Gallery

技术说明文档

2017-12-7

吴志成

目录

技术栈

1. 前端

框架：React.js+Redux（管理数据状态）+dva（基于 redux、redux-saga 和 react-router 的轻量级前端框架）

UI 库：ant-design

工具：Eslint（代码规范工具）、Sass（css 预处理器）、webpack（打包工具）、webstorm（IDE）

2. 后端

语言：Node.js

框架：express

数据库：SQLite

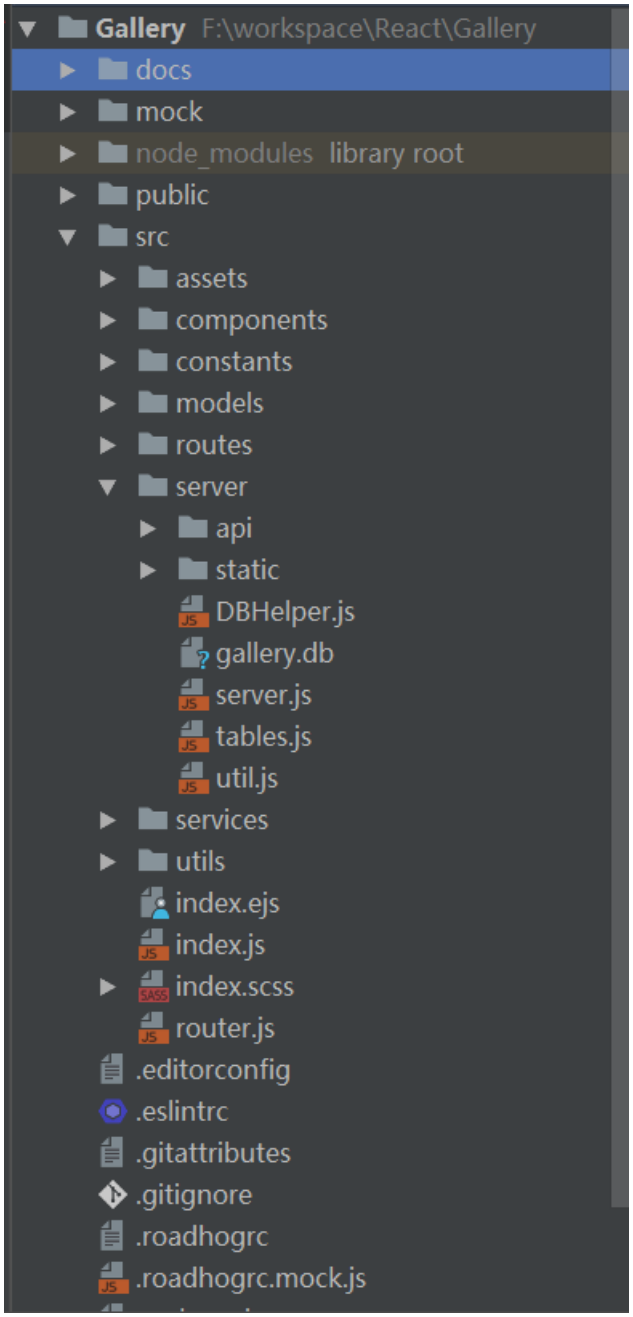
测试：mocha

集成工具：travis

项目架构

文件目录如下

前端采用 MVVM 架构



docs 文件夹：项目文档

assets 文件夹：存放前端静态资源

components 文件夹：前端组件

constants 文件夹：定义全局常量，比如 Http 状态码以及消息

models 文件夹：管理前端组件界面数据

routes 文件夹：前端界面路由

server 文件夹：服务器代码，其中 api 文件夹定义前端访问服务器的 api；static 文件夹存放服务器静态资源，如图片等，DBHelper.js 用于数据库操作。server.js 服务器程序入口。

services 文件夹：前端访问服务器的接口。

utils 文件夹：存放工具类

index.js ：前端程序入口

router.js ：管理前端路由跳转

关键实现

前端部分

1. 弹出框三角形的实现

   用 before 和 after 写出两个三角形，一个叠在另外一个上面。

   网上有一种设置 border-width 为 20px（举个例子），width，height 为 0 的 div ，把其他 border 的颜色设置为 transparent 即可实现三角形。

2. Follow Button 样式不生效

   class 选择器的问题，在 Follow Button 最外层的 div 设置一个 class。增加多一层 class 选择即可让样式生效。

3. 图片悬停出现悬浮层

```
<div class="container">
   <img alt="item" src={src} />
   <div class="info">
```

```
      <h3>test content</h3>
    </div>
 </div>

.container {
  position: relative;
}
.info {
    position: absolute;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    //-webkit-backface-visibility: hidden;
    //backface-visibility: hidden;
    background-image: linear-gradient(180deg,rgba(0,0,0,.2)
0,transparent 40%,transparent 60%,rgba(0,0,0,.3));
    opacity: 0;
    @include transition(all .15s ease-in-out);
  }

  .container:hover .info {
    opacity: 1;
  }
```

4. first-child 生效

   `p:first-child` 第一个 p

   `p :first-child` p 的第一个子元素

5. 解决导航栏弹出框被其他 div （假设为 a）挡住

   有两个关键点

1. 弹出框的父元素 div （也可能是祖父元素或者就是弹出框本身元素，总之一定是要跟 a 同级的元素）的 z-index 属性比 div a 的大
2. 只有元素设置了 position 属性（并且是非默认的 static） z-index 才生效

6. 保证图片填充父级元素且不模糊，保持原来的比例

```
<div><img /></div>


// scss 写法
div {
  position: relative;

  img {
     position: absolute;
    left: 0;
    width: 100%;
    height: 100%;
    object-fit: cover;
  }
}
```

object-fit 的五个值

- **fill**: 中文释义“填充”。默认值。替换内容拉伸填满整个 content box, 不保证保持原有的比例。
- **contain**: 中文释义“包含”。保持原有尺寸比例。保证替换内容尺寸一定可以在容器里面放得下。因此，此参数可能会在容器内留下空白。
- **cover**: 中文释义“覆盖”。保持原有尺寸比例。保证替换内容尺寸一定大于容器尺寸，宽度和高度至少有一个和容器一致。因此，此参数可能会让替换内容（如图片）部分区域不可见。
- **none**: 中文释义“无”。保持原有尺寸比例。同时保持替换内容原始尺寸大小。
- **scale-down**: 中文释义“降低”。就好像依次设置了 none 或 contain, 最终呈现的是尺寸比较小的那个。

7. 移动到 container 显示遮盖层 info 的实现

```
.container {
  position: relative;
```

```
  .info {
    position: absolute;
    top: 0;
    bottom: 0;
    left: 0;
    right: 0;
    background-image: linear-gradient(180deg,rgba(0,0,0,.2)
0,transparent 40%,transparent 60%,rgba(0,0,0,.3)); // 遮盖层颜色
    opacity: 0; // 相当于隐藏
    transition: all .15s ease-in-out;
    }

  &:hover .info {
    opacity: 1; // 显示遮盖层
  }
}
```

8. 背景图片蒙版的实现

```
<div class="background" style="background-image: url('图片路径')">
    <div class="container"></div>
</div>


.background {
  position: relative;
  background-position: 50%;
  background-size: cover;
  z-index: 1;

  &:before { // 蒙版效果
    position: absolute;
    right: 0;
    left: 0;
    display: block;
    height: 100%;
    content: "";
    background: linear-gradient(hsla(0, 0%, 100%, .8), hsla(0, 0%,
100%, .9) 50%, $theme-purple-color-ultra-light);
  }
```

```
  .container {
    position: relative; // 让 container 不会被蒙版遮盖住
  }

}
```

后端部分

1. 数据库操作

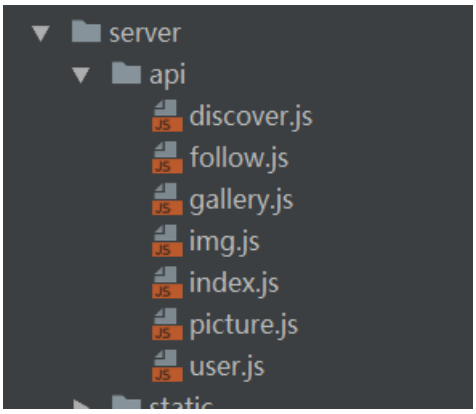
   将数据库的连接，sql 语句操作进行封装，放在 DBHelper.js 中，以减少代码量，提供复用率和可维护性

```
/**
 * Created by Hitigerzzz on 2017/12/4.
 */
const SQLite3 = require('sqlite3').verbose();
const path = require('path');

const DATABASE_FILE = path.join(__dirname, 'gallery.db');
let db;

exports.connect = () => {
  return new Promise((resolve, reject) => {
    db = new SQLite3.Database(DATABASE_FILE, (err) => {
      if (err) reject(new Error(err));
      resolve('connect database successfully');
    });
  });
};

exports.sql = (sql, param, mode) => {
  return new Promise((resolve, reject) => {
    db[mode](sql, param, (err, data) => {
      if (err) {
        reject(new Error(err));
      } else {
        resolve(data);
      }
    });
  });
};
```

2. 前后端通信

前后端通信通过 Restful 实现，前端通过 api 访问服务器，服务器返回对应的 json 数据。

api



封装返回 json 数据函数 responseClient

```
responseClient(res, httpCode = 500, code = 3, message = '服务端异常', data = {}) {
  const responseData = {};
  responseData.code = code;
  responseData.message = message;
  responseData.data = data;
  res.status(httpCode).json(responseData);
},
```