به نام خدا

عنوان

راهنمای پروژه front-end میکروکنترولر توسعه فناوری کشاورزی دقیق

مسئول

مهندس روزبه بابازاده

تنظیم کننده

حجت عزیزی

آدرس سورس پروژه

https://github.com/hjtazzi/patd

تاریخ

1401/6/20

فهرست

توضیحات اولیه پروژه

برای پیکربندی تنظیمات میکروکنترولر موجود، نمایش اطلاعات و وضعیت آن و دیگر امکانات، نیازمند ارائه یک صفحه وب پویا به کاربر متصل به دستگاه هستیم تا کاربر بتواند به خوبی با دستگاه در تعامل باشد. این صفحه وب در زمان اتصال کاربر به Wi-Fi موجود در دستگاه و بارگذاری آدرس مشخص شده در مرورگر قابل دسترسی است.

مشکلات پروژه

اولین مشکل در فرایند توسعه، مشکل کمبود فضای ذخیره سازی و حافظه موقت است. کل فضای ذخیره سازی موجود در این دستگاه حدود 4MB است که پس از تقسیم این فضا برای عملیات‌های مختلف، حدود 200KB تا 300KB فضای ذخیره سازی به بخش Front-end اختصاص داده میشود.

دومین مشکل، عدم دسترسی کاربران متصل به دستگاه، به اینترنت است. احتمال این که کاربران در هنگام استفاده از دستگاه در شرایط مختلف به اینترنت دسترسی نداشته باشند بسیار بالا است. در نتیجه نمیتوان از پکیج‌های موجود در اینترنت (مثل bootstrap و jQuery) به صورت CDN یا غیره که نیاز به اتصال اینترنت دارد استفاده کرد.

در فرایند تولید و توسعه نیازمند یک ساختار واضح و گویا برای اجزای مختلف پروژه داریم تا در این فرایندها بتوان بهتر و سریعتر عمل کرد و در زمان صرفه جویی شود. اما به دلیل دو موردی که به آن اشاره کردیم قادر به استفاده از چهارچوب‌هایی مانند ReactJS که یک ساختار کامپوننت محور را به توسعه دهنده ارائه می‌دهند، نیستیم.

راه حل‌های پیشنهادی

در اسناد HTML:

اولین مورد ایجاد یک سند HTML کامل است که در آن همه اجزای مورد نیاز نوشته شده و با استفاده از JavaScript و پکیج JQuery اجزای سند کنترل می‌شود. اما در این راه حل ساختاری وجود ندارد و حجم کدها در سند بسیار بزرگ و تغییر آن‌ها در آینده بسیار سخت می‌شود.

مورد بعدی ایجاد یک سند اصلی index.html است که موارد اصلی و ساختمان پروژه در آن ایجاد شده و در کنار این سند، اجزای دیگر هر کدام در سندهای جدا نوشته می‌شوند و با استفاده از JavaScript در صورت نیاز کاربر هر کدام از سرور درخواست شده و به کاربر ارائه می‌شود. در این مورد ساختاری برای اجزا ایجاد شده اما تعداد فایل‌های اسناد بالا است و امکان ایجاد خطا در درخواست‌ها وجود دارد همچنین کنترل المان‌های اجزا دشوار می‌شود.

در استایل دهی صفحات:

در استایل دهی صفحات از راه‌های متعددی می‌توان استفاده کرد. مانند استفاده از کتابخانه‌هایی مثل Bootstarp، TailwindCSS و... و همچنین استفاده از پیش پردازنده‌های CSS مانند SASS و LESS. در استفاده از کتابخانه‌ها و زبان‌های استایل دهی تنها به حجم خروجی و صرفه جویی در زمان توجه داریم.

در JavaScript:

از JavaScript برای توسعه و برنامه‌نویسی صفحات و از پکیج‌های آن برای توسعه سریع‌تر استفاده می‌کنیم. در این مورد هم نیاز به ساختاری داریم که سبک و قابل فهم برای توسعه در آینده باشد. می‌توان کدهای JS و پکیج‌ها را در فایل‌های مختلف قرار داد و به آسانی به سند اصلی لینک کنیم اما این کار احتمال ایجاد خطا را افزایش می‌دهد. همچنین ما را از ایجاد تعداد زیاد فایل JS برای ماژولار کردن پروژه، باز میدارد. با توجه به این موارد به استفاده از Bundler نیاز داریم.

بررسی راه‌حل و ابزارهای استفاده شده

ما در این پروژه به دنبال یکپارچه سازی، حجم کم فایل‌های خروجی و همچنین تعداد هرچه کمتر فایل‌های خروجی هستیم. همچنین به دنبال ایجاد یک ساختار و روند بهینه برای توسعه بهتر هستیم.

ابتدا به معرفی ابزارها و وابستگی‌های پروژه می‌پردازیم.

در استایل دهی صفحات از SASS که یک پیش پردازنده برای CSS (CSS-Preprocessor) است استفاده کرده‌ایم. پیش پردازنده‌ها با هدف صرفه جویی در وقت و میزان کار توسعه دهنده، قابلیت‌هایی را به فایل‌های CSS می‌افزاید. برای مثال می‌توانند با افزودن متغیرها، اپراتورها، توابع، mixins و... عملکرد CSS را ارتقا دهد. همچنین به داشتن ساختار ماژولار و استفاده مجدد از کدهای نوشته شده کمک می‌کند.

```
"dependencies": {
  "jquery": "^3.6.0",
  "query-string": "^7.1.1",
  "timestamp-to-date": "^1.1.0"
},
"devDependencies": {
  "esbuild": "0.15.5"
}
```

تصویر بالا قسمتی از فایل package.json است که در آن وابستگی‌های پروژه و وابستگی‌های توسعه را نمایش می‌دهد. فایل‌های وابستگی‌ها در پوشه node_modules قرار دارد که در صورت موجود نبودن این پوشه، با اجرای دستور npm i در ترمینال در دایرکتوری اصلی پروژه قابل نصب هستند.

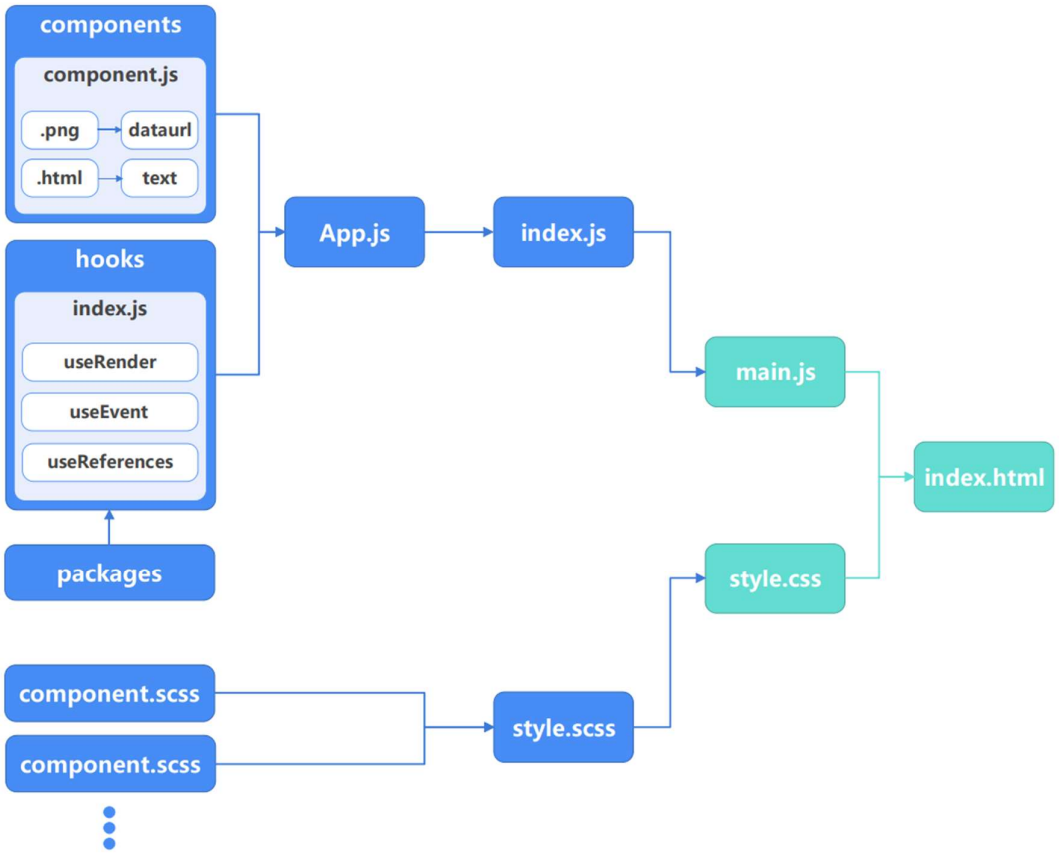
در وابستگی‌های توسعه (devDependencies)، پکیجی به نام "esbuild" با نسخه "0.15.5" وجود دارد. همان طور که گفته شد برای ادغام اجزا و ماژول‌ها نیاز به یک Bundler داریم. esbuild برای ما کار bundle ماژول‌ها را با سرعت بسیار بالا و حجم پایین‌تر نسبت به دیگر پکیج‌ها انجام می‌دهد. همچنین از قابلیت‌های دیگر این پکیج مثل، کنترل نسخه هدف جاوا اسکریپت، minify کردن فایل خروجی و ایجاد فایل سورس مپ، استفاده از loaderهای موجود آن (برای مثال برای تبدیل فایل html به text و ذخیره آن در یک متغیر استرینگ js) و... استفاده می‌شود.

در وابستگی‌های پروژه (dependencies)، سه پکیج jquery، query-string و timestamp-to-date وجود دارد. کتابخانه اصلی استفاده شده در پروژه jQuery است که بار کنترل و نمایش اجزا در سند، کنترل ایونت‌ها، کنترل المان‌های صفحه، کنترل درخواست‌ها و... بر دوش این پکیج است.

دو پکیج دیگر بسیار سبک هستند و فقط در برخی موارد مانند تجزیه رشته‌ها و تبدیل فرمت تاریخ استفاده شده‌اند.

در ادامه به بررسی اجزای اساسی پروژه می‌پردازیم. به طور کلی اجزای مختلف پروژه را می‌توان با تصویر زیر نشان داد.

همان طور که در تصویر مشخص شده، در استایل دهی صفحات بخش‌های مختلف در اجزای جداگانه در فایل‌هایی با پسوند scss. نوشته می‌شوند. همه این موارد در فایل style.scss وارد شده و پس از کامپایل به فایل style.css تبدیل می‌شود که قابل استفاده در صفحات وب است.



همان طور که گفته شد همه پکیج‌های استفاده شده در پوشه node_modules در دایرکتوری اصلی پروژه موجود است. همچنین سورس اصلی پروژه در پوشه src موجود می‌باشد. به طور کلی سورس اصلی و فایل‌های JS را میتوان به سه بخش packages، hooks و components تقسیم بندی کرد که هر کدام دارای بخش‌های مجزا هستند.

بخش packages همان طور که از اسم آن پیداست شامل تمامی پکیج‌های پروژه می‌باشد. برای به حداقل رساندن وابستگی بین packages و components بخش hooks نوشته شده تا اجزای موجود در هر پیکیج قابل دسترس‌تر باشند همچنین در زمان ایجاد تغییرات بتوان بهتر عمل کرد. در نتیجه هوک‌ها مسئول ایجاد ارتباط بین پکیج‌ها و اجزا می‌باشد.

بخش components، اجزای اصلی صفحات می‌باشند که حتما شامل یک فایل JS است و می‌تواند شامل فایل‌های html. و png. و... هم باشد. در آخر همه اجزا به App.js وارد میشود و از این فایل یک تابع به نام App وارد فایل index.js می‌شود که با استفاده از تابع ready کتابخانه جی‌کوئری پس از ایجاد سند شروع به ایجاد اجزا در سند می‌کند.

پس از اجرای اسکریپت build، همه اجزا و فایل‌های موجود در پوشه src و به واسطه index.js و با استفاده از پکیج esbuild، به یک فایل مستقل به نام main.js در پوشه build تبدیل می‌شود که قابل استفاده در مرورگرها است.

در آخر هر دو فایل خروجی style.css و main.js به سند اصلی پروژه یعنی index.html لینک شده و قابل استفاده می‌باشند.

راهنمای پروژه

در این بخش به بررسی جزئیات اجزای موجود در پروژه می‌پردازیم.

ابتدا نگاهی به اسکریپت‌های موجود در package.json می‌اندازیم:

```
"scripts": {
   "watch":
       "esbuild ./src/index.js --bundle --outfile=dist/main.js
       --target=es6,chrome58,edge18,firefox54,safari11,opera55
       --loader:.html=text --loader:.png=dataurl --watch",
   "build":
       "esbuild ./src/index.js --bundle --minify --outfile=build/main.js
       --target=es6,chrome58,edge18,firefox54,safari11,opera55
       --loader:.html=text --loader:.png=dataurl --sourcemap"
 },
```

دو اسکریپت watch و build تعریف شده‌اند که در بیشتر موارد مشترک هستند.

watch: از پکیج esbuild استفاده می‌کند و فایل index.js که در پوشه src قرار دارد را اجرا کرده و همه اجزای مورد نیاز این فایل را در کنار هم قرار داده و در پوشه dist و فایل main.js را ایجاد می‌کند (خط اول). نسخه هدف جاوا اسکریپت و همچنین نسخه مرورگرها که از این نسخه جاوا اسکریپت پشتیبانی میکنند را انتخاب می‌کند (خط دوم). فایل‌ها با پسوند html. که به فایل‌های js وارد شده‌اند را به عنوان text و فایل‌هایی با پسوند png. را هم به عنوان dataurl در نظر می‌گیرد که می‌توان در سورس از آن‌ها به عنوان یک رشته string استفاده کرد. بعد از هر تغییر و ذخیره فایل‌ها، به صورت خودکار همه عملیات دوباره انجام می‌شود و میتوان تغییرات را مشاهده کرد.

build: از آن برای خروجی نهایی استفاده می‌شود که تنها چند تفاوت با اسکریپت watch دارد: فایل خروجی را در پوشه build قرار می‌دهد همچنین آن را به صورت فشرده و با حجم کمتر ایجاد می‌کند (خط اول). فقط در هر بار اجرای این دستور

عملیات‌ها انجام می‌شود و همیچنین در کنار فایل خروجی، فایل سورس مپ به نام main.js.map را ایجاد می‌کند (خط سوم).

قبل از بررسی کامپوننت‌ها، به بررسی کلی hookها می‌پزدازیم. هوک‌ها مسئول ارتباط با پکیج‌ها و دارای توابع پرکاربرد در پروژه می‌باشد. سه فایل useRender.js، useEvent.js و useReferences.js، هر کدام به اصطلاح هوک، یک آبجکت جاوا اسکریپت را صادر می‌کنند که هر کدام دارای توابعی می‌باشند که از آن‌ها در کامپوننت‌ها استفاده می‌شود.

useRender: دارای توابعی می‌باشد که با استفاده از توابع jQuery اجزای html را در سند اصلی صفحه ایجاد و یا حذف می‌کند. برای مثال تابع html دو پارامتر selector (المانی که می‌خواهیم آن را تغییر دهیم) و content (المانی که میخواهیم آن را ایجاد کنیم) را دریافت می‌کند و content را در المان selector ایجاد می‌کند.

useEvent: با استفاده از توابع jQuery، مسئول کنترل ایونت‌های المان‌های صفحه است که هر تابع دو پارامتر selector (المانی که می‌خواهیم ایونت‌های آن را کنترل کنیم) و action (تابعی است که پس از رخ دادن ایونت اجرا می‌شود) را دریافت می‌کند. یک تابع به نام on نیز وجود دارد که یک پارامتر اضافه به نام event را دریافت می‌کند که با استفاده از آن می‌توان رویدادهای بیشتری را کنترل کرد.

useReferences: دارای پروپرتی‌هایی برای دریافت، ایجاد و تغییر در اتربیوت‌های المان‌های صفحه، مقادیر css آن‌ها، دریافت المان‌های دیگر، اعمال افکت‌ها، کنترل فرم‌ها و کنترل درخواست‌ها می‌باشد. دو خاصیت forms و ajax که دو خاصیت مهم این هوک هستند را در ادامه بررسی می‌کنیم.

forms: برای دریافت مقادیر ورودی کاربر در فرم‌ها و تبدیل آن‌ها به نوع‌های مورد نظر استفاده می‌شود. برای مثال تابع serialize با استفاده از توابع کتابخانه jQuery، با دریافت پارامتر selector همه مقادیر وارد شده در اینپوت‌های موجود در فرم مورد نظر را به صورت یک رشته (query string) باز می‌گرداند. تابع qsParse با استفاده از کتابخانه query-string و با دریافت پارامتر value که یک رشته است، رشته query string دریافتی را به

صورت یک آبجکت جاوا اسکریپت باز می‌گرداند. تابع blockSpecial با دریافت پارامتر selector که یک اینپوت است، مقادیر ورودی کاربر را کنترل می‌کند که تنها مقادیر مجاز کاراکترهای a-z و A-Z و 0-9 و نقطه می‌باشد.

ajax: دارای دو تابع post و get می‌باشد که با استفاده از تابع ajax کتابخانه jQuery درخواست‌های کاربر را کنترل می‌کند. پارامترهای ورودی آن به صورت زیر است:

url: آدرس درخواست به صورت رشته که قبل از آن مقدار ثابت baseURL که در ابتدای هوک تعریف شده اضافه می‌شود.

data: فقط تابع post این پارامتر را دریافت می‌کند که شامل داده‌های مورد نظر برای ارسال به سرور به صورت رشته و با فرمت urlencoded (query string) می‌باشد.

success: تابعی است که پس از موفق بودن درخواست اجرا می‌شود و مقادیر ارسال شده از سمت سرور (result) باز می‌گرداند.

error: تابعی است که پس از شکست درخواست اجرا می‌شود و دو مقدار status و statusText را باز می‌گرداند.

complete: تابعی است که پس از شکست یا موفقیت درخواست اجرا می‌شود و اطلاعات درخواست (xhr) را باز می‌گرداند.

timeout: مقدار عددی اختیاری برای زمان تلاش درخواست به میلی ثانیه و مقدار پیشفرض آن 10000 میلی ثانیه است.

در ادامه به بررسی دیگر اجزای پروژه می‌پردازیم:

همان طور که گفته شد اولین فایل اجرایی index.js است. در این فایل با استفاده از تابع ready که متعلق به کتابخانه jQuery است بعد از اجرای سند اصلی html، تابع App که صادر شده از فایل App.js است را اجرا می‌کند. تابع App، کامپوننت Login که خود نیز یک تابع می‌باشد را اجرا می‌کند.

کامپوننت Login ابتدا اجزای فایل login.html را در div اصلی سند با آی‌دی root ایجاد می‌کند که شامل یک فرم برای ورود می‌باشد که پس از ارسال فرم به سمت سرور و موفقیت در ورود، ابتدا #root را خالی کرده و به ترتیب دو کامپوننت Header و Main را اجرا می‌کند.

کامپوننت Header اجزای موجود را به #root اضافه می‌کند و اتربیوت src المان #header-logo را برابر با dataurl تصویر لوگو که در پوشه imgs دایرکتوری اصلی پروژه قرار دارد، می‌سازد.

کامپوننت Main ابتدا اجزای main.html را به #root اضافه می‌کند سپس تابع sidebar را از sidebar/sidebar.js اجرا می‌کند.

تابع sidebar اجزای aside.html را به المان #collapseNavbarAside موجود در main را اضافه می‌کند سپس یک درخواست get به سرور برای دریافت ساعت و تاریخ به آدرس get-time/. ارسال کرده و پس از موفقیت درخواست، با استفاده از کتابخانه timestamp-to-date با فرمت خوانا در #aside-footer-time ایجاد می‌کند. همچنین هر ثانیه مقدار آن را افزایش می‌دهد. این تابع با استفاده از هوک useEvent، دارای توابعی برای کنترل نمایش یا عدم نمایش ساید بار است. همچنین با تغییر اندازه افقی صفحه نمایش سایدبار را کنترل می‌کند. در آخر دو تابع NavMenu و itemsEvent را اجرا می‌کند.

تابع NavMenu، ابتدا یک ul در المان nav-menu. ایجاد می‌کند سپس با استفاده از اعضای آرایه menuItems که از فایل contents/menuItems.js/.. صادر شده، اجزای منو سایدبار را در #list-menu ایجاد می‌کند.

تابع itemsEvent، ابتدا تابع Content که صادر شده از فایل Content/Content.js/.. است را اجرا می‌کند. این تابع مسئول ایجاد المان‌ها در #main-content است و یک پارامتر ورودی به نام content دریافت می‌کند و با استفاده از شرط موجود در آن، توابع مورد نظر را اجرا می‌کند. در ادامه تابع itemsEvent، رویداد کلیک اعضای منو را کنترل می‌کند و با توجه به این رویداد و اطلاعات موجود در اتربیوت المان کلیک شده، دوباره تابع Content را با مقادیر ورودی جدید اجرا می‌کند.

تابع Content دارای یک switch است که caseهای آن با توجه به اعضای آرایه menuItems، توابع کامپوننت‌ها را اجرا می‌کند. این کامپوننت‌ها در مسیر src/components/main/contents/components در دایرکتوری اصلی پروژه قرار دارند که توابع موجود در آن‌ها به فایل index.js در همان مسیر وارد شده و به دلیل کاهش وابستگی بین اجزا، همه به صورت یک آبجکت از این فایل صادر می‌شوند.