

摘要

Web2.0 是相对于 Web1.0 的新的时代。将 Web 2.0 放到科技发展与社会变革的大视野下来看，Web 2.0 是信息技术发展引发网络革命所带来的面向未来、以人为本的创新 2.0 模式在互联网领域的典型体现，是由专业人员织网到所有用户参与织网的创新民主化进程的生动注释。时至今日，网页依然占据互联网的大头，APP 内嵌、SPA 等等等等，网页无处不在，无所不能。

关键词：前端开发，JavaScript，IndexedDB，电子相册

目录

1 简介	2
1.1 WEB 端	2
1.2 关于前后端分离	2
1.3 服务端	3
2 项目目录结构介绍	4
3 前端代码分析	5
3.1 js/Elements.js	5
3.2 js/PageManager.js	6
3.3 js/HomePageManager.js	7
3.4 js/FavoritePageManager.js	7
3.5 js/ImageBoxTemplate.js	7
3.6 js/ImageBoxFunctions.js	7
3.7 js/login_regist_public.js	7
3.8 modules/Authenticator.js	7
3.9 modules/UUID.js	8
3.10 modules/parallax.js	8
3.11 module/ VerificationCode.js	9
3.12 module/ThemeManager.js	9
3.13 module/UserManager.js	9
3.14 module/ImageManager.js	9
3.15 data/images.json	10
3.16 assets/fonts/*	10
3.17 image/preview/*	10
3.18 Image/origin/*	10

1 简介

使用原生 WEB 技术实现一个网页电子相册

前端: JavaScript

数据库: 业务逻辑不复杂, 采用浏览器的 IndexedDB 建立图片数据索引、用户账号密码。

1.1 WEB 端

每个页面都有很多张图片, 对于呈现在可视范围内的图片我们可以通过放置一张降低画质的预览图来代替此举不仅降低了服务端的带宽开销也非常有效的缓解了浏览器显示大量高清图片卡顿的问题, 对于那些不显示在可视范围内的图片我们可以使用懒加载技术延迟加载。电子相册的前端 UI 设计采用新拟态风格, 采用响应式数据的方式来监听页码变化以请求新的数据。

1.2 关于前后端分离

web1.0 时代, 网页仅仅是向人们展示内容, 作为用户我们也只能浏览。此时后端代码与前端代码混写没有太大的代价。而到了 web2.0 时代, 页面的交互变得逐渐复杂, 此时后端代码与前端代码混写会使得后期的维护变的非常困难。

例如在传统的 PHP WEB 开发过程中, 一个网页不是由后端开发者来独立完成的:
前端开发 -> HTML 静态页面 -> 后端开发 -> PHP

前端会把页面做出来, 我们后端需要开发, 就把前端页面嵌入到 PHP 中, 也需要添加标签才能把数据整合起来。因为核心就是: 如何把我们后端返回的数据添加到页面中, 无论是 PHP、ASP、JSP 模板都一样。

如果此时网页遇到一些问题, 我们把 PHP 发给前端开发, 前端开发人员看不懂 PHP。此时前端也不好解决, 后端也不好解决。这样沟通和开发效率非常低, 前后端的代码耦合度太高, 无论是开发还是维护起来都非常麻烦。

web2.0 时代的前端已经演化成了“大前端”, 各类框架的出现进一步推动了前后端分离开发的主流趋势。伴随着 Angular, React, Vue 等数据驱动视图的框架出现和 webpack、node 的出现彻底解决了 web1.0 时代前后端代码高度耦合的问题。

现在的前端只需要独立编写客户端代码, 后端也只需要独立编写服务端代码提供数据接口即可。前端通过 AJAX、Fetch 请求来访问后端的数据接口, 然后将 Model 展示到 View 中即可。

这就是我们说的前后端分离开发，而代价则是更多的请求数量。

1.3 服务端

由于本项目使用的是纯前端处理所以在后端方面不会使用任何语言。所有的数据都存在浏览器的 IndexedDB 内部。我们通过抽象代码和分模块来实现一个类似前后端分离的架构。把所有的请求操作、所有的图片操作封装为一个个的类，以此来划分好业务代码和数据逻辑代码。

为了处理高并发的图片请求、IndexedDB 数据库请求，我们要尽可能的使用异步编程的方式来处理这些问题。最重要的一点就是代码风格需标准且尽量模块化。由于 Chrome V8 进程只有一个主线程在执行程序代码，所以 JavaScript 是单线程的。但得益于 JavaScript 的异步编程方法，使得 JavaScript 在服务端、客户端处理高并发请求时比传统的语言快很多。

单线程的优势：

1. 多线程占用的内存更多
2. 多线程间的切换使得 CPU 开销大
3. 多线程由内存同步开销
4. 编写单线程程序简单
5. 线程安全问题

单线程的劣势：

1. CPU 密集型任务占用 CPU 时间长
2. 无法利用 CPU 的多核
3. 单线程抛出异常使得程序停止（可通过 try catch）

2 项目目录结构介绍



3 前端代码分析

3.1 js/Elements.js

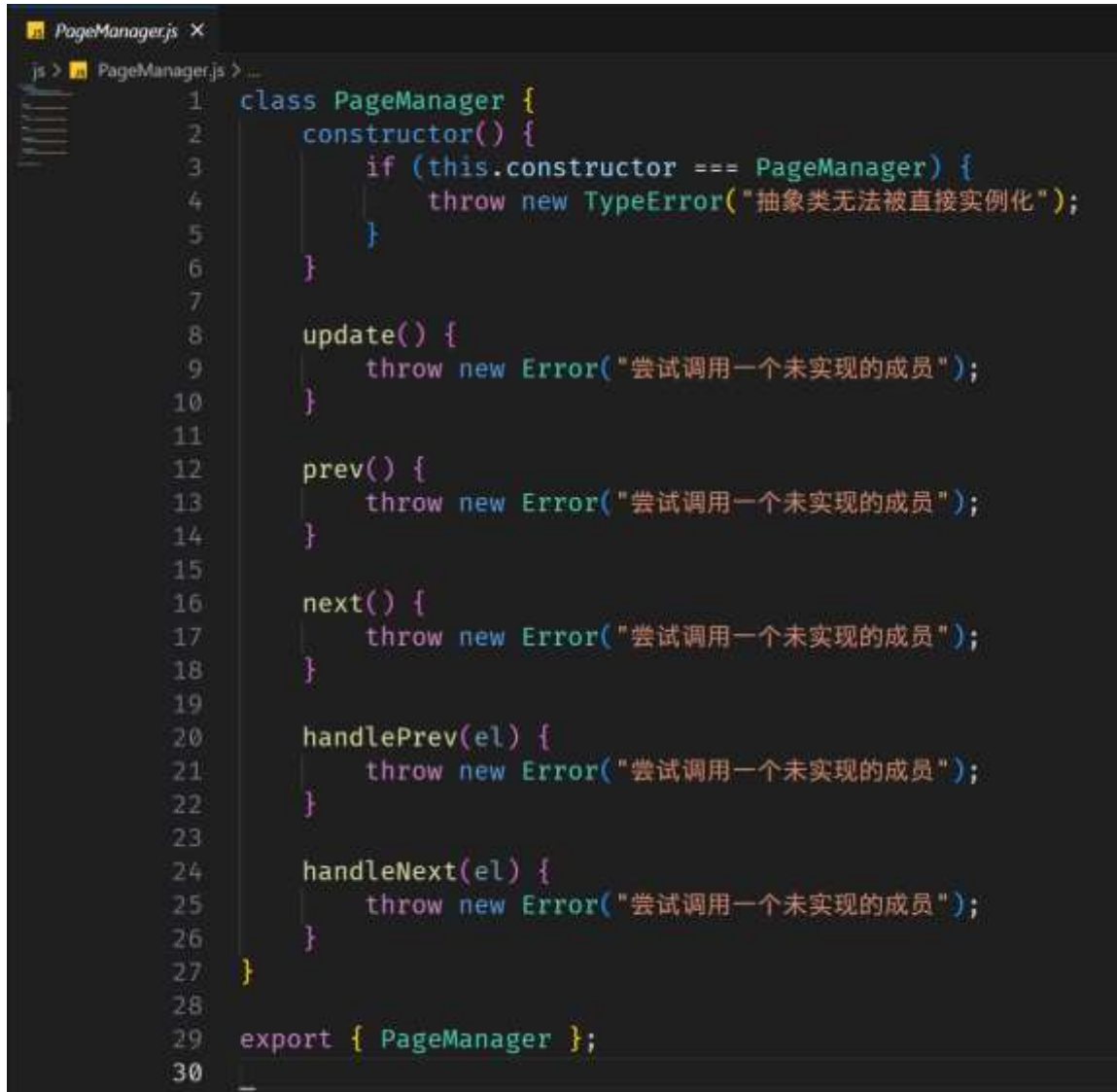
Elements.js 导出一个对象，对象内包含了一些高频操作的 DOM 元素。以下为对象属性介绍：



```
1 export default {
2   HomeButton: document.querySelector("nav ul.left li.home"), // 首页按钮
3   FavoriteButton: document.querySelector("nav ul.left li.favorite"), // 收藏页按钮
4   LogoutButton: document.querySelector("nav ul.left li.logout"), // 注销按钮
5   PageNumber: document.querySelector("nav li.page"), // 页码
6   PrevPageButton: document.querySelector("nav ul.right li.prev"), // 上一页
7   NextPageButton: document.querySelector("nav ul.right li.next"), // 下一页
8
9   ThemeSwitchButton: document.querySelector("#switch-theme"), // 主题切换
10  ImageListBox: document.querySelector("main"), // 图片展示框的根元素
11  PreviewImage: document.querySelector("#preview-image img"), // 预览图元素
12  PreviewImageBox: document.querySelector("#preview-image"), // 预览图元素 (img)
13 };
14
```

3.2 js/PageManager.js

PageManager.js 是一个抽象类 (js 不支持抽象类, 这里的抽象类是我通过代码模拟的) 其中, update 负责刷新页面数据 (图片), prev 代表上一页, next 则会获取下一页数据, handlePrev 接受一个 dom 元素, 并在 dom 元素被点击时调用 prev 函数。handleNext 亦是如此。update 负责根据当前的页码更新并生成 html。



```
1 class PageManager {
2   constructor() {
3     if (this.constructor === PageManager) {
4       throw new TypeError("抽象类无法被直接实例化");
5     }
6   }
7
8   update() {
9     throw new Error("尝试调用一个未实现的成员");
10  }
11
12  prev() {
13    throw new Error("尝试调用一个未实现的成员");
14  }
15
16  next() {
17    throw new Error("尝试调用一个未实现的成员");
18  }
19
20  handlePrev(el) {
21    throw new Error("尝试调用一个未实现的成员");
22  }
23
24  handleNext(el) {
25    throw new Error("尝试调用一个未实现的成员");
26  }
27 }
28
29 export { PageManager };
30
```

3.3 js/HomePageManager.js

PageManager.js 的派生类。

3.4 js/FavoritePageManager.js

PageManager.js 的派生类。

3.5 js/ImageBoxTemplate.js

此函数接受一个 { id: string, filename: string, filetype: string, tag: string } 类型的对象, 并依据传入的对象生成装有图片、图片控件的盒子。在 html 字符串中还绑定了 ToggleFavorite (收藏、取消收藏)、ViewImage (查看大图)、DownloadImage (下载图片)、ShareImage (图片直链) 等函数到对应元素的点击事件上。

3.6 js/ImageBoxFunctions.js

js/ImageBoxTemplate.js 中生成的 html 的所有点击事件所对应的函数, 都写在这个文件内。如 ToggleFavorite (收藏、取消收藏)、ViewImage (查看大图)、DownloadImage (下载图片)、ShareImage (图片直链) 等函数都写在这里。

3.7 js/login_regist_public.js

由于登录页面和注册页面具有高度一致的 js 代码, 所有相同的部分被抽出来放在了这里。这其中包含了一些功能: 鼠标移动时的视差效果、背景图片动态切换效果。

3.8 modules/Authenticator.js

身份验证模块, 主要负责验证是否登录 (isLogin)、用户登录 (login)、用户登录状态注销 (logout) 功能。

3.9 modules/UUID.js

内部包含一个名为 GenerateUUID 的函数，此函数用于生成一个唯一的 UUID（种子是当前的时间戳，所以理论上并发小于 10w 应该不会出现重复），参考官方文档和 AI 实现。

3.10 modules/parallax.js

视差效果的核心库。手工实现。内部包含的变量：

ArrayList: ((dx: int, dy: int) => void)[]

clientX: 鼠标 x 坐标

clientY: 鼠标 y 坐标

实现原理：

1. 通过监听 document 的 mousemove 事件来实现鼠标位置变化检测，在事件监听器内部获取鼠标的 x y 并把对应的值保存在 clientX、clientY 里。
2. cvo 函数用于将鼠标位置坐标归一化（鼠标 [x|y] 位置 / 可视区域 [宽度|高度] 即可得到一个 [0, 1] 之间的小数，对这个区间进一步扩大 3 倍就可以让视差强度得到提升了）
3. 调用浏览器为我们提供的运行在渲染线程中并且跟随显示器刷新率垂直同步的内置函数 requestAnimationFrame 并把 randerLoop 函数作为参数传入，让 randerLoop 跟随显示器刷新率每秒执行 30/60/90/120/144/160/180/200/240 次。
4. randerLoop 内部循环遍历 ArrayList 内部的函数，依次将 dx, dy（归一化的 x y）传入并调用它。
5. 这样就封装成了一个视差效果的渲染器，我们只需要往 ArrayList 内 push 一些函数，在这些函数内利用闭包的特性就可以任意使用 parallax.js 这个我们封装好的库了。

3.11 module/ VerificationCode.js

验证码模块。此部分是先前课堂上验证码改造而来。封装成了一个类。

主要做了一些改动：

1. 小图片全部转换为 base64 字符串保存
2. 单例模式

其中包含了 GetInstance 静态方法用于创建一个单例模式的对象。

Generate 接受宽高和长度返回 html 格式的验证码，这里的宽高是指单个 img 的宽高，所以实际宽是 传入的宽度 * 传入的长度；实际的高和传入的高度一致。

verify 函数接受一个字符串，验证这个字符串和上一个生成的验证码结果是否一致。

3.12 module/ThemeManager.js

主题管理器，内部通过引用 js/Element.js 中的主题切换按钮，来实现用户点击按钮时切换主题。切换主题的核心原理是在编写 css 是所有涉及到颜色的部分均使用 var(--xxx) 这种形式的 css 变量标识。light.css 和 dark.css 中只写那些 css 变量的颜色取值。这样我们只需要动态切换引入的 light.css 和 dark.css 即可实现深色与浅色模式之间的切换。

3.13 module/UserManager.js

基于 IndexedDB 的用户管理器。支持以单例对象的方式创建。也支持直接构造（需要传入 IDBStore 对象）。此模块支持用户创建（addUser）、删除用户（removeUser）、更新用户（updateUser）、查询用户信息（getUser）、获取用户计数（getUserCount）功能。

3.14 module/ImageManager.js

基于 IndexedDB 的图片管理器。支持以单例对象的方式创建。也支持直接构造（需要传入 IDBStore 对象）。创建实例时，须传入用户名，这样才会生成仅针对这个用户的数据库。使得用户之间的数据保持独立性。此模块支持添加图片信息（addImage）、删除图片信息（removeImage）、更新图片信息（updateImage）、获取图片信息（getImage）、设置图片标签（setImageTag）、获取图片列表（可分页查询）（getImageList）、根据标签

获取图片列表（可分页查询）（getImageListByTag）、获取图片计数（getImageCount）功能。

3.15 data/images.json

默认图片信息，新用户登录后，会同 Ajax 请求此文件，并依据此文件内的数据创建默认图片库。

3.16 assets/fonts/*

iconfont 图标库，页面中的一些图标均由此库提供。

3.17 image/preview/*

预览图目录，此处储存低清晰度的图片。

3.18 Image/origin/*

原图目录，此处储存原始清晰度的图片提供下载、高清预览。

参考文献

1. MDN Web Docs [<https://developer.mozilla.org/zh-CN/>]