[快速上手 2](#_Toc23940)

[第一个本地实例 2](#_Toc26362)

[安装脚手架工具 2](#_Toc31547)

[创建一个项目 2](#_Toc6139)

[使用组件 2](#_Toc13025)

[开发调试 2](#_Toc21936)

[构建和部署 3](#_Toc32239)

[兼容性 3](#_Toc20891)

[个人总结 3](#_Toc13836)

[项目实战 3](#_Toc23629)

[安装 dva-cli 3](#_Toc22781)

[创建新应用 3](#_Toc5640)

[使用 antd 4](#_Toc324)

[定义路由 4](#_Toc25958)

[编写 UI Component 4](#_Toc24754)

[定义 Model 4](#_Toc27687)

[connect model 和 component 5](#_Toc13420)

[构建应用 5](#_Toc13215)

[在 create-react-app 中使用 5](#_Toc9782)

[安装和初始化 5](#_Toc21843)

[引入 antd 6](#_Toc27907)

[高级配置 6](#_Toc18891)

[引入 react-app-rewired 6](#_Toc20252)

[配置按需加载 6](#_Toc5794)

[自定义主题 7](#_Toc31318)

[在 TypeScript 中使用 7](#_Toc7764)

[安装和初始化 7](#_Toc2871)

# 快速上手

## 第一个本地实例

实际项目开发中，你会需要对 ES2015 和 JSX 代码的构建、调试、代理、打包部署等一系列工程化的需求。 我们提供了一套 npm + webpack 的开发工具链来辅助开发。

### 安装脚手架工具

antd-init 是一个用于演示 antd 如何使用的脚手架工具，实际业务项目建议使用 dva-cli 和 create-react-app 进行搭建。

npm install antd-init -g

### 创建一个项目

使用命令行进行初始化。

mkdir antd-demo && cd antd-demo

antd-init

antd-init 会自动安装 npm 依赖，若有问题则可自行安装。

若安装缓慢报错，可尝试用 cnpm 或别的镜像源自行安装：

rm -rf node\_modules && cnpm install。

初始化不成功，报没有eslint-plugin-import@^1.16.0，自行安装：

npm install --save-dev eslint-plugin-import@^1.16.0

### 使用组件

用 React 的方式直接使用 antd 组件。

import React from 'react';

import ReactDOM from 'react-dom';

import { LocaleProvider, DatePicker, message } from 'antd';

// 由于 antd 组件的默认文案是英文，所以需要修改为中文

import zhCN from 'antd/lib/locale-provider/zh\_CN';

import moment from 'moment';

import 'moment/locale/zh-cn';

moment.locale('zh-cn');

### 开发调试

一键启动调试，访问 http://127.0.0.1:8000 查看效果。

npm start

### 构建和部署

npm run build

入口文件会构建到 dist 目录中，你可以自由部署到不同环境中进行引用。

## 兼容性

Ant Design React 支持所有的现代浏览器和 IE9+。

对于 IE 系列浏览器，需要提供 es5-shim 和 es6-shim 等 Polyfills 的支持。

如果你使用了 babel，强烈推荐使用 babel-polyfill 和 babel-plugin-transform-runtime 来替代以上两个 shim。

避免同时使用 babel 和 shim 两种兼容方法，以规避 #6512 中所遇问题

如果在 IE 浏览器中遇到 startsWith 的问题，引入 es6-shim 或 babel-polyfill。

## 个人总结

npm install antd-init -g

mkdir antd-demo && cd antd-demo

antd-init

更改文件，用React 的方式直接使用 antd 组件。

npm start

npm run build

兼容性导包es5-shim+es6-shim，或者babel-polyfill+babel-plugin-transform-runtime。

# 项目实战

dva 是一个基于 React 和 Redux 的轻量应用框架，概念来自 elm，支持 side effects、热替换、动态加载、react-native、SSR 等，已在生产环境广泛应用。

## 安装 dva-cli

通过 npm 安装 dva-cli 并确保版本是 0.9.1 或以上。

npm install dva-cli -g

dva -v

## 创建新应用

dva new dva-quickstart 创建dva项目

cd dva-quickstart && npm start 进入项目目录，启动项目

欢迎界面：[http://localhost:8000](http://localhost:8000。)

报错，无模块：'babel-plugin-import'；安装模块，重新生成项目后正常。

再次启动时报同样错误，将nodejs下文件夹拷到指定位置。

## 使用 antd

通过 npm 安装 antd 和 babel-plugin-import 。babel-plugin-import 用来按需加载 antd 的脚本和样式 。

npm install antd babel-plugin-import --save

编辑 .webpackrc，加入信息；使 babel-plugin-import 插件生效。

{

"extraBabelPlugins": [

["import", { "libraryName": "antd", "libraryDirectory": "es", "style": "css" }]

]

}

## 定义路由

在.\src\routes下创建路由文件；路由可以想象成是组成应用的不同页面。

import React from 'react';

const Products = (props) => (

<h2>List of Products</h2>

);

export default Products;

编辑 router.js；导入路由，配置映射。

import Products from './routes/Products';

<Route path="/products" exact component={Products} />

查看结果： [http://localhost:8000/#/products](http://localhost:8000/" \l "/products)

## 编写 UI Component

在多个页面分享 UI 元素 (或在一个页面使用多次)，抽成 component，在.\src\components下 。

## 定义 Model

dva 通过 model 的概念把一个领域的模型管理起来，包含同步更新 state 的 reducers，处理异步逻辑的 effects，订阅数据源的subscriptions 。

新建 model ./src/models/products.js ：

export default {

namespace: 'products', //全局 state 上的 key

state: [], //初始值，在这里是空数组

reducers: { //等同于 redux 里的 reducer，接收 action，同步更新 state

'delete'(state, { payload: id }) {

return state.filter(item => item.id !== id);

},

},

};

在 index.js 里载入：

app.model(require('./models/products').default);

## connect model 和 component

dva 提供了 connect 方法，串联model 和 component。

在路由Products中导入connect方法、component组件；

import { connect } from 'dva';

import ProductList from '../components/ProductList';

## 构建应用

完成开发并且在开发环境验证之后，就需要部署给我们的用户了。先执行下面的命令：

npm run build

component负责组件显示，model负责领域的模型管理，逻辑处理；路由负责组合显示。connect方法连接组件--路由。index.js加载

初始数据？

区别？export default connect()(IndexPage)，export default Products;

# 在 create-react-app 中使用

## 安装和初始化

在命令行中安装 create-react-app 工具，你可能还需要安装 yarn。

npm install -g create-react-app yarn

新建项目；工具会自动初始化一个脚手架并安装 React 项目的各种必要依赖。

create-react-app antd-demo

进入项目并启动。

cd antd-demo && yarn start

## 引入 antd

从 yarn 或 npm 安装并引入 antd。

yarn add antd

修改 src/App.js，引入 antd 的按钮组件，并显示。

import Button from 'antd/lib/button';

class App extends Component {

render() {

return (

<div className="App">

<Button type="primary">Button</Button>

</div>

);

}

}

修改 src/App.css，在文件顶部引入 antd/dist/antd.css。

@import '~antd/dist/antd.css';

## 高级配置

### 引入 react-app-rewired

安装

yarn add react-app-rewired --dev

修改 package.json 里的启动配置；修改start、build、test对应命令为react-app-rewired

/\* package.json \*/

"scripts": {

"start": "react-app-rewired start",

"build": "react-app-rewired build",

"test": "react-app-rewired test --env=jsdom",

}

### 配置按需加载

在项目根目录创建一个 config-overrides.js， 用于修改默认配置。

module.exports = function override(config, env) {

// do stuff with the webpack config...

return config;

};

使用 babel-plugin-import

安装

yarn add babel-plugin-import --dev

修改 config-overrides.js 文件。

const { injectBabelPlugin } = require('react-app-rewired');

module.exports = function override(config, env) {

config = injectBabelPlugin(['import', { libraryName: 'antd', libraryDirectory: 'es', style: 'css' }], config);

return config;

};

移除src/App.css 中导入的样式代码

修改src/App.js引入模块代码。

import { Button } from 'antd';

## 自定义主题

自定义主题需要用到 less 变量覆盖功能。引入react-app-rewire-less加载 less 样式。

yarn add react-app-rewire-less --dev

修改 config-overrides.js 文件。

const { injectBabelPlugin } = require('react-app-rewired');

const rewireLess = require('react-app-rewire-less');

module.exports = function override(config, env) {

config = injectBabelPlugin(['import', { libraryName: 'antd', style: true }], config);

config = rewireLess.withLoaderOptions({

modifyVars: { "@primary-color": "#1DA57A" },//重新配色

})(config, env);

return config;

};

# 在 TypeScript 中使用

## 安装和初始化

使用 yarn 创建项目

yarn create-react-app antd-demo-ts --scripts-version=react-scripts-ts

使用 npm创建项目

npm install -g create-react-app

create-react-app antd-demo-ts --scripts-version=react-scripts-ts

进入项目并启动。

cd antd-demo-ts && yarn start