复习

https://www.processon.com/view/link/5d1eb5a0e4b0fdb331d3798c

作业

属性更新的实现

知识点

编译器原理

v-if、v-model等实现原理

input type="text/radio" v-model="arr[0].foo"

```
_c('input',{directives:[{name:"model",rawName:"v-model",value:
(title),expression:"title"}],attrs:{"type":"text"},domProps:{"value":
(title)},on:{"input":function($event)
{if($event.target.composing)return;title=$event.target.value}}})],2)}
})

input.value = this.title
input.addEventl('input', function($event)
{if($event.target.composing)return;title=$event.target.value}}})],2)}
})
```

vue项目最佳实践

项目配置

创建vue.config.js，指定应用上下文、端口号、主页title

```
// vue.config.js
const port = 7070;
const title = "vue项目最佳实践";

module.exports = {
  publicPath: '/best-practice', // 部署应用包时的基本 URL
  devServer: {
    port: port,
  },
  configureWebpack: {
    // 向index.html注入标题
    name: title
  }
};
```

```
// index.html
<title><%= webpackConfig.name %></title>
```

链式操作：svg icon引入

安装依赖：svg-sprite-loader

```
npm i svg-sprite-loader -D
```

下载图标，存入src/icons/svg中

修改规则和新增规则，vue.config.js

```
// resolve定义一个绝对路径获取函数
const path = require('path')

function resolve(dir) {
  return path.join(__dirname, dir)
}
//...
chainWebpack(config) {
    // 配置svg规则排除icons目录中svg文件处理
    config.module
        .rule("svg")
            .exclude.add(resolve("src/icons"))
            .end();
    // 新增icons规则，设置svg-sprite-loader处理icons目录中的svg
    config.module
        .rule("icons")
            .test(/\.svg$/)
                .include.add(resolve("src/icons"))
                .end()
            .use("svg-sprite-loader")
            .loader("svg-sprite-loader")
                .options({ symbolId: "icon-[name]" })
                .end();
}
```

图标自动导入

```
// icons/index.js
const req = require.context('./svg', false, /\.svg$/)
req.keys().map(req);

// main.js
import './icons'
```

创建SvgIcon组件，./components/SvgIcon.vue

```
<template>
  <svg :class="svgClass" aria-hidden="true" v-on="$listeners">
    <use :xlink:href="iconName" />
  </svg>
```

```
</template>

<script>
export default {
  name: 'SvgIcon',
  props: {
    iconClass: {
      type: String,
      required: true
    },
    className: {
      type: String,
      default: ''
    }
  },
  computed: {
    iconName() {
      return `#icon-${this.iconClass}`
    },
    svgClass() {
      if (this.className) {
        return 'svg-icon ' + this.className
      } else {
        return 'svg-icon'
      }
    }
  }
}
</script>

<style scoped>
.svg-icon {
  width: 1em;
  height: 1em;
  vertical-align: -0.15em;
  fill: currentColor;
  overflow: hidden;
}
</style>
```

权限控制和动态路由

路由定义

路由分为两种：`constantRoutes` 和 `asyncRoutes`，router.js

```
import Vue from "vue";
import Router from "vue-router";
import Layout from '@/layout'; // 布局页

Vue.use(Router);

// 通用页面：不需要守卫，可直接访问
export const constRoutes = [
  {
```

```
    path: "/login",
    component: () => import("@/views/Login"),
    hidden: true // 导航菜单忽略该项
  },
  {
    path: "/",
    component: Layout,// 应用布局
    redirect: "/home",
    children: [
      {
        path: "home",
        component: () =>
          import(/* webpackChunkName: "home" */ "@/views/Home.vue"),
        name: "home",
        meta: {
            title: "Home", // 导航菜单项标题
            icon: "qq" // 导航菜单项图标
        }
      }
    ]
  }
];

// 权限页面：受保护页面，要求用户登录并拥有访问权限的角色才能访问
export const asyncRoutes = [
  {
    path: "/about",
    component: Layout,
    redirect: "/about/index",
    children: [
      {
        path: "index",
        component: () =>
          import(/* webpackChunkName: "home" */ "@/views/About.vue"),
        name: "about",
        meta: {
            title: "About",
            icon: "qq",
            roles: ['admin', 'editor']
        },
      }
    ]
  }
];

export default new Router({
  mode: "history",
  base: process.env.BASE_URL,
  routes: constRoutes
});
```

创建布局页面，layout/index.vue

```
<template>
  <div class="app-wrapper">
    <!-- <sidebar class="sidebar-container" /> -->
    <div class="main-container">
      <router-view />
    </div>
  </div>
</template>
```

创建用户登录页面，views/Login.vue

```
<template>
  <div>
    <h2>用户登录</h2>
    <div>
      <input type="text" v-model="username">
      <button @click="login">登录</button>
    </div>
  </div>
</template>
<script>
export default {
  data() {
    return {
      username: "admin"
    };
  },
  methods: {
    login() {}
  }
};
</script>
```

测试效果~

用户登录状态维护

通过vuex维护用户登录状态：用户登录 -> 获取token并缓存

vuex根模块实现，创建store/index.js

```
import Vue from 'vue'
import Vuex from 'vuex'
import user from './modules/user'

Vue.use(Vuex)

const store = new Vuex.Store({
  modules: {user}
})

export default store
```

user模块：维护用户数据、处理用户登录等，store/modules/user.js

```
const state = {
  token: localStorage.getItem('token'),
  // 其他用户信息
};

const mutations = {
  SET_TOKEN: (state, token) => {
    state.token = token;
  }
};

const actions = {
  // 模拟用户登录
  login({ commit }, userInfo) {
    const { username } = userInfo;
    return new Promise((resolve, reject) => {
      setTimeout(() => {
        if (username === "admin" || username === "jerry") {
          commit("SET_TOKEN", username);
          localStorage.setItem('token', username);
          resolve();
        } else {
          reject("用户名、密码错误");
        }
      }, 1000);
    });
  }
};

export default {
  namespaced: true,
  state,
  mutations,
  actions
};
```

请求登录，Login.vue

```
login() {
    this.$store
        .dispatch("user/login", { username: this.username })
        .then(() => {
        this.$router.push({
            path: this.$route.query.redirect || "/"
        });
    }).catch(error => {
        alert(error);
    });
}
```

路由守卫

创建./src/permission.js，**并在main.js中引入**

```
import router from './router'
```

```
const whiteList = ['/login'] // 无需令牌白名单

router.beforeEach(async (to, from, next) => {

  // 获取令牌判断用户是否登录
  const hasToken = localStorage.getItem('token')

  // 已登录
  if (hasToken) {
    if (to.path === '/login') {
      // 若已登录没有必要显示登录页，重定向至首页
      next({ path: '/' })
    } else {
      // 去其他路由，暂时放过
      next()
      // 接下来执行用户角色逻辑，todo
    }
  } else {// 未登录
    if (whiteList.indexOf(to.path) !== -1) {
      // 白名单中路由放过
      next()
    } else {
      // 重定向至登录页
      next(`/login?redirect=${to.path}`)
    }
  }
})
```