Vue06

复习

选项

- DOM
  - el：脚本方式，设定 vue对象管理的元素
  - render：脚手架方式，加载 .vue 文件到vue对象里
- 数据
  - data：存储在这里的数据 可以到页面上用
  - props：组件声明形参/属性，接收外来的实参
  - computed：计算属性，这里的函数使用时不用()
  - methods：存储函数，可以到处使用
  - watch：监听器，监听任意属性的变化
- 资源
  - directives：自定义指令，用 v- 开头
  - filters：过滤器，可以转换 {{}}中的值
    - 格式：{{ 值 | 过滤器 }}
  - components：注册组件
    - 默认语法：components：{组件名} 组件名就是标签名
    - 自定义语法：components:{标签名：组件名}
- 生命周期钩子
  - 钩子函数：在固定事件发生时，自动触发的函数就叫 钩子函数
  - 创建：内存中,还没显示到页面
    - 创建前：beforeCreate
    - 创建完：created
  - 挂载：显示到页面上
    - 挂载前：beforeMount
    - 挂载完：mounted
  - 更新：页面发生更新
    - 更新前：beforeUpdate
    - 更新完：updated
  - 销毁：组件被移除
    - 移除前：beforeDestroy
    - 移除后：destroyed

指令

- v-text：innerText
- v-html：innerHTML
- v-show：css方式隐藏 display:none
- v-if： 根据条件 移除/添加DOM元素，适合网络请求场景
  - v-else
  - v-else-if
- v-for：遍历
  - key：唯一标识，提高数组内容变化时，页面重绘时的 重用效率
  - 3种语法
    - 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: 双向数据绑定
    - 搭配 表单元素：输入框，单选，多选，下拉... 这些用户可操作的组件
    - 作用：实时把组件的值 更新到绑定的变量上
- v-slot: 插槽
    - 组件负责布局操作，使用插槽作为占位符使用
    - 组件在使用时，其双标签写法中的内容，会替换掉插槽
    - 命名插槽: `<slot name='名字'/>`
        - 使用时有3种语法
        - slot='名字' - vue1
        - v-slot:名字 - vue2
        - #名字 - vue2语法的语法糖
- v-pre: 原样显示 {{}}
- v-once: 一次性展示，后续不更新

特殊属性

- ref: 快速绑定变量和元素， 代替了DOM的查询操作
    - 使用时从 `$refs` 中读取

axios

- 网络请求模块，需要手动安装`npm i axios vue-axios`
- 使用方式有两种
    - 单独引入：在.vue文件里，用import 引入axios，直接用
    - 全局引入：当采用组件开发时，多个.vue文件，要想使用axios，单独引入费劲
        - main.js 中引入即可
            - 简单粗暴的原型注入： `Vue.prototype.axios= axios`
                - 缺点：使用时没有代码提示。 `this.axios`
            - 优雅：采用 vue-axios 模块辅助引入
                - `Vue.use(VueAxios, axios)`
                - 优点：有代码提示。 `this.axios`

过滤器

```
<template>
  <div>
    <!-- 作业 -->
    <p>{{ 'how old are you' | upper }}</p>
    <p>{{ 'give you some color to see see' | upper }}</p>
    <!-- pow: 算次幂， 2的10次方 -->
    <p>{{ 2 | pow(10) }}</p>
    <p>{{ 2 | pow(3) }}</p>
    <!-- 练习: add过滤器，可以计算数字的总和 -->
    <p>{{ 10 | add(20, 30) }}</p>
    <!-- 过滤器可以连着用 -->
    <p>{{ 3 | pow(2) | add(89, 11) }}</p>
  </div>
</template>

<script>
export default {
```

```
  // 配置项
  filters: {
    add(value, n1, n2) {
      return value + n1 + n2
    },

    pow(value, exp) {
      return Math.pow(value, exp)
    },

    upper(value) {
      return value.toUpperCase()
    },
  },
}
</script>

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

路由系统

官网： https://router.vuejs.org/zh/

> 什么是路由系统：通过浏览器地址栏路径的变化，切换页面上显示的组件
>
> - 实现页面内容切换的效果
> - 竞争力技术： SPA - Single Page Application 单页应用
>   - 传统的网站：通过多个 HTML文件 实现多页开发
>   - VUE：单页应用，只有一个 index.html 文件
>   - 通过 局部 内容的切换，来实现页面的变化：效率更高,体验更好,服务器压力小

路由配置

```
import Vue from 'vue'
import VueRouter from 'vue-router'
import Zhuang from '../views/Zhuang.vue'
import Bian from '../views/Bian.vue'
import Hao from '../views/Hao.vue'
import NotFound from '../views/NotFound.vue'
import Home from '../views/Home.vue'

Vue.use(VueRouter)

// routes: 在这里设置 路径 和 组件的对应关系
// {地址:'合肥市xx区xx小区xx楼xx号', 房子:壮壮家}
const routes = [
  // 首页
  {
    path: '/', // /是根路径
    component: Home,
  },

  // 通配符: *  匹配任何找不到的路径
```

```
  {
    path: '*',
    component: NotFound,
  },

  {
    path: '/hh',
    component: Hao,
  },
  {
    path: '/zz', //路径  localhost:8080/zz
    // 对应的组件
    component: Zhuang,
  },
  {
    path: '/bb',
    component: Bian,
  },
]

const router = new VueRouter({
  mode: 'history',
  base: process.env.BASE_URL,
  routes,
})

export default router
```

```
// vrouter
import Vue from 'vue'
import VueRouter from 'vue-router'
// 非懒加载语法：提前引入组件，不管用户是否使用
// 适合访问频繁的页面，提前加载到内存里，用户访问速度更快，体验好
// 浪费内存，提升用户体验
import Home from '../views/Home.vue'

// 壮壮：
// 1. 壮壮提前购买食物，饿了的时候吃
// 2. 壮壮饿了，出去买吃的  -- 懒

Vue.use(VueRouter)
// 组件的两种引入方式
const routes = [
  {
    // 新式传参语法，把传统的 ?参数=值&参数=值 拆分
    // 通过 : 代表参数名
    path: '/kai/:age/:phone/:wife',
    name: '凯凯',
    props: true, // 默认值false,  true就代表允许用 props接收路由参数
    component: () => import('../views/Kai.vue'),
  },
  {
    path: '/about',
    name: 'About',
    component: () => import('../views/About.vue'),
  },
  {
    path: '/xu',
    name: '旭旭',
```

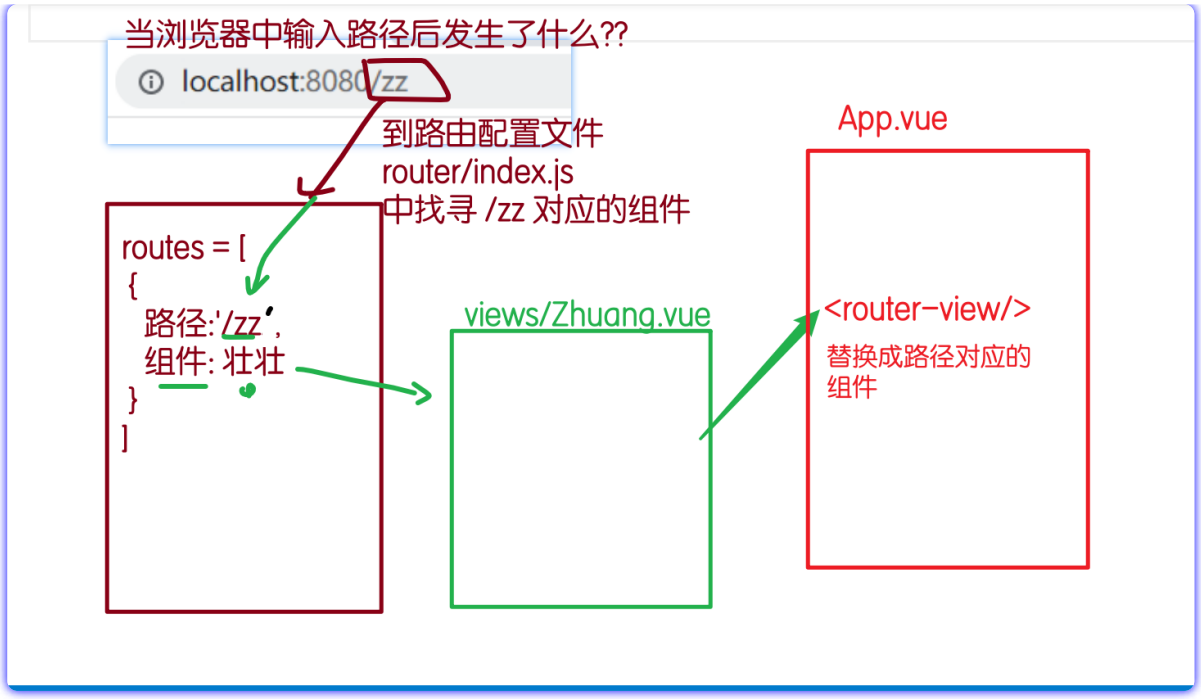
```
    component: () => import('../views/Xu.vue'),
  },
  {
    path: '/hh',
    name: '浩浩',
    component: () => import('../views/Hao.vue'),
  },
  {
    path: '*', //统配所有找不到路径的页
    name: '404',
    component: () => import('../views/NotFound.vue'),
  },
  // 懒加载写法用 vr 能自动生成
  // vroute-named
  {
    path: '/bb',
    name: '边边', //给这段路由关系起个名字，调试时使用
    component: () => import('../views/Bian.vue'),
  },

  {
    path: '/',
    component: Home,
  },
  {
    path: '/zz',
    // 懒加载引入方式：
    // 优点：不会提前加载组件到内存，不会浪费额外的资源
    // 例如：1个网站有100个页面，如果用户只访问了10个
    // - 懒加载方案：只需要用到10个的时候加载，剩余90个不用加载
    component: () => import('../views/Zhuang.vue'),
  },
]

const router = new VueRouter({
  base: process.env.BASE_URL,
  mode: 'history',
  routes,
})

export default router
```

路由




```
<template>
  <div>
    <!-- 使用路由系统，需要生成项目时必须勾选 Router 选项 -->

    <!--
      组件现在有两种用途：
      - 1. 组成页面的一部分，存储在 components 目录里
      - 2. 被路由切换：存储在 views 目录里

      //类似：壮壮的手纸放在 左右两个兜里
      -- 左边：上厕所    右边：吃饭擦嘴
     -->

    <h1>柯晗：哈哈哈哈哈</h1>
    <div class="box">
      <!-- 路由的占位符：不是真正显示的内容，在运行时会被替换 -->
      <router-view />
    </div>

    <h1>浩浩：肯德基真好吃，被踢也要吃</h1>
  </div>
</template>

<script>
export default {}
</script>

<style lang="scss" scoped>
.box {
  min-height: 200px;
  background-color: #aaa;
}
</style>
```

router-link

```
<template>
```

```
  <div>
    <!-- 点击方式做跳转 -->
    <div id="nav">
      <!-- 区别: a的跳转会刷新整个页面, 观察标签栏 -->
      <!-- router-link: 局部刷新 而不是整个页面 -- 体验更好,速度快 -->

      <!-- 之前用 a 跳转 -->
      <a href="/zz">壮壮</a>
      <!-- 现在: vue中要求用 router-link 代替 a -->
      <router-link to="/bb">边边</router-link>
      <router-link to="/">首页</router-link>
      <router-link to="/hh">浩浩</router-link>
    </div>

    <!-- 占位符: 主动查地址栏的路径, 到配置文件中找路径对应的组件,展示 -->
    <router-view />
  </div>
</template>

<script>
export default {}
</script>

<style lang="scss" scoped>
// router-link 最终表现出来的是 a 标签, 样式给a标签加即可
#nav {
  background-color: #eee;
  display: flex;
  a {
    padding: 10px 40px;
    color: white;
    background-color: #151e39;
    text-decoration: none;

    // 作者提前制作了激活时的 高亮样式, 我们可以直接用
    // 样式分两种: 精确的(exact) 和 模糊的
    &.router-link-exact-active {
      background-color: orange;
    }

    // 模糊版本:
    // 例子: 壮壮说: 喜欢 黄浩浩, 姓黄的, 黄浩, 黄浩浩 都喜欢
    // 距离: /bb/cc  匹配到 /  和 /bb  和 /bb/cc
    // 所以: /bb 会匹配到 / 和 /bb   首页是/ , 就会一直亮
    // &.router-link-active {
    //   background-color: orange;
    // }
  }
}
</style>
```

路由懒加载

```
<template>
  <div>
    <div>
      <router-link to="/">首页</router-link>
      <router-link to="/zz">壮壮</router-link>
```

```
      <router-link to="/bb">边边</router-link>
    </div>

    <!-- 占位符：主动查看地址栏，到配置文件中找到路径对应的组件，加载 -->
    <router-view />
  </div>
</template>

<script>
export default {}
</script>

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

编程式跳转

```
<template>
  <div>
    <!-- 路由系统：通过路径的变化，切换页面上显示的组件 -->
    <!-- SPA：单页应用。 网页只有index.html，通过局部的组件切换实现多页的效果。速度更快，体验更好。 -->

    <!-- 最终展示的是 a 标签。此方式切换路径不会造成页面刷新 -->
    <router-link to="/">首页</router-link>
    <router-link to="/zz">壮壮</router-link>
    <router-link to="/bb">边边</router-link>
    <!-- 编程式跳转：通过JS代码方式触发跳转 -->
    <div>
      <button @click="goHao">浩浩</button>
      <button @click="goBack">上一页</button>
    </div>

    <!-- 路由占位符：主动检测浏览器地址栏的路径，根据路径到配置文件中找到相关的组件，进行展示 -->
    <router-view />
  </div>
</template>

<script>
export default {
  methods: {
    goHao() {
      // main.js 中实例化Vue对象时,会注入 router 路由对象
      console.log(this) // 找到router属性
      // $router：就是注入到vue中的router对象
      // 这里包含了所有操作路由相关的方法
      // push：推。
      // 数组的push：推入一个新的元素到数组末尾
      // git的push：把代码推到服务器
      // 路由的push：把新的路径推出到浏览器的地址栏

      // 编程式跳转：不能重复跳转到当前路径
      // 在教室：旭旭跟浩浩说 -- 走，咱们去吃KFC
      // 在KFC：旭旭跟浩浩说 -- 走，咱们去吃KFC， 浩浩：你有病吧..

      // 做判断：获取当前路径信息
      // $route：当前所在路径的相关信息
      if (this.$route.path != '/hh') {
```

```
        this.$router.push('/hh')
      }
    },
    goBack() {
      // 上一页
      // go(n): n=0刷新  n>0前进n页   n<0后退n页
      this.$router.go(-1)
    },
  },
}
</script>

<style lang="scss" scoped>
a {
  display: inline-block;
  text-decoration: none;
  font-size: 1.5em;
  margin: 5px;
  color: #777;

  &:hover {
    color: blue;
  }

  // 作者为 router-link 提供了 active 动态样式，代表谁是当前项
  &.router-link-exact-active {
    // exact精确的
    color: blue;
  }
}
</style>
```

路由参数

```
<template>
  <div>
    <!-- 路由组件的传参 -->
    <!-- URL的参数语法  路径?参数=值&参数=值... -->
    <router-link to="/xu?age=20&like=唱歌">旭旭1</router-link>
    <router-link to="/xu?age=22&like=跳舞">旭旭2</router-link>
    <router-link to="/about?name=壮壮&age=25&phone=13598987778"
      >壮壮</router-link
    >
    <!-- 新式传参语法 -->
    <router-link to="/kai/23/18878787878/壮壮">凯凯</router-link>

    <!-- 路由占位符 -->
    <router-view />
  </div>
</template>

<script>
export default {}
</script>

<style lang="scss" scoped>
a {
  margin: 10px;
```

```
}
</style>
```

```
<template>
  <div>
    <!-- 面试常考题：说一说 $route里 query 和 params 属性的区别？ -->
    <!-- 传统方式传参：?语法，存放在query里 -->
    <!-- 新式传参：:参数名..   存储在params里 -->

    <h1>凯凯</h1>
    <p>年龄：{{ $route.params.age }}</p>
    <p>手机号：{{ $route.params.phone }}</p>
    <p>妻子：{{ $route.params.wife }}</p>
    <button @click="showParams">查看路由参数</button>
    <p>{{ age }}, {{ phone }}, {{ wife }}</p>
  </div>
</template>

<script>
export default {
  // 语法糖：直接用props声明属性接收路由参数
  // 此语法糖默认不开启，需要在路由配置中手动开启
  props: ['age', 'phone', 'wife'],

  methods: {
    showParams() {
      console.log(this.$route)
    },
  },
}
</script>

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

vuex

```
<template>
  <div>
    <!-- Vuex模块：全局状态共享 -->
    <!-- 在多个 .vue 文件中，共享数据的技术  -->
    <com-one />
    <com-two />
    <com-three />
  </div>
</template>

<script>
import ComOne from './components/ComOne.vue'
import ComThree from './components/ComThree.vue'
import ComTwo from './components/ComTwo.vue'
export default {
  components: { ComOne, ComTwo, ComThree },
}
</script>
```

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

```
import Vue from 'vue'
import Vuex from 'vuex'

Vue.use(Vuex)

// 到 main.js 查看:  store在vue初始化时，会注入到vue对象里
export default new Vuex.Store({
  // 严格模式下：不允许直接修改state中的值，必须特殊申请，才可以
  // 效果：会报错 但 依然能改
  strict: true, //严格模式：真
  // 5个核心属性
  // 状态：存储共享的数据项
  // 到后台查看vue对象，能不能找到这里
  state: {
    count: 1,
    uname: '壮壮',
    // 存储购物车的数据
    cart: ['旭旭', '剑桥'],
  },
  getters: {},
  // 存储方法，用来修改state中的值
  // 类似：壮壮说-要用纸的人 跟我申请，我帮你去拿
  mutations: {
    // 用于向 cart 属性里，添加中
    cartAdd(state, user) {
      // 参数1：固定的state,代表共享的数据
      // 其他参数看情况，比如此处要传递增加的 用户是谁
      state.cart.push(user) //数组新增数据到末尾
    },
    countAdd1(state) {
      // 固定的参数1：即共享的数据们
      state.count++
    },
  },
  actions: {},
  modules: {},
})
```

one

```
<template>
  <div class="com-one">
    <h1>组件1</h1>
    <!-- <button @click="count++">{{ count }}</button> -->

    <p>共享的count: {{ $store.state.count }}</p>
    <button @click="$store.state.count++">count+1</button>

    <!-- 把值存储在公共的区域，就可以在多个组件中使用 -->
    <button @click="showVue">查看vue对象</button>

    <hr />
    <!-- 如何触发 store中的 mutations 里的方法？commit -->
    <button @click="$store.commit('cartAdd', '壮壮')">
      把壮壮加入购物车
    </button>
```

```
    <button @click="$store.commit('cartAdd', '浩浩')">
      把浩浩加入购物车
    </button>
    <button @click="$store.commit('cartAdd', '兵兵')">
      把兵兵加入购物车
    </button>
  </div>
</template>

<script>
export default {
  methods: {
    showVue() {
      console.log(this) // 找到 store
    },
  },
  data() {
    return {
      count: 1,
    }
  },
}
</script>

<style lang="scss" scoped>
.com-one {
  background-color: brown;
  height: 300px;
  margin-bottom: 20px;
}
</style>
```

two

```
<template>
  <div class="com-two">
    <h1>组件2</h1>

    <!-- Vuex的基本原理：把数据共享，多个组件都使用共享数据 -->

    <!-- 场景：壮壮把新买的纸巾放在讲台上共享使用 -->
    <!-- 结果发现：所有班级的同学都来拿他的纸巾，一上午就没了 -->
    <!-- 开启严格模式： -->

    <p>共享的count: {{ $store.state.count }}</p>
    <button @click="$store.state.count++">count+1</button>

    <!-- $store提供了一个 commit(委托) 方法，委托store执行mutations中的函数 -->
    <button @click="$store.commit('countAdd1')">count+1</button>
    <hr />
    <div v-for="(item, i) in $store.state.cart" :key="i">
      {{ item }}
    </div>
  </div>
</template>

<script>
export default {}
</script>
```

```
<style lang="scss" scoped>
.com-two {
  height: 300px;
  background-color: aquamarine;
}
</style>
```

three

```
<template>
  <div class="com-three">
    <h1>组件3</h1>
    <p>{{ count }}, {{ uname }}, {{ cart }}</p>
    <button @click="cartAdd('陈胜')">陈胜加入</button>
  </div>
</template>

<script>
import { mapMutations, mapState } from 'vuex'
export default {
  // mutataions是存函数，在methods中引入
  methods: {
    // 引入vuex里的方法，然后直接用就可以
    ...mapMutations(['cartAdd']),
  },

  // 语法糖：快速在计算属性中引入 store.state 中的变量
  computed: {
    // ... : 展开语法
    // mapState：会自动遍历数组中的元素，生成计算属性 --
    // 对原理感兴趣的看扩展视频 -- 百度网盘(未录制)
    ...mapState(['count', 'uname', 'cart']),
  },
}
</script>

<style lang="scss" scoped>
.com-three {
  background-color: yellowgreen;
  min-height: 100px;
  margin-top: 10px;
}
</style>
```

总结

路由系统：

- router-view 占位符：路由系统提供的
  - 作用：会扫描 浏览器的地址栏路径，找到其对应的组件 进行展示
- 组件:如果是路由切换的组件，则必须存储在 views 目录里
  - 命名大驼峰，不需要必须2个单词
- 路由配置文件： router/index.js
  - 配置 path 路径 和 组件的对应关系

    - props属性：为true，代表允许组件通过 props 属性接收路由参数
    - 路径
        - *：通配符，匹配所有没有的路径
        - /：根路径 首页
        - /???：自定义的其他路径
- 传参的两种方式：
    - 传统：路径?参数=值&参数=值...
    - 新的：路径/值/值/值　需要配合 路由的配置 path:'/路径/:参数/:参数'

> router-view -> 查找浏览器地址栏 -> 到 index.js 中匹配 -> 找到组件

Vuex

全局状态共享：把数据在多个.vue文件中共享

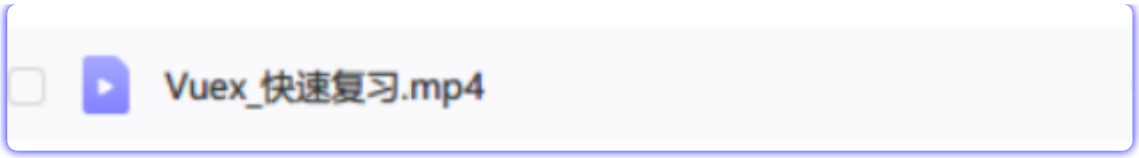
- 实现方案：把 store对象，存储到 vue的实例对象里 -- main.js
- 使用时
    - 把共享的数据存储在 store.state 属性里
    - 利用 $store.state 来读取属性
    - 为了安全性考虑：要修改属性必须通过指定的方法
        - 在mutations属性里，制作函数
        - 触发时，需要用 commit 来触发

提前下载项目资源

FTP也会传一份

周末

- 做考试题：最终月考题从 4个阶段的考试题里随机抽取
- 把今天的路由 和 vuex 要去练
    - 扩展视频：网盘中有纯净版的 Vuex 扩展视频，可以去看


Vuex_快速复习.mp4