Admin Portal

https://internal.switch.site

- dev: https://internal-dev.switch.site /
- test: https://internal-test.switch.site
- staging: https://internal-staging.switch.site /
- prod: https://internal.switch.site /

Design

figma

Introduction

This proejct follow this Web Architecture document.

Project Structure

`/aws-lambda`:

- Demo of aws lambda, which is not the actual config file of this project.

`/docs`:

- Original README.md or any others documents you want to write.

`/public`:

- Static resource, such as index.html, manifest.json, logo.png, robots.txt...

`/scripts`:

- Scripts which are defined to do something in project or CI/CD. You can put scripts in this folder and define command line in package.json. There's only react-intl scripts for lokalise currently.

`/src`:

- `/components`:
  - Common components in this project;
  - Should move all components, those can be used across projects, to switch-web-packages. Leave other project custom components still be here. (In Progress)
- `/constants`:
  - Some project common constants, such as product type, navigator, lang type...
- `/fonts`:
  - Project custom fonts, static files & css, auto generated by google-webfonts-helper.
- `/hooks`:

    - Project custom react hooks. If you can't find the react hook from react or react-use, create it in this folder.
- `/images`:
    - Images collected from figma;
    - Normally, prefer to use svg because of generality and resolution, that we can change size by css styles, and set sample svg image's color to follow component's color by set svg `fill` property with `currentColor` value;
    - If svg file is too large, like over than 100kb, consider to separate it to several parts, or just feel free to use compressed png or jpg images instead when design approved. Please keep in mind that we should keep project images' size as small as we can.
- `/langs`:
    - Project global intl files.
- `/pages`:
    - Route pages.
    - Usually grouped by side bar's routes or some certain functional scope.
- `/service`:
    - `/request`
        - Abstract function for request service in project
    - `/typings`
        - Type definition for service in project that can be used directly without import. With the convenience to use directly without import in current project, please care about the uniqueness of name declaration.
        - If you care about the risk of repeated name declaration, just use `export/import`;
    - `/v1`
        - Service functions to request.
        - If version, currently is `v1`, is changed to `v2` or others, please create corresponding folder to put new service functions.
- `/store`:
    - Some global store context, such as auth, regions and so on...
    - Prefer to use package unstated-next to create store. If you want to introduce some other management tool package, please feel free to create Merge Request and ask others to discuss.
- `/styles`:
    - Some common styles, such as colors, icons, font styles...
    - Also can move them to common package switch-web-packages folder.
- `/test-utils`: To be done
- `/typings`:
    - Project global type definition, if you want to add some type definition to global variable like Window, you can put them here.
- `/utils`:
    - Common utils, such as date formatter functions, operation system test and regexp...
- `routes.ts`:
    - App routes config
- `App.tsx`:
    - App root file.
    - Put your global provider and route config here.
- `index.tsx`:

    - Project root file.
    - Usually insert your app to static html.
- `react-app-env.d.ts`:
    - Project env type definition

`.gitlab-ci.yml`:

- CI/CD config

Common Packages

- switch-web-packages
- demo

Create New Project

If you want to create new SPA in Reinvent, use CRA & CRAO with the latest version of react, react-dom and react-router if you can. After that follow the *Web Architecture* to implment the development / deployment process.

Contribution

If you find any feature, bug or you just upgrade some version of dependencies. Feel free to create Merge Request and test it. Then we can merge to the master branch. So this feature / fix will be applied to new projects. Old projects can consider cherry-pick the new commit to use it.

Original README.md

This project was bootstrapped with Create React App.If you want to check original *README*, please go to docs/create-react-app.

Intl

一、在项目根目录建一个：`.env.local`文件(该文件为私密文件，不加入 git 仓库)，里面的内容为：

```
LOKALISE_TOKEN = xxx
```

> token 在`lokalise`中获取: profile

二：同步`lokalise`国际化配置。项目自动化部署过程中会自动抽取并上传新增的国际化文案，本地项目如需同步配置，请执行如下命令

```
yarn lang:sync
```

三：组件库 switch-web-packages 的国际化设置，请通过引用 `react-intl` 中的 hook `useIntl` 引入工具，主项目会扫描公共组件库的文件并抽取。

Others

prettier

- 项目里使用 prettier 插件，确保代码风格统一

commitlint

```
[
    "build", //   build: 对项目构建相关的更改
    "ci", //   ci: 修改.gitlab-ci.yml文件提交
    "docs", //   docs: 文档更新
    "feat", //   feat: 新增功能
    "fix", //   fix: bug 修复
    "perf", //   perf: 性能优化
    "refactor", //   refactor: 重构优化代码
    "revert", //   revert: 回滚之前的提交
    "style", //   style: ui等非逻辑更改
    "chore", //   chore: 不属于以上类型的其他类型
],

example: chore: update readme
```

环境变量

```
{
  ...`.env.development`,
  ...`.env.staging`,
  ...`.env.production`,
  ...`.env.local`
}

const env = process.env.XXX;
```

store

```
// 需要共享状态的父组件注入 Provider

import { useState, useCallback } from "react";
import { createContainer } from "unstated-next";
```

```
const container = createContainer(() => {
    ...
  return { ... };
});

export const DemoProvider = container.Provider;
export const useDemo = container.useContainer;
export default container;

funcion Test() {
    return (
    <DemoProvider>
        <Content/>
    </DemoProvider>
    )
}

function Content() {
    useDemo();
}
```

PortalLayout

```
<PortalLayout>Content</PortalLayout>
```

Form

`form`表单目前使用`react-hook-form`集成

```
import { FormProvider } from "react-hook-form";

<PortalLayout>
  <S.FormWrapper>
    <FormContent />
  </S.FormWrapper>
</PortalLayout>;

function FormContent() {
  return (
    <FormProvider {...methods}>
      <S.Form onSubmit={methods.handleSubmit((data, e) => onSubmit(data,
e))}>
        Content
      </S.Form>
    </FormProvider>
  );
}
```