

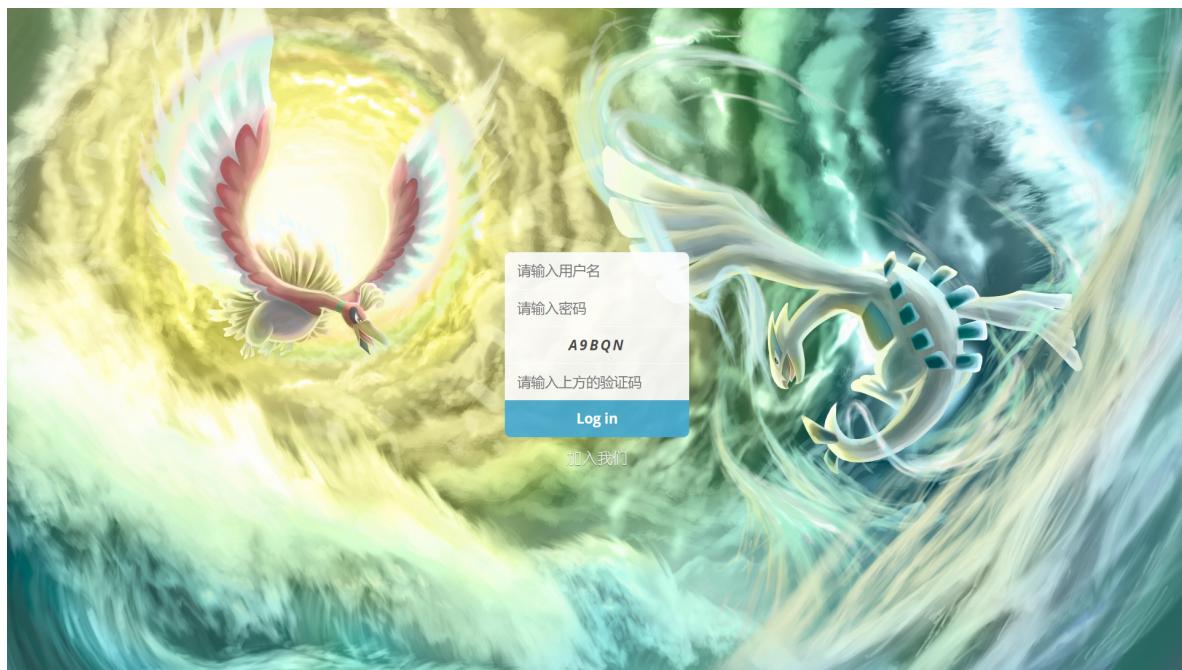
# Web作业 总说明文档

邝鸿燊 171250560

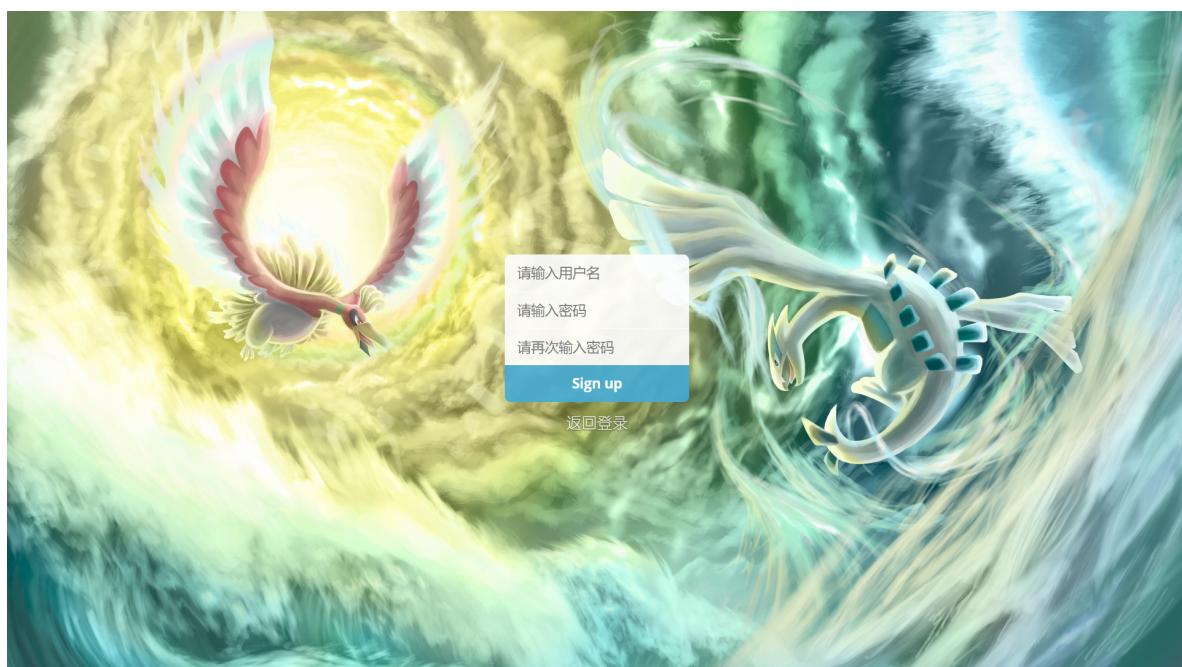
## 作业2 登录注册

### 界面截图

登录：



注册：



主页面：



## 实现功能：

浏览器端与服务器端分开，验证码，通信内容加密，服务器端使用nodejs

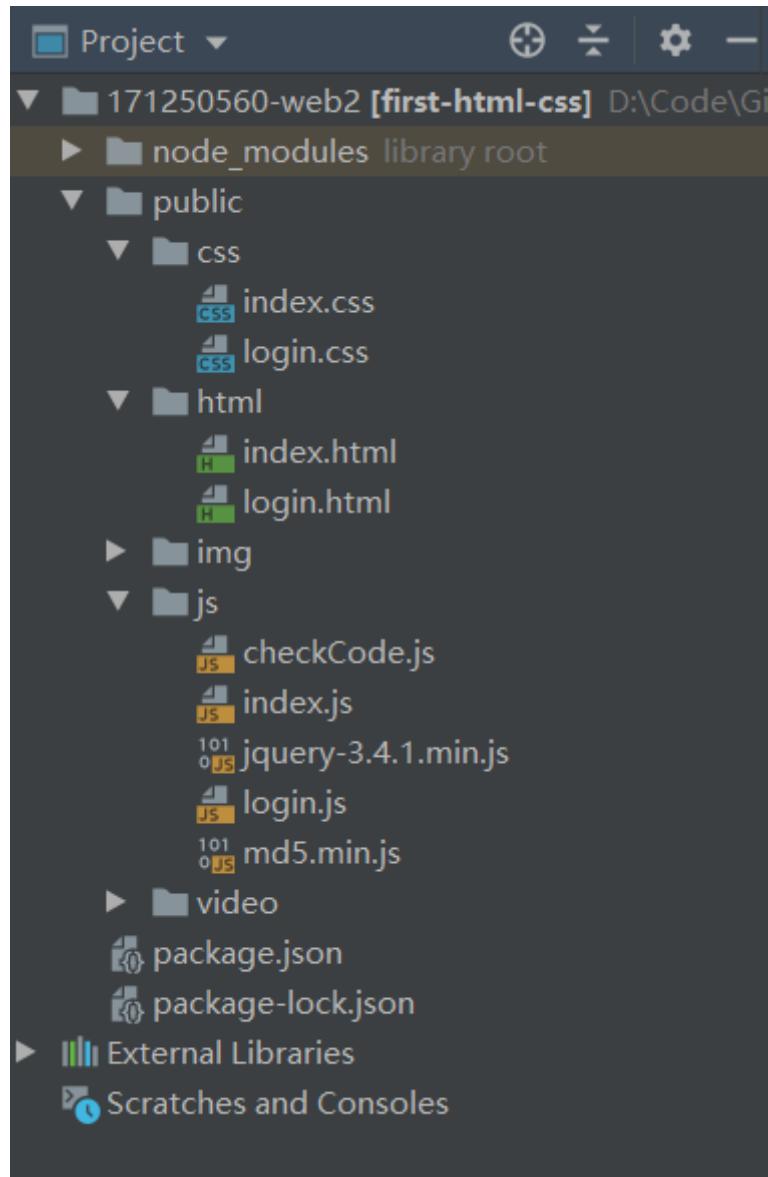
浏览器端项目为：171250560-web2

服务器端项目为：171250560-web2-server

使用技术：nodejs, express, md5加密, jQuery, Ajax

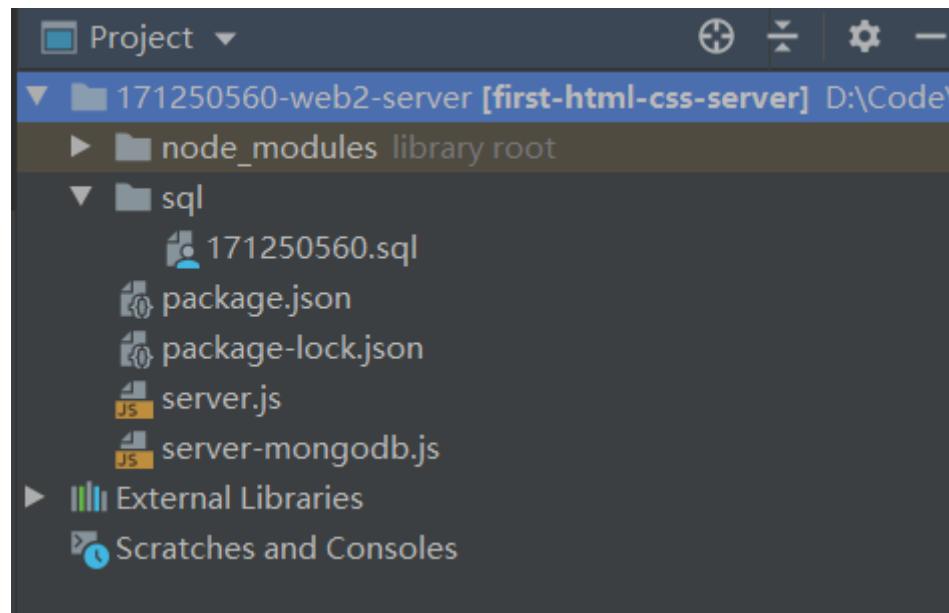
## 目录结构

### 浏览器端



- /public - 静态资源文件目录
  - /css - html需要的css文件
  - /html - 两个主页面
    - index.html - 登陆后的主页
    - login.html - 登录和注册的页面
  - /img - 网页中需要的图片
  - /js - html需要的js文件
    - checkCode.js - 生成验证码与验证验证码
    - md5.min.js - 通过MD5对内容进行加密
    - jquery-3.4.1.min.js - jQuery的js文件
  - /video - 视频

## 服务器端



- /sql - mysql文件
- express本地服务器, 端口: <http://127.0.0.1:8082>
- server.js - express服务端, 使用MySQL存储
- server-mongodb.js - 使MongoDB存储的版本

### 服务端的另一种尝试：HTTP

从server.js的130行开始

通过http.createServer创建服务端, 监听3002端口, 将前端ajax发送过来的请求进行解析, 并做出相应的处理。

与express的方法的不同之处在于, 无论前端的请求是get还是post, 都可以在createServer的function里进行处理, 而express需要对同一请求url设置get和post的两个处理方法。

```
130 // http服务端版本
131 let server = http.createServer( requestListener: function(req : IncomingMessage , res : ServerResponse ) {
132     let urlObj = url.parse(req.url, parseQueryString: true);
133     const urlPathname = urlObj.pathname;
134
135     const pathname = url.parse(req.url).pathname; // 请求的文件名
136
137     // 用于保存拼接后的请求体
138     let post = '';
139     // 'data' 事件触发, 将接受的数据块 chunk 拼接到 post 变量上
140     req.on( event: 'data' , listener: function (chunk) {
141         post += chunk;
142     });
143     // 请求完毕, 'end' 事件触发
144     req.on( event: 'end' , listener: function () {
145         // querystring 是 Node.js 自带模块, parse 方法用于将查询字符串解析成对象
146         const queryObj = querystring.parse(post);
147         const query = urlObj.query;
148
149         // 将接收的 POST 请求体以 JSON 格式响应回客户端
150         // res.writeHead(200, { "Content-Type": "text/plain" });
151         // res.write(JSON.stringify(queryObj));
152         // res.end();
153         if (urlPathname === "/post-login")
154         {
155             console.log(urlPathname, query, queryObj);
156             console.log("UserName: " + queryObj.username);
157             console.log("Password: " + queryObj.password);
158             console.log("UserName: " + query.username);
159             console.log("Password: " + query.password);
160             res.write( chunk: '{"login-success":true,"msg":"注册成功"}');
161             res.end();
162         }
163     });
164 };
165
166
167     myReadFile(res, path.join("./html/index.html"));
168     break;
169     case "/registered":
170         myReadFile(res, path.join("./html/registered.html"));
171         break;
172     case "/login":
173         case "/":
174             myReadFile(res, path.join("./html/login.html"));
175             break;
176         default: // 其他静态资源文件
177             myReadFile(res, path.join(".", pathname));
178             break;
179         }
180         console.log("Pathname:" + pathname);
181     }
182
183 });
184
185 // if (req.method == 'POST') {
186
187
188 });
189 });
190
191 server.listen( port: 3002, hostname: "127.0.0.1", listeningListener: function() {
192     let host = server.address().address;
193     let port = server.address().port;
194     console.log("服务器运行中.");
195     console.log("Address: http://%s:%s", host, port);
196 })
```

```
167     myReadFile(res, path.join("./html/index.html"));
168     break;
169     case "/registered":
170         myReadFile(res, path.join("./html/registered.html"));
171         break;
172     case "/login":
173         case "/":
174             myReadFile(res, path.join("./html/login.html"));
175             break;
176         default: // 其他静态资源文件
177             myReadFile(res, path.join(".", pathname));
178             break;
179         }
180         console.log("Pathname:" + pathname);
181     }
182
183 });
184
185 // if (req.method == 'POST') {
186
187
188 });
189 });
190
191 server.listen( port: 3002, hostname: "127.0.0.1", listeningListener: function() {
192     let host = server.address().address;
193     let port = server.address().port;
194     console.log("服务器运行中.");
195     console.log("Address: http://%s:%s", host, port);
196 })
```

## 存储账号信息的数据库

使用数据库： MySql

Database名： web\_database\_171250560

Table名： account\_web\_171250560

```
mysql> use web_database_171250560
Database changed
mysql> select * from account_web_171250560;
+-----+-----+
| username | password |
+-----+-----+
| Mr.MySql3 | b5d9485633ab80309cefefeb8781c85ff8 |
| pokemon | 08a3aa2cd4c129300d494fddd10fb15a |
+-----+-----+
2 rows in set (0.00 sec)
```

已存在的两个账号：

#1用户名： Mr.MySql3

密码： connection2

#2用户名： pokemon

密码： beamaster

## 验证码



由前端随机生成验证码，点击验证码可以更换验证码，验证码的验证由前端负责

点击“加入我们”，登录框会变化为注册框

验证码算法：/js/checkCode.js

原理：从[0-9A-Z]中随机选取5个字符作为当前验证码，并将该验证码字符串用全局函数code保存下来，同时将html中该元素的值刷新为本次验证码，css渲染过后显示为当前的验证码。

```
login.js x checkCode.js x login.html x index.html x login.css x
1 let code; //在全局定义验证码
2 //产生验证码
3 window.onload = function() {
4     createCode();
5 }
6
7 function createCode() {
8     code = "";
9     let codeLength = 5; //验证码的长度
10    let checkCode = document.getElementById("checkCode");
11    let random = new Array(items: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K',
12                           'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z'); //随机数
13    for(let i = 0; i < codeLength; i++) { //循环操作
14        let charIndex = Math.floor(Math.random() * 36); //取得随机数的索引
15        code += random[charIndex]; //根据索引取得随机数加到code上
16    }
17    checkCode.value = code; //把code值赋给验证码
18 }
19 //校验验证码
20 function validate() {
21     let inputCode = document.getElementById("input-check-code").value.toUpperCase(); //取得输入的验证码并转化为大写
22     if(inputCode.length <= 0) { //若输入的验证码长度为0
23         alert("请输入验证码!"); //则弹出请输入验证码
24         createCode(); //刷新验证码
25         return false;
26     } else if(inputCode != code) { //若输入的验证码与产生的验证码不一致时
27         alert("验证码输入错误!"); //则弹出验证码输入错误
28         createCode(); //刷新验证码
29         return false;
30     }
31     else
32 }
```

/public/css/login.css, 前端css渲染

```
.check-code {
    font-family: Arial;
    font-style: italic;
    font-weight: bold;
    border: 0;
    letter-spacing: 3px;
    color: blue;
}
```

## 加密

前端检查验证码正确后会将密码通过MD5进行加密，然后将加密过后的MD5和用户名发到服务器端

▼ Form Data [view source](#) [view URL encoded](#)

**action:** login  
**username:** goood  
**password:** e10adc3949ba59abbe56e057f20f883e

加密通过md5.min.js的md5()方法来进行加密

该算法出处: <https://github.com/blueimp/JavaScript-MD5>

## 通信

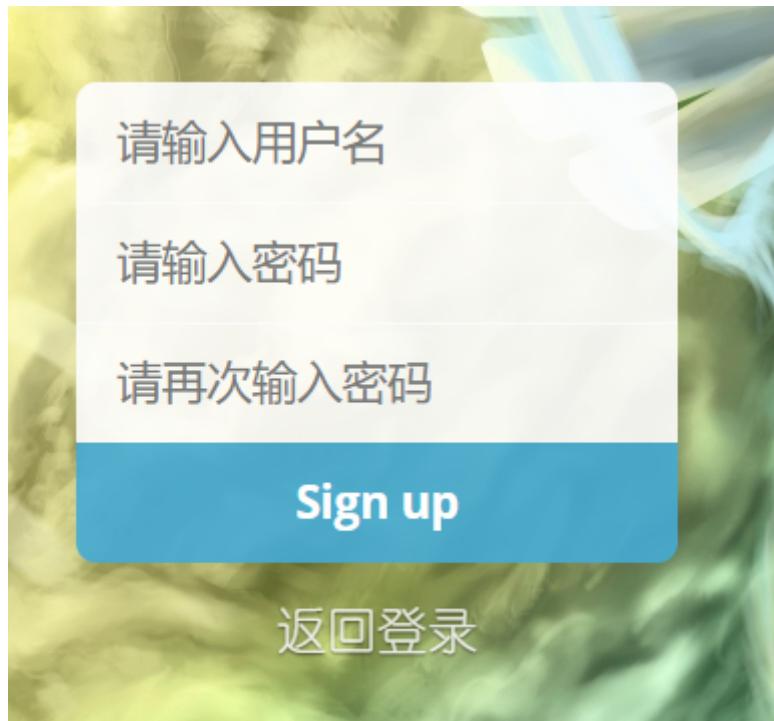
浏览器端通过jQuery选取元素，然后使用ajax向服务器端发送信息

```
// 登录
$("#log-in-button").on( events: "click", handler: function () {
    if (validate()) { // 验证输入的验证码
        $.ajax( settings: {
            // url: "http://127.0.0.1:8082/log-in/" + $("#username").val() + "/" + $("#password").val(),
            url: "http://127.0.0.1:8082/log-in",
            type: "post",
            data: {
                action: "login",
                username: $("#username").val(),
                password: md5($("#password").val()) // 加密
            },
            success: function (res) {
                if (res.status) // 账号密码验证成功
                {
                    alert(res.msg); // 打印信息
                    window.location = "./index.html"; // 跳转
                } else {
                    // alert(res.msg); // 此处打印详细的登陆错误信息
                    alert("账户信息有误，请重新输入");
                }
            },
            error: function (err :jqXHR<any> ) {
                alert("error!");
                alert(err);
            }
        });
    }
    else
    {
        alert("请输入正确的验证码");
    }
});
```

## 验证身份

服务器端收到用户名和加密后的密码，与数据库中的账户信息进行对比，并将对比结果返回浏览器端

## 注册



输入注册信息，浏览器端验证过格式正确后会将信息发送到服务器端，服务器端验证数据库中是否存在已有的用户名，如果没有就注册成功

前端格式验证：用户名至少6个字符，密码至少6个字符，两次密码必须一致

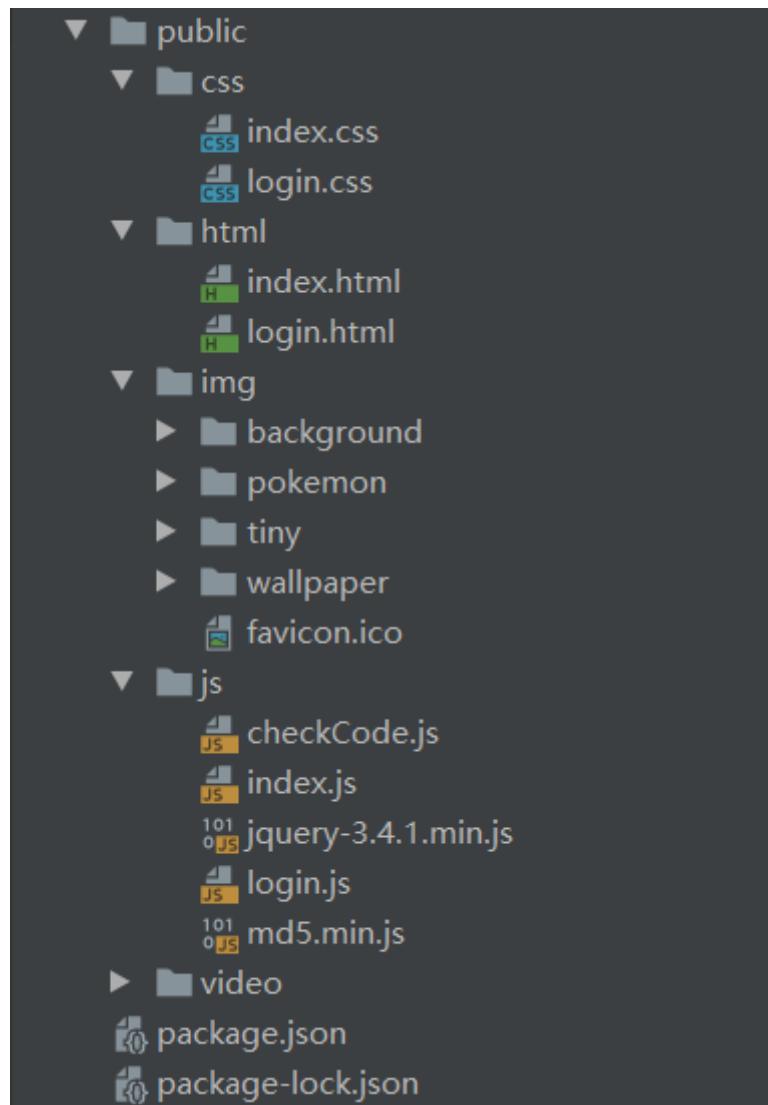
密码也会通过MD5加密过后才发送到服务器端

## 作业3 图片加载

网页交互展示：



## 目录结构



/public - 静态文件存放目录

- /html - 本次作业的实现在index.html中
- /css - 本次作业的实现在index.css中
- /js - 本次作业的实现在index.js中
- /img - 图片存放
  - /background - 背景图片
  - /pokemon - 左侧浮动栏内的图片
  - /tiny - 所有图片的略缩图
  - /wallpaper - 大图片瀑布流中的图片
  - favicon.ico - 网站标签的icon图片
- /video - 页面中的视频

## 使用技术

为了用户在低网速下访问网站的体验，要避免在加载多张图片时用户的等待时间，因此在网站中结合使用了**懒加载**和**渐进式加载**两种方法。

### 懒加载

页面中的图片主要分成三部分：

- 背景图片

由于我的背景图大部分都是同一个颜色，因此在加载背景图之前先将背景颜色设置为背景图的主颜色，优化用户体验

```
body {  
    background-color: #DFF4FC;  
    background-image: url("../img/background/Legendary-Birds.png"); /*背景图片*/  
    background-repeat: no-repeat; /*不平铺*/  
    background-size: cover; /*自动缩放到屏幕大小*/  
    background-attachment: fixed; /*背景不随内容而滚动*/  
    height: 95%;  
    width: 95%;  
    font: 12px/14px "Microsoft Yahei",arial;  
}
```

- 左侧浮动栏中的图片



```

<div id="model" style="..." >
    <div class="ad_container">
        <div class="box click-div">
            
        </div>
        <div class="box click-div">
            
        </div>
        <div class="box click-div">

            
        </div>
        <div class="box click-div">
            
        </div>
        <div class="box click-div">
            
        </div>
        <div class="box click-div">
            
        </div>
        <div class="box" id="log-out-div">
            
            <p>登出</p>
        </div>

    </div>
</div>

```

- 大图片瀑布流



```

<div id="model" style="..." >
    <div class="ad_container">
        <div class="box click-div">
            
        </div>
        <div class="box click-div">
            
        </div>
        <div class="box click-div">

            
        </div>
        <div class="box click-div">
            
        </div>
        <div class="box click-div">
            
        </div>
        <div class="box click-div">
            
        </div>
        <div class="box" id="log-out-div">
            
            <p>登出</p>
        </div>

    </div>
</div>

```

在这三部分图片中，有多图片的两部分都加载了略缩图

而原图url放在每张图片的data-src属性中

略缩图目录为/public/img/tiny， 略缩图大小在10kb以内

名称	日期	类型	大小	标记
080.jpg	2019/12/24 9:38	Image (jpg) File	2 KB	
254_2.jpg	2019/12/24 9:38	Image (jpg) File	4 KB	
257_2.jpg	2019/12/24 9:38	Image (jpg) File	4 KB	
260_2.jpg	2019/12/24 9:38	Image (jpg) File	4 KB	
330.jpg	2019/12/24 9:39	Image (jpg) File	3 KB	
384.jpg	2019/12/24 9:39	Image (jpg) File	4 KB	
55875.jpg	2014/11/7 22:29	Image (jpg) File	8 KB	
414276.jpg	2019/12/24 9:13	Image (jpg) File	3 KB	
510944.jpg	2019/12/24 9:12	Image (jpg) File	3 KB	
541201.jpg	2019/12/24 9:13	Image (jpg) File	3 KB	
648584.jpg	2019/12/24 9:14	Image (jpg) File	4 KB	
661134.jpg	2019/12/24 9:15	Image (jpg) File	3 KB	
938287.jpg	2019/12/24 9:15	Image (jpg) File	2 KB	
43682641_p0.jpg	2016/8/22 18:20	Image (jpg) File	7 KB	
47538259_p0.jpg	2019/12/24 9:13	Image (jpg) File	3 KB	
pokemon-logo.p...	2019/12/24 9:42	Image (png) File	8 KB	
Satoshi.jpg	2019/12/24 9:22	Image (jpg) File	3 KB	

原图大小为几百Kb到几Mb不等

名称	日期	类型	大小	标记
080.jpg	2019/11/27 10:24	Image (jpg) File	1,020 KB	
254_2.jpg	2019/11/27 10:25	Image (jpg) File	555 KB	
257_2.jpg	2019/11/27 10:26	Image (jpg) File	583 KB	
260_2.jpg	2019/11/27 10:26	Image (jpg) File	675 KB	
330.jpg	2019/11/27 10:27	Image (jpg) File	925 KB	
384.jpg	2019/11/27 10:27	Image (jpg) File	1,081 KB	
pokemon-logo.p...	2019/11/26 23:14	Image (png) File	625 KB	

名称	日期	类型	大小	标记
414276.jpg	2019/11/26 22:27	Image (jpg) File	480 KB	
510944.jpg	2019/12/23 16:35	Image (jpg) File	5,373 KB	
541201.jpg	2019/12/23 16:35	Image (jpg) File	899 KB	
648584.jpg	2019/11/26 22:36	Image (jpg) File	2,440 KB	
661134.jpg	2019/11/26 22:44	Image (jpg) File	3,115 KB	
938287.jpg	2019/11/26 22:44	Image (jpg) File	1,571 KB	
43682641_p0.jpg	2016/8/22 18:20	Image (jpg) File	1,027 KB	
47538259_p0.jpg	2016/8/22 18:18	Image (jpg) File	1,337 KB	
Satoshi.jpg	2019/11/26 22:25	Image (jpg) File	322 KB	

## 懒加载方式

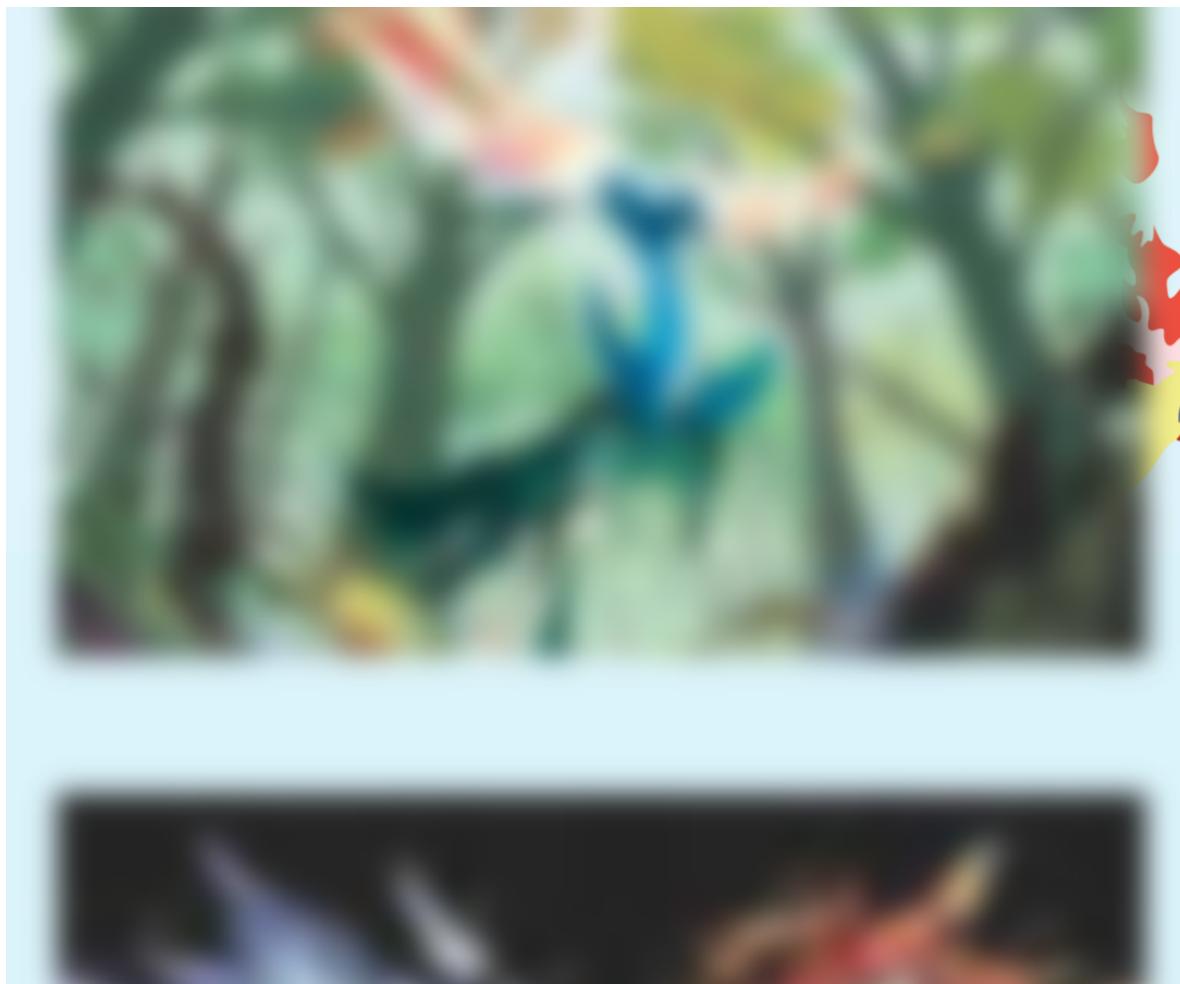
在/public/js/index.js中，使用API接口IntersectionObserver创建观察者

当观察到有图片进入视图100px以上时，就开始加载原图，将对应图片的data-src的值赋给它的src属性，赋值完后便停止观察该图片

```
// 图片进入到视野中才开始加载图片
function ObserverLazyLoadImages () {
    let images = document.querySelectorAll( selector: ".wallpaper");
    let observer = new IntersectionObserver( callback: entries => {
        entries.forEach(item => {
            if (item.isIntersecting) {
                item.target.src = item.target.dataset.src; // 开始加载图片,把data-src的值放到src
                //item.target.classList.remove('blur'); //移除高斯模糊class, 如果在这里去掉的话, 原图还没加载完就会去掉模糊
                observer.unobserve(item.target); // 停止监听已开始加载的图片
            }
        });
    },
    options: {
        rootMargin: "0px 0px -150px 0px" // 交叉过视图的150, 才开始派发事件
    }
);
images.forEach( callbackfn: item => observer.observe(item));
}
```

## 渐进式加载

考虑到低网速时的加载体验, 略缩图对于用户而言太模糊, 在原图还没加载完成前, 只加载略缩图对用户体验而言不够友好, 因此在略缩图上加上一个高斯模糊, 优化了观感



高斯模糊效果使用css完成, 以class的方式加入到每张图片中:

```

/*高斯模糊*/
.blur {
    -webkit-filter: blur(10px); /* Chrome, Opera */
    -moz-filter: blur(10px);
    -ms-filter: blur(10px);
    filter: blur(10px);
}

```

等原图加载完后，移除该图片的高斯模糊class

```

12 // 等原图加载完，再移除模糊滤镜
13 $(' .wallpaper ').on( events: 'load', handler: function () {
14     if ($this).attr( attributeName: 'src' ) === $this.attr( attributeName: 'data-src' ) {
15         $this[0].classList.remove( tokens: 'blur' ); // 移除高斯模糊Class，jQuery写法，当知道只会有一个结果返回时，$this[0]等价于this
16         // this.classList.remove('blur'); // 移除高斯模糊Class，DOM写法
17     }
18 });
19
20 // 懒加载图片
21 ObserverLazyLoadImages();

```

## 视频延迟加载

视频的加载等到页面加载完成后再加载，左侧浮动栏图片列表的原图也是再页面加载完后再加载

```

25 window.onload=loadImg; // 加载好页面再加载图片
26 window.onload=loadVideo; // 加载完页面再加载视频
27
28 // 加载左侧浮动栏的图片
29 function loadImg() {
30     $('.float-icon').each( function: function () {
31         const node = $this;
32         node.attr( attributeName: 'src', node.attr( attributeName: 'data-src'));
33     });
34 }
35
36 // 加载视频
37 function loadVideo(){
38     $('#pokemon-op source').attr( attributeName: 'src', $('#pokemon-op source').attr( attributeName: 'data-src')); // 设置src
39     $('#pokemon-op')[0].load(); // 重新加载视频
40 }

```

## 小功能：背景切换

点击页面中左侧浮动栏中的任意一个图片（除了最后一个以外），就会触发随机背景切换

```

// 点击这个类，就会更换背景图片
$('.click-div').on( events: 'click', handler: function () {
    changeBackGround();
});

```

切换背景时会先切换为要加载的**背景主颜色**，优化加载体验

```

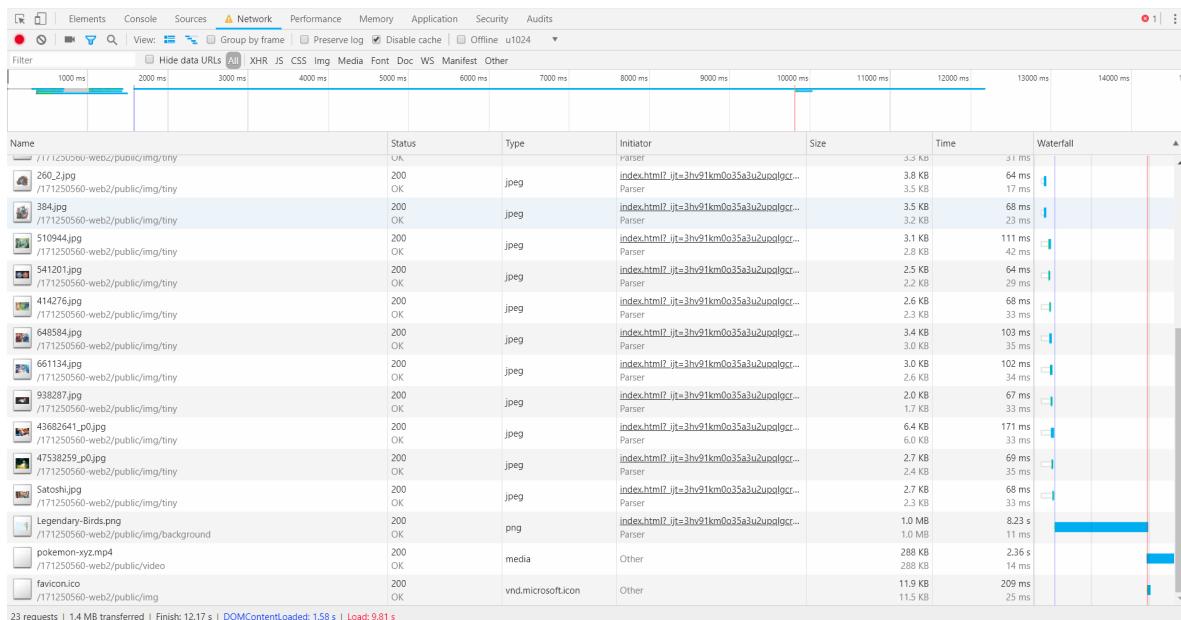
41
42 // 切换背景图，同时切换背景颜色
43 function changeBackGround() {
44     // 壁纸列表
45     const background_array = ["../img/background/Eon-Pokemon.png",
46                               "../img/background/Mythical-Pokemon.png",
47                               "../img/background/Swords-Of-Justice.png",
48                               "../img/background/Legendary-Birds.png"];
49     // 每张壁纸对应的背景颜色
50     const background_color = ['#bedc8', '#b0b6c2', '#5d8d86', '#DFF4FC'];
51     // const background_color = [0xbcdce8, 0xb0b6c2, 0x5d8d86, 0xDFF4FC];
52     const key = Math.floor( x: Math.random() * background_array.length );
53     $('body').css( propertyName: "background-color", value_function: "#" + background_color[key]); // 先改背景颜色
54     $('body').css( propertyName: "background-image", value_function: 'url('+ background_array[key] + ')'); // 再改背景图片
55
56 }

```

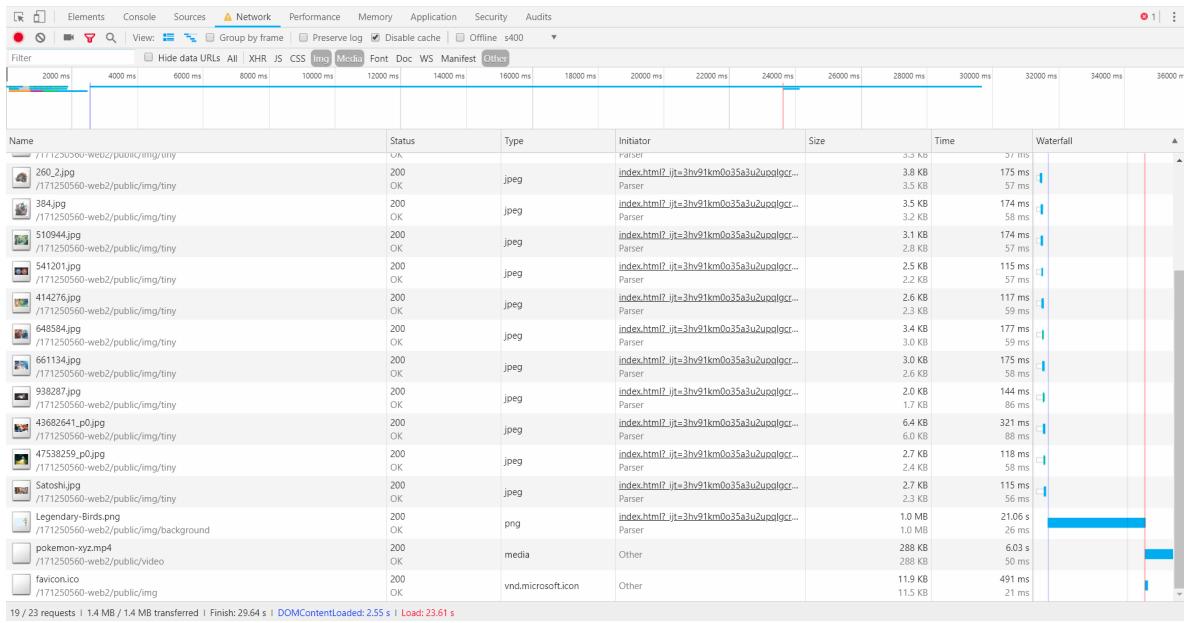
## Chrome开发者工具截图

DOM主体（除了背景图片和视频之外）大小在100kb以内

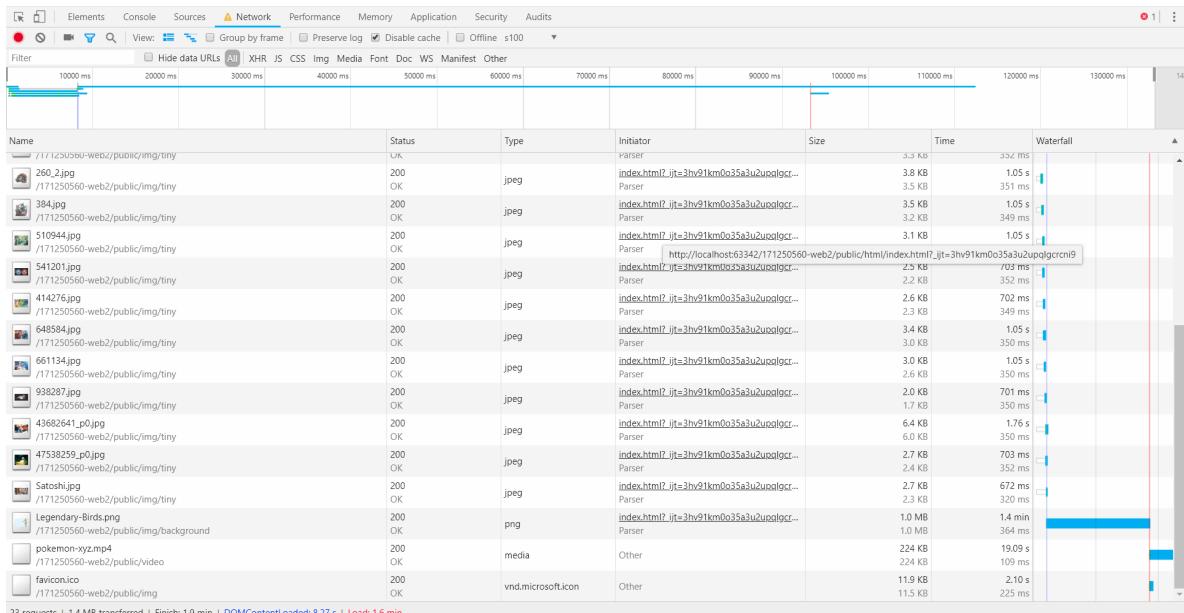
**下载速度：1024kb/s**



**下载速度：400kb/s**



下载速度: 100kb/s



加载原图时的网页体验:

