知乎

React + typescript + antd开发前端应用（二）Hello world

ep76

关注

赞同

分享

上一篇：React + typescript + antd开发前端应用（一）搭建开发环境

完成开发环境搭建后，就可以开始开发自己的React组件了。建议备份创建项目时生成的程序文件，以便作为后续开发时可参考的样例代码。

1、修改端口号

用VSCode打开目录D:\devtools\jssrc\mydemopro，修改默认的3000端口号为8080：

打开src目录下的package.json文件，找到"start": "react-scripts start"，修改为"start": "set PORT=8080 && react-scripts start"，端口号修改为8080。

2、创建新的index.tsx

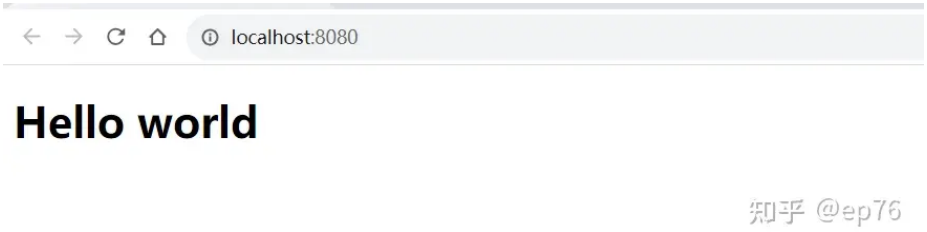
修改创建项目时生成的index.tsx文件名为index.tsx.default，然后创建一个空的index.tsx文件，修改index.tsx文件内容如下：

```
import React from 'react';
import ReactDOM from 'react-dom/client';

const root = ReactDOM.createRoot(document.getElementById('root') as HTMLElement);

root.render(
    <React.StrictMode>
        <h1>Htllo world</h1>
    </React.StrictMode>
);
```

修改完成后，打开VSCode的终端，执行npm run start命令（注意：请先停止cmd中启动的前端经常，Ctrl+C即可停止），默认浏览器将出现Hello world界面，如下图所示：



Hello world界面

从程序可以看到：

1、先采用import语句导入React相关模块：import是typescript的导入模块的语法，和ES6的语法一样。

2、通过ReactDOM模块的API，创建一个root对象：创建对象时需要index.html中id为root的div元素做为参数。

可以通过浏览器F12开发者工具的“检查”功能参考最终在页面上生成的Dom元素。

3、页面组件开发

React之所以如此风靡，其实是它提供了页面组件的开发能力。个人理解的页面组件：就是把一个完整页面的部分区域抽象成一个独立的tsx文件，这个tsx文件就是一个组件。因此我们创建一个目录demo，在demo目录下创建一个文件DemoApp.tsx，编辑文件内容如下：

```
import React from 'react';

function DemoApp() {
    return (
        <h1>Hello world</h1>
    );
}

export default DemoApp;
```

DemoApp.tsx文件定义了一个函数式组件。定义函数式组件的基本过程就是import React相关模块，然后定义一个普通函数，但这个函数的关键点是return()语句中的内容，这些内容近似于需要渲染在页面的html源码。最后就是export default导出函数，这个export default的意义与ES6的意义完全一样。

修改index.tsx文件，引入DemoApp这个组件并渲染：

```
import React from 'react';
import ReactDOM from 'react-dom/client';
import DemoApp from './demo/DemoApp';

const root = ReactDOM.createRoot(document.getElementById('root') as HTMLElement);

root.render(
    <React.StrictMode>
        <DemoApp/>
    </React.StrictMode>
);
```

可以看到，在index.tsx中，首先import了自定义的组件，然后在<React.StrictMode>标签中进行了渲染。

修改完成后，再次刷新浏览器页面，可以得到和之前直接在index.tsx中编写代码得到的结果一模一样。

这就是页面组件化开发。当然就目前的案例来看，这个组件化好像意义不大，不过随着学习的不断深入，会发现组件化看法会越来越方便。需要注意：自定义的组件，标签必须大写字母开始，标准的HTML标签以小写字母开始。

4、验证antd是否可用

修改DemoApp.tsx文件内容，增加两个案例，一个是原生按钮，另外一个是antd的按钮，修改后的代码如下：

```
import React from 'react';
import { Button } from 'antd
```

知乎

```
    return (
        <h1>Hello world</h1>
        <Button type='primary'>antd按钮</Button>
        <button>普通按钮</button>
    );
}

export default DemoApp;
```

在VSCode中，以上代码会编译失败，提示JSX expressions must have one parent element.。意思是标签需要用一个根元素包裹，因此我们可以用 元素包裹所有的标签，但这种方式会在页面上引入一个冗余的 元素（可以通过浏览器开发者工具检查页面元素）。因此我们采用<React.Fragment>标签进行包裹，修改后的代码如下：

```
import React from 'react';
import { Button } from 'antd';

function DemoApp() {
    return (
        <React.Fragment>
            <h1>Hello world</h1>
            <Button type='primary'>antd按钮</Button>
            <button>普通按钮</button>
        </React.Fragment>
    );
}

export default DemoApp;
```

保存后，浏览器中的页面会自动刷新，如下图所示：

antd按钮与普通按钮

通过按钮外观，可以看出antd按钮比普通按钮顺眼多了，这也证明了工程中引入antd UI组件成功。

至此，基本完成了开发环境的搭建验证，同时也完成了第一个自定义组件的定义。

下一篇：React + typescript + antd开发前端应用（三）todo list

编辑于 2023-10-02 02:00 · IP 属地上海

Angular Ant Design TypeScript

知乎

欢迎参与讨论

还没有评论，发表第一个评论吧

推荐阅读

TypeScript在react项目中的实践

慕课网 发表于猿论

2021 年 TypeScript + React 工程化指南

Henry 发表于阿里 CC...

Typescript & React 相关积累

介绍 Typescript 配合 React 的基本使用，和一些实用的小技巧。 主要目的在于完善基本认知的统一，并对一些十分常见且实用的功能能够有一定的深入学习。 如何寻找一个类型文件位置常见 node...

折木

为什 Fibe

前端