前端国际化

需求分析：

需要给用户呈现符合其使用偏好和本地环境的前端界面，关注的要素有：使用的语言、排版风格。

项目环境说明

本方案仅适用reactjs技术栈的项目。

使用工具

- react-i18next
- i18next

方案设计

项目结构说明：

```
lang
├── antd.local.js 汇聚第三方组件的语言包和本地化配置
├── i18n.js 注册语言包;初始化i18n的环境
└── locale
    ├── en_us.js en_us地区语言包
    ├── index.js
    └── zh_cn.js zh_cn地区语言包
```

语音包的文件名称取值规范：

- 全部小写
- 与标准地区编码一致
- 使用下划线代替中划线

i18n.js文件内容

```
import i18n from 'i18next';
import { initReactI18next } from 'react-i18next';
import * as languages from './locale';
import * as snUiLocales from 'sn-ui-common-components/dist/locale';

const filenames = Object.keys(languages);
const resources = {};
for (let i = 0; i < filenames.length; i++) {
  const e = filenames[i];
  resources[e] = {
    translation: {
      // 注册应用的语言包
```

```
      ...languages[e],
      // [可选]注册组件库的语言包
      ...snUiLocales[e],
    }
  };
};

i18n
  .use(initReactI18next)
  .init({
    resources: resources,
    // 设置默认语言,取浏览器的默认语言
    lng: navigator.language.replace('-', '_').toLowerCase(),
  });
export default i18n;
```

语音包设计

可以按照模块（领域）划分命名空间，比如：

```
// zh_cn.js
const zh_cn = {
  title: '标题',
  // 通用或者多次使用的文本放在common空间
  common: {
    confirm: {
      title: '确认',
    },
    table: {
      operate: '操作',
    },
    action: {
      add: '添加',
      edit: '编辑',
      delete: '删除',
      deleteBatch: '批量删除',
    }
  },
  // NOTE:按照模块划分命名空间
  /* user模块放置用户相关的数据 */
  user: {
    // 表单域中的提示信息
    tooltip: {
      name: '长度1-10个字符',
    },
    // 校验器提示信息
    validator: {
      name: {
        unique: '名称已被使用',
      },
      ip: {
```

```
        format: 'IP格式不正确',
        range: 'IP取值不合法',
      }
    },
    // 一些枚举数据
    enum: {
      status: {
        enable: '开启',
        disable: '关闭',
      }
    },
    // 标签
    label: {
      name: '用户名称',
      group: '组别',
      status: '状态',
      ip: 'IP地址',
    },
    // 从上下文中获取信息
    // eg：参数name在使用时候由用户传入
    deleteMessage: '你确定要删除{{name}}?',
  },
  // ...
  // 其他模块以此类推
};
```

切换语言

通过调用`i18n.changeLanguage(lng,callback)`，来切换i18next内语言

```
import { useState } from 'react';
import { useTranslation } from 'react-i18next';
import { ConfigProvider } from 'antd';
import * as antdLanguages from './lang/antd.local.js';
// 可选语言包列表
const languageList = ['zh_cn', 'en_us'].map(e => ({ label: e, value: e }))
function App() {
  const { i18n } = useTranslation();
  const [locale, setLocale] = useState(languageList[0]);

  const handleChange = e => {
    i18n.changeLanguage(e);
    setLocale(e);
  };

  return (
    <>
      {/* 切换antd的语言设置*/}
      <ConfigProvider locale={antdLanguages[locale]}>
        <Select value={locale} onChange={handleChange} options=
{languageList} />
```

```
        </ConfigProvider >
      </>
    );
  }

  export default App;
```

在组件中使用

在组件中，通常使用useTranslation这个hook提供t函数，获取翻译后的文本。

以下列举几个需要更替文本的场景：
1.DOM节点中的文本

```
import { useTranslation } from "react-i18next"
export default function MyTitle() {
  const { t } = useTranslation();
  return (
    <h1>{t('title')}</h1>
  )
}
```

2.表单域中的提示信息（在React.Node的props中使用）

```
import { Form, Input } from 'antd'
import { useTranslation } from 'react-i18next'

export default function MyForm() {
  const { t } = useTranslation();
  return (
    <Form>
      <Form.Item
        label={t('user.label.name')}
        name='name'
        tooltip={t('user.tooltip.name')}
        rules={[{
          required: true,
        }]}>
        <Input />
      </Form.Item>
    </Form>
  )
}
```

3.表单校验器中的提示信息（在普通函数中使用）

```
import { Form, Input, Select } from 'antd'
import { useTranslation } from 'react-i18next'

export default function MyForm() {
  const { t } = useTranslation();
  return (
    <Form>
      <Form.Item
        label={t('user.label.ip')}
        name='ip'
        rules={[{
          validator: (_, value) => {
            if (!value) {
              return Promise.resolve();
            }
            const arr = value.split('.');
            if (arr.length !== 4) {
              return Promise.reject(t('user.validator.ip.format'));
            } else {
              if (arr[0] === '0') {
                return Promise.reject(t('user.validator.ip.range'));
              }
            }
          }
        }]}
      >
        <Input />
      </Form.Item>
    </Form>
  )
}
```

4.依赖上下文的提示信息

```
import { Dropdown, Menu, Modal, Table } from 'antd';
import { useTranslation } from 'react-i18next';

function MyTable() {
  const { t } = useTranslation();

  const handleDelete = record => {
    Modal.confirm({
      title: t('common.confirm.title'),
      // NOTE:传递当前上下文的信息
      content: t('user.deleteMessage', { name: record.name }),
    });
  };

  const operationRender = (v, r) => {
    return <Dropdown overlay={
```

```
        <Menu>
          <Menu.Item onClick={() => handleDelete(r)}>删除</Menu.Item>
        </Menu>
      } >
        <span> {t('common.table.operate')} </span>
      </Dropdown>;
    }
    const getColumns = () => [{
      title: t('user.label.name'),
      dataIndex: 'name',
      align: 'center',
    }, {
      title: t('user.label.group'),
      dataIndex: 'group',
      align: 'center',
    }, {
      title: t('user.label.status'),
      dataIndex: 'status',
      align: 'center',
      render: v => v
        ? t('user.enum.status.enable')
        : t('user.enum.status.disable'),
    }, {
      title: t('common.table.operate'),
      dataIndex: 'operate',
      align: 'center',
      render: operationRender,
    }];
    return (
      <div>
        <Table
          columns={getColumns()}
          dataSource={[{
            name: 'san',
            group: 'vip',
            status: true,
          }]} />
      </div>
    )
  }
  export default MyTable;
```

> t函数API介绍

组件库的国际化设计

组件库需要提供默认的语言包。组件库也使用i18next，完成多语言切换。 （可选操作）外部应用在注册自身语言包时候，可以将注册组件库的语言包来覆盖组件库自身的原有的语言包。 只要保持组件语言包的**语言**与外部应用的一致。外部切换语言的时候，组件库也会同步更替文本。

> 如果组件库的语言是zh-CN,外部应用的语言是zh_cn,就需要重新注册语言包。 项目结构的设计如下：

```
lang
├── i18n.js
└── locale
    ├── en_us.js
    ├── index.js
    └── zh_cn.js
```

与外部应用的设计基本一致。

与后端交互

由于部分提示信息来自后端，因此与后端交互过程中需要传递客户端当前的语言设置。

发送请求前，在请求头中添加accept-language字段，传递客户端的当前语言设置。

```
accept-language: zh_cn
```

> 该字段的标准取值应该是zh-CN,此处使用zh_cn是为了兼容前端设计

参考资料

- 《OLT平台实现中英文切换.md》
- 如何解决前端多语言选型和实现难题？ - 掘金
- Introduction - i18next documentation
- Navigator.language - Web API 接口参考 | MDN
- Accept-Language - HTTP | MDN