件
// 2. 全局: 每个.vue文件都可用, 引入一次 所有文件都生效

// 局部方式: 同 const axios = require('axios')
import axios from 'axios'

export default {
  data() {
    return {
      // 声明一个属性, 存储请求到的数据
      data: null, //初始没有值
    }
  },
  methods: {
    suibian() {
      // 使用范式:  axios.请求方式(地址).then(响应=>{})
      // then: 然后      res: response 响应值 - 服务器发来的
      // 接口网站: https://api.xin88.top/
      const url = 'https://api.xin88.top/game/items.json'

      axios.get(url).then(res => {
        console.log(res)
        // 请求的数据存储在 res的data属性里
        // 页面上展示的数据必须存在哪里?? data配置项中
        // 所以: 必须在data中声明一个属性, 来存储 请求而来的数据
        this.data = res.data //本地的数据 = 响应的数据
      })
    },
  },
}
</script>

<style lang="scss" scoped></style>
```

优酷

```
<template>
  <div>
    <button @click="getData">获取数据</button>
    <template v-if="data">
      <p>{{ data.footer.aboutUs.title }}</p>
      <a
        v-for="x in data.footer.aboutUs.data"
        :key="x.id"
        :href="x.url"
        target="_blank"
      >
        {{ x.name }}
      </a>
      <!--  -->
      <p>{{ data.footer.ykHot.title }}</p>
```

```
      <a
        v-for="x in data.footer.ykHot.data"
        :key="x.id"
        :href="x.url"
        target="_blank"
        >{{ x.name }}</a
      >
    </template>
  </div>
</template>

<script>
import axios from 'axios'

export default {
  data() {
    return {
      data: null,
    }
  },
  methods: {
    getData() {
      const url = 'https://api.xin88.top/youku/footer.json'

      axios.get(url).then(res => {
        console.log(res)
        this.data = res.data
      })
    },
  },
}
</script>

<style lang="scss" scoped>
p {
  font-size: 1.2em;
}
a {
  color: #444;
  display: inline-block;
  text-decoration: none;
  margin: 6px;

  &:hover {
    color: blue;
  }
}
</style>
```

番剧

```
<template>
  <div>
    <button @click="getData">获取最新番剧</button>
    <template v-if="bb">
      <a
        target="_blank"
        v-for="s in bb.data.season"
```

```
        :key="s.season_id"
        :href="s.url"
      >
        <img :src="s.cover" alt="" />
        <div>{{ s.title }}</div>
      </a>
    </template>
  </div>
</template>

<script>
import axios from 'axios'
export default {
  data() {
    return {
      bb: null,
    }
  },
  methods: {
    getData() {
      const url = 'https://api.xin88.top/bilibili/recommend.json'
      //
      axios.get(url).then(res => {
        console.log(res)
        this.bb = res.data
      })
    },
  },
}
</script>

<style lang="scss" scoped>
img {
  width: 200px;
}
</style>
```

练习

```
<template>
  <div>
    <button @click="getData">获取数据</button>
    <template v-if="data">
      <!-- for循环的东西，必须在for循环里用 -->
      <div v-for="sub in data.subjects" :key="sub.id">
        <a :href="sub.url">
          <img :src="sub.cover" alt="" />
          <span>{{ sub.title }}</span>
        </a>
      </div>
    </template>
  </div>
</template>

<script>
import axios from 'axios'
// https://api.xin88.top/douban/movies.json
export default {
```

```
  data() {
    return {
      data: null,
    }
  },
  methods: {
    getData() {
      const url = 'https://api.xin88.top/douban/movies.json'
      axios.get(url).then(res => {
        console.log(res)
        this.data = res.data
      })
    },
  },
}
</script>

<style lang="scss" scoped></style>
```

练习

```
<template>
  <div>
    <button @click="getData">获取数据</button>
    <br />
    <template v-if="data">
      <div
        class="item"
        v-for="arc in data.data.archives"
        :key="arc.aid"
      >
        <img :src="arc.pic" alt="" />
        <div>{{ arc.title }}</div>
      </div>
    </template>
  </div>
</template>

<script>
import axios from 'axios'
// https://api.xin88.top/bilibili/news.json
export default {
  data() {
    return {
      data: null,
    }
  },

  methods: {
    getData() {
      const url = 'https://api.xin88.top/bilibili/news.json'

      axios.get(url).then(res => {
        console.log(res)
        this.data = res.data
      })
```

```
    },
  },
}
</script>

<style lang="scss" scoped>
.item {
  width: 200px;
  display: inline-block;
  margin: 10px;

  img {
    width: 100%;
  }
}
</style>
```

斗鱼

```
<template>
  <div>
    <!-- 首次获取，获取第一页的 -->
    <button @click="getData(1)">获取数据</button>
    <!-- 就是希望有个块元素包裹，所以直接用div+v-if -->
    <div v-if="data">
      <button @click="getData(data.data.nowPage + 1)">
        下一页
      </button>

      <div v-for="x in data.data.list" :key="x.rid">
        <img :src="x.roomSrc" alt="" />
        <div>{{ x.roomName }}</div>
      </div>
    </div>
  </div>
</template>

<script>
import axios from 'axios'
// https://douyu.xin88.top/api/room/list?page=1&type=yz
export default {
  data() {
    return {
      data: null,
    }
  },
  methods: {
    getData(page) {
      // 关于接口中参数的作用：复习 亮亮的express
      // page：代表请求的页数
      const url = `https://douyu.xin88.top/api/room/list?page=${page}&type=yz`
      console.log('url:', url)

      axios.get(url).then(res => {
        console.log(res)
        this.data = res.data
      })
    },
  },
```

```
}
</script>

<style lang="scss" scoped></style>
```

网络请求

- 第三方的axios模块：需要自己安装 `npm i axios vue-axios`
- 引入方式分两种
  - 局部：在.vue文件中，`import axios from 'axios'`
  - 全局：没讲
- 使用时，固定语法
  - get：`axios.get(地址).then(res=>{})`
- 如何展示到页面上
  - 前提设定：只有存储在配置项data中的属性，才能在页面上显示
  - 在data中，声明一个属性
  - 在请求中，把请求结果中的数据存入本地的：`this.data = res.data`
- 使用时： 为了提高效率 和 用户体验
  - 用v-if判定:data存在再用！
  - template：虚拟容器，可以为多个元素统一进行if判断

总结

指令

- v-model：双向数据绑定，给表单元素使用，实时收集用户的数据
- 自定义指令：`v-` 开头，可以快速操作元素的DOM
  - 书写在 `directives` 配置项中
  - 用法有两种
    - 函数类型：元素创建时触发
    - 对象类型：可以指定在某个生命周期中触发
      - `inserted`：DOM元素加载到页面上 显示出来时触发
- 网络请求
  - 安装：用 `npm i axios vue-axios` 进行安装
  - 使用分两种
    - 局部：`import axios from 'axios'`
    - 全局
  - 用法：`axios.get(请求地址).then(res=>{})`

作业

把 `api.xin88.top` 的接口，按照效果图完成