link: null
title: 珠峰架构师成长计划
description: src\index.tsx
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=29 sentences=171, words=1004

1.生成项目

```
create-react-app zhufengdrag --typescript
cd zhufengdrag
cnpm install antd koa  koa-body koa-static -S
cnpm start
```

2.上传组件

src\index.tsx

```
import React from 'react';
import ReactDOM from 'react-dom';
import 'antd/dist/antd.css';

import { Upload, Icon, message } from 'antd';
const { Dragger } = Upload;
const props = {
    name: 'file',
    action: 'https://www.mocky.io/v2/5cc8019d300000980a055e76',
    onChange(info: any) {
        console.log(info);
        const { status } = info.file;
        if (status === 'done') {
            message.success(`${info.file.name} 上传成功!`);
        } else if (status === 'error') {
            message.error(`${info.file.name} 上传失败!`);
        }
    },
};
ReactDOM.render(
    <Dragger {...props}><Icon type="inbox" />Dragger>,
    document.getElementById('root'),
);
```

src\Dragger\index.tsx

```
import React from 'react';
interface Props {
}
const Dragger: React.SFC = function (props: Props): JSX.Element {
    return (
        <div>
            Dragger
        div>
    )
}
export default Dragger;
```

3. 布局上传组件

Dragger\index.tsx

```
import React from 'react';
import './index.css';
type Props = React.PropsWithChildren
const Dragger: React.SFC = function (props: Props): JSX.Element {
    return (
        <div className="dragger-container">
            {props.children}
        div>
    )
}
export default Dragger;
```

src\Dragger\index.css

```
.dragger-container{
    position: relative;
    width: 100%;
    height:200px;
    background: #FAFAFA;
    border:1px dashed #D9D9D9;
    border-radius: 4px;
    cursor: pointer;
    display: flex;
    justify-content: center;
    align-items: center;
    transition: border-color .8s;
}
.dragger-container:hover{
    border-color:#40A9FF;
}
.dragger-container i{
    font-size:100px;
}
```

4. 进度条

```
import React from 'react';
import ReactDOM from 'react-dom';
import 'antd/dist/antd.css';
import Dragger, { UploadFile, DragProps } from './Dragger';
import { Upload, Icon, message } from 'antd';

const props: DragProps = {
    name: 'file',

    action: 'http://localhost:8080/upload',
    onUpload(uploadFile: UploadFile) {
        console.log(uploadFile);
        if (uploadFile.error) {
            message.error(`${uploadFile.file.name} 上传失败!`);
        } else {
            message.success(`${uploadFile.file.name} 上传成功!`);
        }
    },
};
ReactDOM.render(
    <Dragger {...props}><Icon type="inbox" />Dragger>,
    document.getElementById('root'),
);
```

- `useRef` 返回一个可变的 `ref` 对象，其 `.current` 属性被初始化为传入的参数 `(initialValue)`返回的 ref 对象在组件的整个生命周期内保持不变
- `useEffect` 就是一个 Effect Hook，给函数组件增加了操作副作用的能力。它跟 class 组件中的 componentDidMount、componentDidUpdate 和 componentWillUnmount 具有相同的用途，只不过被合并成了一个 API
- `useState` 就是一个 Hook，通过在函数组件里调用它来给组件添加一些内部 state,React 会在重复渲染时保留这个 state，useState 会返回一对值：当前状态和一个让你更新它的函数

拖动目标上触发事件(源元素)

事件 触发 ondragstart 用户开始拖动元素时触发 ondrag 元素正在拖动时触发 ondragend 用户完成元素拖动后触发

释放目标时触发的事件

事件 触发 ondragenter 当被鼠标拖动的对象进入其容器范围内时触发此事件 ondragover 当某被拖动的对象在另一对象容器范围内拖动时触发此事件,如果需要设置允许放置，我们必须阻止对元素的默认处理方式 ondragleave 当被鼠标拖动的对象离开其容器范围内时触发此事件 ondrop 在一个拖动过程中，释放鼠标键时触发此事件

src\Dragger\index.tsx

```
import React, {
    useRef, MutableRefObject, RefObject, useEffect, useState
} from 'react';
import './index.css';
import { Progress, Icon } from 'antd';
export type DragProps = React.PropsWithChildrenonUpload: any;
    name: string;
    action: string
}>
export interface UploadFile {
    file: File;
    percent?: number;
    url?: string;
    uploading?: boolean;
    error?: boolean
}
const Dragger: React.SFC = function (props: DragProps): JSX.Element {
    let [uploadFiles, setUploadFiles] = useState<Array>([]);
    let uploadContainer: MutableRefObjectundefined> = useRef();
    const onDragEnter: (ev: DragEvent) => any = (ev: DragEvent): any => {
        ev.preventDefault();
        ev.stopPropagation();
    };
    const onDragOver = (ev: DragEvent): any => {
        ev.preventDefault();
        ev.stopPropagation();
    };
    const onDragLeave = (ev: DragEvent): any => {
        ev.preventDefault();
        ev.stopPropagation();
    };
    const onDrop = (ev: DragEvent): any => {
        ev.preventDefault();
        ev.stopPropagation();
        let transfer: DataTransfer | null = ev.dataTransfer;
        if (transfer && transfer.files) {
            upload(transfer.files);
        }
    };
    function upload(files: DataTransfer['files']) {
        for (let i = 0; i < files.length; i++) {
            let file = files[i];
            let formData = new FormData();
            formData.append('filename', file.name);
            formData.append(props.name, file);
            var xhr: XMLHttpRequest = new XMLHttpRequest();
            xhr.open('POST', props.action, true);
            xhr.responseType = 'json';
            let uploadFile: UploadFile = { file, percent: 0, uploading: true, error: false };
            xhr.onreadystatechange = function () {
                if (xhr.readyState == 4 && xhr.status === 200) {
                    debugger;
                    uploadFile.url = xhr.response.url;
                    props.onUpload(uploadFile);
                }
            }
            uploadFiles.push(uploadFile);
            xhr.onprogress = updateProgress;
            xhr.upload.onprogress = updateProgress;
            function updateProgress(event: ProgressEvent) {
                if (event.lengthComputable) {
                    let percent: number = parseInt((event.loaded / event.total * 100).toFixed(0));
                    uploadFile.percent = percent;
                    if (percent >= 100) {
                        uploadFile.uploading = false;
                    }
                    setUploadFiles([...uploadFiles]);
                }
            }
            xhr.onerror = function () {
                uploadFile.error = true;
                uploadFile.uploading = false;
                setUploadFiles([...uploadFiles]);
            }
            xhr.ontimeout = function () {
                uploadFile.error = true;
                uploadFile.uploading = false;
                setUploadFiles([...uploadFiles]);
            }
            xhr.send(formData);
        }
    }
    useEffect(() => {
        uploadContainer.current!.addEventListener('dragenter', onDragEnter);
        uploadContainer.current!.addEventListener('dragover', onDragOver);
        uploadContainer.current!.addEventListener('drop', onDrop);
        uploadContainer.current!.addEventListener('dragleave', onDragLeave);
        return () => {
            uploadContainer.current!.removeEventListener('dragenter', onDragEnter);
            uploadContainer.current!.removeEventListener('dragover', onDragOver);
            uploadContainer.current!.removeEventListener('drop', onDrop);
            uploadContainer.current!.removeEventListener('dragleave', onDragLeave);
        }
    })
    return (
        <>
            <div className="dragger-container" ref={uploadContainer as RefObject<HTMLDivElement> | null | undefined}>
                {props.children}
            div>
            {
                uploadFiles.map((uploadFile: UploadFile, index: number) => (
                    <div key={index}>
                        <div>
                            {!uploadFile.error && <Icon type={uploadFile.uploading ? 'loading' : 'paper-clip'} />}
                            <span style={{ marginLeft: 10 }}>{uploadFile.file.name}span>
                        div>
                        <Progress status={uploadFile.error ? 'exception' : undefined} key={index} percent={uploadFile.percent} />
                    div>
                ))
            }
        </>
    )
}
export default Dragger;
```

5. 后端接口

```
let koaStatic = require('koa-static');
let path = require('path');
let koaBody = require('koa-body');
let fs = require('fs');
let Koa = require('koa');
let app = new Koa();
app.use(async (ctx, next) => {
    ctx.set('Access-Control-Allow-Origin', '*');
    ctx.set('Access-Control-Allow-Headers', 'Content-Type, Accept');
    ctx.set('Access-Control-Allow-Methods', 'PUT, POST, GET, DELETE, OPTIONS');
    if (ctx.method == 'OPTIONS') {
        ctx.body = 200;
    } else {
        await next();
    }
});
app.use(koaBody({
    formidable: { uploadDir: path.resolve(__dirname, './uploads') },
    multipart: true
}));

app.use(koaStatic(
    path.resolve(__dirname, './uploads')
));
app.use(async (ctx, next) => {
    if (ctx.url === '/upload') {
        let file = ctx.request.files.file;
        let filename = path.basename(file.path) + path.extname(file.name);
        fs.renameSync(file.path, path.join(path.dirname(file.path), filename));
        ctx.body = { url: "http://localhost:8080/" + filename };
    } else {
        await next();
    }
});

app.listen(8080, () => {
    console.log('服务器成功在8080端口上启动!');
});
```