Vue05

> 提前准备项目包 并 启动

复习

选项

- DOM
  - el: 脚本方式中，初始化时指定vue管理的元素
  - render: 脚手架项目包中，渲染 .vue 文件到 vue对象里
- 数据
  - data: 存放数据，可以在页面上使用
  - methods: 存放方法，可以在页面上使用
  - computed: 计算属性，存放方法
    - 特点: 使用时不需要()触发
- 资源
  - directives: 指令
    - 自定义指令都是 v- 开头
    - 两种语法
      - 值是函数: 在DOM元素创建时触发，还未展示到页面
      - 值是对象: 精确配置在哪个生命周期触发
        - 例如: inserted DOM元素显示到页面上之后 -- 调用获得焦点
- 指令
  - v-text: innerText
  - v-html: innerHTML
  - v-show: css方式 display:none 实现隐藏
  - v-if: 根据条件来加载/删除DOM元素;
    - 实战场景: 网络请求的数据不存在的时候先不加载元素，等存在之后在加载
    - v-else
    - v-else-if
  - v-for: 循环遍历生成元素
    - key: 唯一标识. 在数组的数据动态变化时 提升复用效率
    - 语法
      - v-for="值 in/of 数组"
      - v-for="(值, 序号) in/of 数组"
      - v-for="值 in/of 数字"
  - v-on:事件
    - vue2语法 @事件名
  - v-bind:属性
    - vue2语法 :属性名
  - v-model: 双向绑定
    - 方向1: 数据绑定在DOM元素上
    - 方向2: DOM元素发生变化，实时更新给数据
    - 适用范围: 表单元素 - 输入框，单选框，多选框，下拉框...
  - v-pre: 原样显示 {{}}
  - v-once: 一次性展示
    - 初始化显示，后续不再更新

组件

组件：组成页面的零件

- 用于拆分大型页面，拆分成多个独立的 .vue 文件 -- 团队合作，复杂->简单
- 存放在 components 目录里
- 文件名要求 最好 大驼峰，至少两个单词
  - 原因：怕和系统标签重名
- 推荐根元素的 class名 是 文件名的 中划线命名法
- 组件使用分三步： 引入 -=> 注册 -> 使用
  - 注册
    - 普通方式： components：{组件名}
    - 自定义方式： components:{ 自定义名称：组件 }
  - 使用：单标签 + 双标签 + 大驼峰 + 中划线 写法都支持，挑你喜欢的
    - 推荐：<xxx-xxx />

```
<template>
  <div>
    <!-- 组件：各大框架的核心竞争力功能 -->
    <!-- 组件：组成页面的零件 -->
    <!-- 活字印刷术，模块化：把一个大型的页面拆分成多个小模块 -->
    <!-- 优点：分工合作，复用，简化大型页面的制作 -->
    <!-- components：此文件夹专门存储组件 -->
    <h1>我是App.vue</h1>
    <!-- 使用组件 -->
    <!-- 单标签必须闭合，加 / -->
    <zz />
    <!-- 推荐此写法 -->
    <my-header />
    <MyHeader />

    <!-- 也支持双标签 -->
    <zz></zz>
    <my-header></my-header>
    <MyHeader></MyHeader>
  </div>
</template>

<script>
// 组件使用分三步：引入 注册 和 使用
// 步骤1：引入
import MyHeader from './components/MyHeader.vue'

export default {
  // 步骤2：注册
  // 配置项components：用来配置有哪些外来的组件
  // { 属性名：值 } 如果值是变量 {属性名：变量名} 如果一样则简化
  // 例如：{ zz: zz }  简化为 {zz}
  components: { MyHeader, zz: MyHeader }, // {MyHeader: MyHeader}
}
</script>

<style lang="scss" scoped>
// 遵守规矩：组件中的根元素的class名，是其文件名的中划线命名法
// 在使用时就能预测到class名，比较方便
// 组件在使用时，通常需要用css进行布局：调整摆放的位置
.my-header {
```

```
  margin-top: 10px;
}
</style>
```

header

```
<template>
  <div class="my-header">
    <!-- 要求1: 推荐组件名用 大驼峰命名法, 至少两个单词 -->
    <!-- 要求2: 推荐为根元素添加class, 名字为蛇形命名法 -->
    <span>网页</span>
    <span>新闻</span>
    <span>贴吧</span>
    <span>知道</span>
    <span>音乐</span>
    <span>图片</span>
  </div>
</template>

<script>
export default {}
</script>

<style lang="scss" scoped>
.my-header {
  background-color: #aaa;

  span {
    display: inline-block;
    margin: 5px;
  }
}
</style>
```

组件练习

```
<template>
  <div>
    <!-- vscode的自动辅助: 使用 <组件名  会自动引入 -->
    <!-- 极小概率会失败, 确保是 1对1服务的 Vscode  -->
    <my-footer />
  </div>
</template>

<script>
import MyFooter from './components/MyFooter.vue'
export default {
  components: { MyFooter },
}
</script>

<style lang="scss" scoped>
.my-footer {
  position: fixed;
  bottom: 0;
}
</style>
```

```
<template>
  <!-- 为什么要有代码规范？ -->
  <!-- 希望 多人协作开发时，最终的代码风格一致，看起来像一个人写的 -->
  <div class="my-footer">
    <!-- 为什么根元素要给class？  名字规范？ -->
    <!-- 组件在 App.vue 中组装时，通常需要布局，所以要给class便于选择 -->
    <!-- 名称规范：文件名改 中划线命名法，便于预测 -->
    <p>买家帮助</p>
    <div>
      <div>新手指南</div>
      <div>服务保障</div>
      <div>常见问题</div>
    </div>
  </div>
</template>

<script>
export default {}
</script>

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

生命周期

```
<template>
  <div>
    <!-- 组件的生命周期 -->
    <button @click="show = true">生</button>
    <button @click="show = false">死</button>
    <my-com v-if="show" />
  </div>
</template>

<script>
import MyCom from './components/MyCom.vue'
export default {
  components: { MyCom },
  data() {
    return {
      show: false,
    }
  },
}
</script>

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

```
<template>
  <div>
    <h1>我是组件</h1>
    <button @click="num++">{{ num }}</button>
  </div>
</template>
```

```
<script>
// 生命周期的两个用途：
// 1. 面试用：几乎必考
// 2. 配合网络请求用，实现自动发请求的操作
//  -- 例如：让组件显示到页面上后，立刻自动发请求

export default {
  data() {
    return {
      num: 1,
    }
  },
  // 生命周期：一个事物从生到死经历的过程
  // 人的周期：备孕->怀孕->待产->出生->开始学习->学习完毕->快没了->没了
  // 组件的一生：准备创建->创建完毕->准备显示->显示完毕->准备更新->更新完毕->准备销毁->销毁完毕
  // 作者为组件的每个周期添加了一个 钩子函数（Hook）
  // 钩子函数：一类特殊作用的函数的称呼，他们在固定的事件节点会自动触发

  // before：在...之前
  beforeCreate() {
    console.log('beforeCreate：将要创建-备孕')
  },
  created() {
    console.log('created：创建完毕-怀孕')
  },
  beforeMount() {
    // mount：安装,挂载
    console.log('beforeMount：将要安装到页面上-待产')
  },
  mounted() {
    console.log('mounted：安装到页面完毕-出生')
  },
  beforeUpdate() {
    console.log('beforeUpdate：更新前')
  },
  updated() {
    console.log('updated：更新完毕')
  },
  beforeDestroy() {
    console.log('beforeDestroy：将要销毁-快死了')
  },
  destroyed() {
    console.log('destroyed：销毁 - 死了')
  },
}
</script>

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

斗鱼

```
<template>
  <div v-if="data">
    <div class="cate">
      <span
        v-for="(x, i) in data.data"
        :key="x.cate2Id"
```

```
        @click="now = i"
        :class="{ active: now == i }"
      >
        {{ x.name }}
      </span>
    </div>
    <!-- 斗鱼组件 -->
    <!-- 属性传参：组件中利用props声明了 shortName属性 -->
    <!-- now：记录点击项的序号 -->
    <!-- 逻辑：根据序号从数组中读对应的栏目，找到其中的shortName -->
    <dou-yu :shortName="data.data[now].shortName" />
  </div>
</template>

<script>
import DouYu from './components/DouYu.vue'
export default {
  components: { DouYu },
  data() {
    return {
      data: null,
      now: 0, //当前序号
    }
  },
  // mount：安装,挂载 -- 当前组件显示在页面上时
  mounted() {
    console.log('this:', this) //vue实例对象
    // 通过打印，可以看到 axios存储在this 这个vue实例对象里
    // 所以所有的.vue文件，都可以通过 this.axios 来使用axios

    // 希望：页面显示出来后，立刻触发请求
    this.getData()
  },
  methods: {
    getData() {
      const url = 'https://douyu.xin88.top/api/cate/recList'
      // main.js中：简单粗暴的原型注入，使用时没有代码提示
      // 优雅方案：use + vue-axios ，使用axios有代码提示
      this.axios.get(url).then(res => {
        console.log(res)
        this.data = res.data
      })
    },
  },
}
</script>

<style lang="scss" scoped>
.cate {
  display: flex;
  span {
    margin: 10px;
    padding-bottom: 3px;
    user-select: none;

    &.active {
      border-bottom: 2px solid orange;
      color: orange;
    }
  }
}
```

```
</style>
```

```
<template>
  <!-- 考虑到：整个斗鱼页面都依赖请求数据，直接给根元素加v-if -->
  <div class="dou-yu" v-if="data">
    <p>shortName:{{ shortName }}</p>

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

<script>
// https://douyu.xin88.top/api/room/list?page=1&type=yz
export default {
  // property：属性。 props是此单词的缩写，代表 外来的属性们
  // 理解成 组件的形参，接收组件使用时传入的实参
  props: ['shortName'],
  data() {
    return {
      data: null,
    }
  },
  // 组件显示到页面上时，触发。称为 挂载
  mounted() {
    this.getData()
  },
  // 监听器：监听一个属性的变化
  watch: {
    // 你要监听的属性：function(){}
    shortName(to, from) {
      console.log('to:', to) //新值
      console.log('from:', from) //旧值
      // 当值变化时，触发请求，获取新的数据
      this.getData()
    },
  },
  methods: {
    getData() {
      const url = `https://douyu.xin88.top/api/room/list?
page=1&type=${this.shortName}`

      console.log('url:', url)

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

<style lang="scss" scoped>
.dou-yu > div {
  display: inline-block;
  width: 300px;
```

```
  margin: 0 10px 10px 0;

  img {
    width: 100%;
    height: 160px;
    border-radius: 4px;
  }
}
</style>
```

新闻

```
<template>
  <div v-if="data">
    <!-- 组件 -->
    <my-news :p="now" />

    <div class="pages">
      <span
        v-for="p in data.pageCount"
        :key="p"
        @click="now = p"
        :class="{ active: p == now }"
      >
        {{ p }}
      </span>
    </div>
  </div>
</template>

<script>
import MyNews from './components/MyNews.vue'
// http://www.codeboy.com:9999/mfresh/data/news_select.php
export default {
  components: { MyNews },
  data() {
    return {
      data: null,
      now: 1, //默认值1 代表第一页
    }
  },
  mounted() {
    this.getData()
  },
  methods: {
    getData() {
      const url =
        'http://www.codeboy.com:9999/mfresh/data/news_select.php'

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

```
<style lang="scss" scoped>
.pages {
  user-select: none;
  background-color: #eee;
  padding: 10px;
  display: flex;

  span {
    margin: 6px;
    border: 1px solid #777;
    color: #777;
    width: 40px;
    line-height: 40px;
    text-align: center;
    border-radius: 4px;

    &.active {
      color: white;
      background-color: orange;
      border-color: orange;
    }
  }
}
</style>
```

```
<template>
  <div class="my-news" v-if="data">
    <!-- <h1>新闻</h1> -->
    <div v-for="x in data.data" :key="x.nid">
      <span>{{ x.title }}</span>
      <!-- 过滤器: 可以处理{{}}中的值 -->
      <!-- 语法 {{ 值 | 过滤器名  }} -->
      <span>{{ x.pubTime | date }}</span>
    </div>
    <p>p: {{ p }}</p>
  </div>
</template>

<script>
//http://www.codeboy.com:9999/mfresh/data/news_select.php?pageNum=1

// 传参做法: 先生成参数 然后 再传递
export default {
  props: ['p'], // 专门保存自定义参数/属性

  // 新的配置项 filters
  filters: {
    // 过滤器的格式  {{值|过滤器}}
    // 例如 {{ 值 | date }}
    date(value) {
      // 参数1: |前面的值
      // 返回值就是过滤后的结果
      // 由于服务器返回的时间戳是字符串类型, 需要 *1 转数字
      var d = new Date(value * 1) //Date会识别参数类型, 数字才是时间戳
      var year = d.getFullYear()
      var month = d.getMonth() + 1
      // 小于10 补0      真 && 执行此处代码
      month < 10 && (month = '0' + month)
```

```
      var day = d.getDate()
      if (day < 10) day = '0' + day

      return `${year}/${month}/${day}`
    },
  },

  data() {
    return {
      data: null,
    }
  },
  mounted() {
    this.getData()
  },
  // 监听器:
  watch: {
    // p(){}
    // 变化后，重新发请求，不关心值是什么，所以不需要写形参
    p: function () {
      this.getData()
    },
  },

  methods: {
    getData() {
      const url = `http://www.codeboy.com:9999/mfresh/data/news_select.php?
pageNum=${this.p}`

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

<style lang="scss" scoped>
.my-news {
  width: 800px;

  div {
    display: flex;
    padding: 10px;
    border-bottom: 1px dashed gray;
    justify-content: space-between;
  }
}
</style>
```

axios模块安装

```
npm i axios vue-axios
```

全局引入

```
import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
// 全局引入axios：原理是把axios混入到vue实例对象里
import axios from 'axios'
// 简单粗暴：直接存到vue的原型里，使用时没有提示
// Vue.prototype.axios = axios

// 使用 vue-axios 模块，可以优雅的把 axios 注入到vue里
import VueAxios from 'vue-axios';
// use:使用；   vue提供的专门加载第三方模块的方法
Vue.use(VueAxios, axios) //参数顺序固定

Vue.config.productionTip = false

new Vue({
  router,
  store,
  render: h => h(App)
}).$mount('#app')
```

父子组件传参

> 类似：壮壮爸爸　给 壮壮一个手机
>
> - 壮壮需要做什么？准备好双手 接收手机

程序中：一个组件接收传入的数据，必须要提前声明变量 来接收

- 类似函数的 形参

关于斗鱼练习

- 在 App.vue 发送请求，获取分类数据，进行了展示
  - 设置了now属性，存储当前激活的分类序号
  - 点击分类是，now 会赋值为 点击的序号
- 组件的制作：发送请求 获取数据
  - 在 App.vue 中加载了组件
- 组件的内容可以根据 类型type 而发生变化
  - 组件使用 props 属性，设定了参数 shortName 可以接收外来的实参
- App.vue中，使用组件时，根据当前激活的序号now，读取对应shortName传给组件
- 组件中：
  - 使用监听器watch:监听 shortName属性的变化，然后重新触发请求 获取新的数据
  - 数据的拼接：url地址中的 type的值需要拼接对应shortName

过滤器

仅限在 {{}} 中使用，语法 {{ 值 | 过滤器 }}

通过 过滤器的处理，把返回值显示到页面上

- 新的配置项： `filters` 专门写过滤器

插槽

```
<template>
  <div>
    <!-- 插槽 slot -->
    <!-- 一个页面结构已经固定，就等着放东西 -->
    <!-- 留出的放东西的预定空间 就是插槽 -->
    <!-- 例如 电脑主板上的 CPU 显卡 内存 插槽 -->
    <!-- 例如 化妆盒中的空间： 化妆品，首饰... -->

    <!-- 对于插槽，必须用双标签书写 -->
    <my-slot>
      <div>
        <span>壮壮</span>
        <span>边边</span>
        <span>浩浩</span>
        <span>浩南</span>
      </div>

      <!-- 指定向 名字是 menu的插槽里放东西 -->
      <!-- 有3种语法：vue1  vue2  和 语法糖 -->
      <div slot="menu">这是菜单内容 vue1</div>

      <!-- vue2：必须配合 template 标签用 -->
      <!-- <template v-slot:menu>
        <h2>哈哈哈哈</h2>
        <h2>--柯瞻说</h2>
      </template> -->

      <!-- vue3：必须配合 template，语法糖# -->
      <template #menu>
        <h2>晕晕晕了...</h2>
        <h2>--壮壮</h2>
      </template>
    </my-slot>
  </div>
</template>

<script>
import MySlot from './components/MySlot.vue'
export default {
  components: { MySlot },
}
</script>

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

```
<template>
  <div class="my-slot">
    <!-- 一种特殊的组件：负责布局，没有实际内容 -->
    <div class="header">
      <!-- 插槽占位符组件：在运行时会替换成组件的标签内容 -->
      <slot />
    </div>
```

```
    <div class="body">
      <div class="menu">
        <!-- 命名插槽 -->
        <slot name="menu" />
      </div>
      <div class="content">
        <slot />
      </div>
    </div>
  </div>
</template>

<script>
export default {}
</script>

<style lang="scss" scoped>
.header {
  background-color: #eee;
  // 最小高度
  min-height: 100px;
}

.body {
  display: flex;

  .menu {
    background-color: yellowgreen;
    width: 30%;
    min-height: 400px;
  }

  .content {
    background-color: violet;
    width: 70%;
    min-height: 400px;
  }
}
</style>
```

总结

- 组件： 组成页面的零件
  - 把大型的网页拆分成多个小型的模块 -- 团队合作必备
- 组件放在 components 文件夹里
  - 名称要大驼峰 ，推荐至少两个单词 -- 避免和HTML自带的标签重名
  - 组件的根元素，最好添加class，名字是文件名的蛇形命名法
- 组件传参
  - 组件用 props 来声明参数
  - 使用组件时，通过 属性=值 的方式传递参数
- axios的全局引入方式
  - 在main.js中，使用固定的语法来引入
- watch：监听器，可以监听任意属性的变化
- filters：过滤器，配合 {{}} 使用，语法 {{ 值 | 过滤器 }}

作业

熟练今天的组件练习代码

完成考试题： https://ks.wjx.top/vm/wF707wM.aspx