link: null
title: 珠峰架构师成长计划
description: null
keywords: null
author: null
date: null
publisher: 珠峰架构师成长计划
stats: paragraph=78 sentences=264, words=1543

1.create-react-app [#]

- Create React App (https://www.html.cn/create-react-app/docs/getting-started/)是一个官方支持的创建 React 单页应用程序的方法。它提供了一个零配置的现代构建设置
- create-react-app (https://github.com/facebook/create-react-app)

1.1 下载 [#]

```
git clone https:
cd create-react-app
yarn install
```

1.2 package.json [#]

package.json

```
"scripts": {
+  "create": "node ./packages/create-react-app/index.js",
}
```

1.3 重要步骤 [#]

- 将命令行参数发送到npm脚本 `npm run [command] [-- <args>]</args>`

```
yarn install  #安装项止依赖和软链接
npm run create -- aaa  #执行创建命令
Installing packages. This might take a couple of minutes. #安装依赖包
Installing react, react-dom, and react-scripts with cra-template... #安装依赖包
Installing template dependencies using yarnpkg... #安装模板依赖
Removing template package using yarnpkg... #移除模板模块
Removing module cra-template...  #移除cra-template模块
Success! Created aaa at C:\aprepare\create-react-app\aaa #成功创建
Inside that directory, you can run several commands: #执行命令
cd aaa
yarn start
```

1.4 .vscode\launch.json [#]

.vscode\launch.json

```
{
    "version": "0.2.0",
    "configurations": [
        {
            "name": "Launch via NPM",
            "request": "launch",
            "runtimeArgs": [
                "run-script",
                "create"
            ],
            "runtimeExecutable": "npm",
            "skipFiles": [
                "/**"
            ],
            "type": "pwa-node"
        }
    ]
}
```

2.实现init方法 [#]

2.1 package.json [#]

package.json

```
  "scripts": {
+    "version": "node ./packages/create-react-app3/index.js --version",
+    "create": "node ./packages/create-react-app3/index.js aaa"
  }
```

2.2 create-react-app3\package.json [#]

packages\create-react-app3\package.json

```
{
+ "main": "./index.js"
}
```

2.3 create-react-app3\index.js [#]

packages\create-react-app3\index.js

```

const { init } = require('./createReactApp');
init();
```

2.4 createReactApp.js [#]

packages\create-react-app3\createReactApp.js

```
const {Command} = require('commander');
const chalk = require('chalk');
const packageJson = require('./package.json');
let appName;
async function init() {
    new Command(packageJson.name)
        .version(packageJson.version)
        .arguments('')
        .usage(`${chalk.green('')} [options]`)
        .action(projectDirectory => {
            appName = projectDirectory;
        })
        .parse(process.argv);
    console.log('appName=', appName);
}
module.exports = {
    init
}
```

2.5 执行命令

```
npm run create
```

3.实现createApp方法

3.1 createReactApp.js

packages\create-react-app3\createReactApp.js

```
const {Command} = require('commander');
const chalk = require('chalk');
+const fs = require('fs-extra');
+const path = require('path');
const packageJson = require('./package.json');
let appName;
async function init() {
    new Command(packageJson.name)
        .version(packageJson.version)
        .arguments('')
        .usage(`${chalk.green('')} [options]`)
        .action(projectDirectory => {
            appName = projectDirectory;
        })
        .parse(process.argv);
    console.log('appName=', appName);
+   await createApp(appName);
}
+async function createApp(appName) {
+    const root = path.resolve(appName);
+    fs.ensureDirSync(appName);
+    console.log(`Creating a new React app in ${chalk.green(root)}.`);
+    const packageJson = {
+      name: appName,
+      version: '0.1.0',
+      private: true,
+    };
+    fs.writeFileSync(
+      path.join(root, 'package.json'),
+      JSON.stringify(packageJson, null, 2)
+    );
+    const originalDirectory = process.cwd();
+    process.chdir(root);
+    console.log('root',root);
+    console.log('appName',appName);
+    console.log('originalDirectory',originalDirectory);
+  }
module.exports = {
    init
}
```

4.实现run方法

4.1 createReactApp.js

packages\create-react-app3\createReactApp.js

```
const {Command} = require('commander');
const chalk = require('chalk');
const fs = require('fs-extra');
const path = require('path');
+const spawn = require('cross-spawn');
const packageJson = require('./package.json');
let appName;
async function init() {
    new Command(packageJson.name)
        .version(packageJson.version)
        .arguments('')
        .usage(`${chalk.green('')} [options]`)
        .action(projectDirectory => {
            appName = projectDirectory;
        })
        .parse(process.argv);
    console.log('appName=', appName);
    await createApp(appName);
}
async function createApp(appName) {
    const root = path.resolve(appName);
    fs.ensureDirSync(appName);
    console.log(`Creating a new React app in ${chalk.green(root)}.`);
    const packageJson = {
      name: appName,
      version: '0.1.0',
      private: true,
    };
    fs.writeFileSync(
      path.join(root, 'package.json'),
      JSON.stringify(packageJson, null, 2)
    );
    const originalDirectory = process.cwd();
    process.chdir(root);
    console.log('root',root);
    console.log('appName',appName);
    console.log('originalDirectory',originalDirectory);
+   await run(
+        root,
+        appName,
+        originalDirectory
+   );
}
+async function run(root,appName,originalDirectory) {
+    const scriptName = 'react-scripts';
+    const templateName = 'cra-template';
+    const allDependencies = ['react', 'react-dom', scriptName, templateName];
+    console.log('Installing packages. This might take a couple of minutes.');
+    console.log(
+      `Installing ${chalk.cyan('react')}, ${chalk.cyan(
+        'react-dom'
+      )}, and ${chalk.cyan(scriptName)} with ${chalk.cyan(templateName)}`
+    );
+    await install(root, allDependencies);
+}
+function install(root, allDependencies) {
+    return new Promise((resolve) => {
+      const command = 'yarnpkg';
+      const args = ['add', '--exact', ...allDependencies, '--cwd', root];
+      console.log('command:',command,args);
+      const child = spawn(command, args, { stdio: 'inherit' });
+      child.on('close', resolve);
+    });
+}
module.exports = {
    init
}

command: yarnpkg [
  'add',
  '--exact',
  'react',
  'react-dom',
  'react-scripts',
  'cra-template',
  '--cwd',
  'C:\\aprepare\\create-react-app3\\aaa'
]

yarnpkg add --exact react react-dom react-scripts cra-template --cwd C:\\aprepare\\create-react-app3\\aaa
```

5.执行init初始化命令

5.1 createReactApp.js

packages\create-react-app3\createReactApp.js

```
const {Command} = require('commander');
const chalk = require('chalk');
const fs = require('fs-extra');
const path = require('path');
const spawn = require('cross-spawn');
const packageJson = require('./package.json');
let appName;
async function init() {
    new Command(packageJson.name)
        .version(packageJson.version)
        .arguments('')
        .usage(`${chalk.green('')} [options]`)
        .action(projectDirectory => {
            appName = projectDirectory;
        })
        .parse(process.argv);
    console.log('appName=', appName);
    await createApp(appName);
}
async function createApp(appName) {
    const root = path.resolve(appName);
    fs.ensureDirSync(appName);
    console.log(`Creating a new React app in ${chalk.green(root)}.`);
    const packageJson = {
      name: appName,
      version: '0.1.0',
      private: true,
    };
    fs.writeFileSync(
      path.join(root, 'package.json'),
      JSON.stringify(packageJson, null, 2)
    );
    const originalDirectory = process.cwd();
    process.chdir(root);
    console.log('root',root);
    console.log('appName',appName);
    console.log('originalDirectory',originalDirectory);
    await run(
        root,
        appName,
        originalDirectory
    );
}
async function run(root,appName,originalDirectory) {
    const scriptName = 'react-scripts';
    const templateName = 'cra-template';
    const allDependencies = ['react', 'react-dom', scriptName, templateName];
    console.log('Installing packages. This might take a couple of minutes.');
    console.log(
      `Installing ${chalk.cyan('react')}, ${chalk.cyan(
        'react-dom'
      )}, and ${chalk.cyan(scriptName)} with ${chalk.cyan(templateName)}`
    );
    await install(root, allDependencies);
+    let data = [root, appName, true, originalDirectory, templateName];
+    let source = `
+    var init = require('react-scripts/scripts/init.js');
+    init.apply(null, JSON.parse(process.argv[1]));
+  `
+    await executeNodeScript({ cwd: process.cwd() }, data, source);
+    console.log('Done.');
+    process.exit(0);
}
+function executeNodeScript({ cwd }, data, source) {
+  return new Promise((resolve) => {
+    const child = spawn(
+      process.execPath,
+      ['-e', source, '--', JSON.stringify(data)],
+      { cwd, stdio: 'inherit' }
+    );
+    child.on('close', resolve);
+  });
+}
function install(root, allDependencies) {
    return new Promise((resolve) => {
      const command = 'yarnpkg';
      const args = ['add', '--exact', ...allDependencies, '--cwd', root];
      console.log('command:',command,args);
      const child = spawn(command, args, { stdio: 'inherit' });
      child.on('close', resolve);
    });
}
module.exports = {
    init
}
```

1. monorepo管理

- Monorepo 是管理项目代码的一个方式，指在一个项目仓库(repo)中管理多个模块/包(package)
- monorepo 最主要的好处是统一的工作流和代码共享
- Lerna (https://github.com/lerna/lerna)是一个管理多个 npm 模块的工具,优化维护多包的工作流，解决多个包互相依赖，且发布需要手动维护多个包的问题
- yarn (https://classic.yarnpkg.com/en/docs/cli/)

1.1 安装

```
npm i lerna -g
```

1.2 初始化

```
lerna init
```

1.2.1 package.json

package.json

```
{
  "name": "root",
  "private": true,
  "devDependencies": {
    "lerna": "^3.22.1"
  }
}
```

1.2.2 lerna.json

lerna.json

```
{
  "packages": [
    "packages/*"
  ],
  "version": "0.0.0"
}
```

1.3 yarn workspace

- `yarn workspace`允许我们使用 `monorepo` 的形式来管理项目
- 在安装 node_modules 的时候它不会安装到每个子项目的 node_modules 里面，而是直接安装到根目录下面，这样每个子项目都可以读取到根目录的 node_modules
- 整个项目只有根目录下面会有一份 `yarn.lock` 文件。子项目也会被 `link` 到 `node_modules` 里面，这样就允许我们就可以直接用 import 导入对应的项目
- `yarn.lock`文件是自动生成的,也完全Yarn来处理. `yarn.lock`锁定你安装的每个依赖项的版本，这可以确保你不会意外获得不良依赖

1.3.1 开启workspace

package.json

```
{
  "name": "root",
  "private": true, // 私有的,用来管理整个项目,不会被发布到npm
+  "workspaces": [
+    "packages/*"
+  ],
  "devDependencies": {
    "lerna": "^3.22.1"
  }
}
```

1.3.2 创建子项目

```
lerna create create-react-app3
lerna create react-scripts3
lerna create cra-template3
```

1.3.3 添加依赖

- yarnpkg (https://classic.yarnpkg.com/en/docs/cli)
- lerna (https://github.com/lerna/lerna#readme)

设置加速镜像

```
yarn config get registry
yarn config set registry http://registry.npm.taobao.org/
yarn config set registry http://registry.npmjs.org/
```

作用 命令 查看工作空间信息 yarn workspaces info 给根空间添加依赖 yarn add chalk cross-spawn fs-extra --ignore-workspace-root-check 给某个项目添加依赖 yarn workspace create-react-app3 add commander 删除所有的 node_modules lerna clean 等于 yarn workspaces run clean 安装和link yarn install 等于 lerna bootstrap --npm-client yarn --use-workspaces 重新获取所有的 node_modules yarn install --force 查看缓存目录 yarn cache dir 清除本地缓存 yarn cache clean

2. commander

- chalk (https://www.npmjs.com/package/chalk)可以在终端显示颜色
- commander (https://github.com/tj/commander.js/blob/HEAD/Readme_zh-CN.md)是一个完整的 `node.js`命令行解决方案
- version方法可以设置版本，其默认选项为 `-V`和 `--version`
- 通过.arguments可以为最顶层命令指定参数，对子命令而言，参数都包括在.command调用之中了。尖括号（例如）意味着必选，而方括号（例如[optional]）则代表可选
- 通过 `usage`选项可以修改帮助信息的首行提示

```
const chalk = require('chalk');
const {Command} = require('commander');
console.log('process.argv',process.argv);
new Command('create-react-app')
    .version('1.0.0')
    .arguments('  [optional]')
    .usage(`${chalk.green(' ')} [optional]`)
    .action((must1,must2,optional,...args) => {
       console.log(must1,must2,optional,args);
    })
    .parse(process.argv);
```

3. cross-spawn

- cross-spawn (https://www.npmjs.com/package/cross-spawn)是node的 `spawn`和 `spawnSync`的跨平台解决方案
- inherit (https://nodejs.org/dist/latest-v15.x/docs/api/child_process.html)表示将相应的 `stdio`流传给父进程或从父进程传入

```
const spawn = require('cross-spawn');
const child = spawn('node', ['script.js','one','two','three'], { stdio: 'inherit' });
child.on('close',()=>{
    console.log('child is done!');
});
const result = spawn.sync('node', ['script.js','one','two','three'], { stdio: 'inherit' });
console.log(result);
```