[快速上手 4](#_Toc10460)

[第一个本地实例 4](#_Toc17164)

[安装脚手架工具 4](#_Toc17349)

[创建一个项目 4](#_Toc8940)

[使用组件 4](#_Toc13298)

[开发调试 4](#_Toc15904)

[构建和部署 5](#_Toc19910)

[兼容性 5](#_Toc8826)

[个人总结 5](#_Toc11993)

[项目实战 5](#_Toc27126)

[安装 dva-cli 5](#_Toc22749)

[创建新应用 5](#_Toc10400)

[使用 antd 6](#_Toc11360)

[定义路由 6](#_Toc11956)

[编写 UI Component 6](#_Toc2801)

[定义 Model 6](#_Toc26814)

[connect model 和 component 7](#_Toc25404)

[构建应用 7](#_Toc13766)

[在 create-react-app 中使用 7](#_Toc16137)

[安装和初始化 7](#_Toc3019)

[引入 antd 8](#_Toc27215)

[高级配置 8](#_Toc17688)

[引入 react-app-rewired 8](#_Toc17330)

[配置按需加载 8](#_Toc1332)

[自定义主题 9](#_Toc2602)

[在 TypeScript 中使用 9](#_Toc23176)

[安装和初始化 9](#_Toc22690)

[定制主题 10](#_Toc10727)

[Component 10](#_Toc18384)

[General 10](#_Toc7067)

[Button-按钮 10](#_Toc12186)

[Icon-图标 10](#_Toc15786)

[Layout 10](#_Toc3260)

[Grid-栅格 10](#_Toc28014)

[Layout-布局 11](#_Toc29841)

[Navigation 11](#_Toc27314)

[Affix固钉 11](#_Toc4134)

[Breadcrumb面包屑 12](#_Toc18672)

[Dropdown下拉菜单 12](#_Toc8462)

[Menu导航菜单 12](#_Toc22385)

[Pagination分页 13](#_Toc25987)

[Step步骤条 13](#_Toc8015)

[Data Entry 13](#_Toc7894)

[AutoComplete自动完成 13](#_Toc20143)

[Cascader级联选择 14](#_Toc21579)

[Checkbox多选框 14](#_Toc25264)

[DatePicker日期选择框 14](#_Toc32199)

[Form表单 15](#_Toc11740)

[Input输入框 15](#_Toc23806)

[InputNumber数字输入框 16](#_Toc1827)

[Mention提及 16](#_Toc16644)

[Rate评分 16](#_Toc3493)

[Radio单选框 16](#_Toc3945)

[Select选择器 17](#_Toc4054)

[Slider滑动输入条 17](#_Toc1874)

[Switch开关 17](#_Toc22725)

[TreeSelect树选择 18](#_Toc5670)

[TimePicker时间选择框 18](#_Toc31881)

[Transfer穿梭框 18](#_Toc4802)

[Upload上传 19](#_Toc17694)

[Data Display 19](#_Toc7010)

[Avatar头像 19](#_Toc11073)

[Badge徽标数 19](#_Toc29585)

[Calendar日历 20](#_Toc25352)

[Card卡片 20](#_Toc27362)

[Carousel走马灯 20](#_Toc577)

[Collapse折叠面板 21](#_Toc23116)

[List列表 21](#_Toc14057)

[Popover气泡卡片 21](#_Toc30024)

[Tooltip文字提示 22](#_Toc30262)

[Table表格 22](#_Toc21815)

[Tabs标签页 22](#_Toc4358)

[Tag标签 22](#_Toc6291)

[Timeline时间轴 23](#_Toc16562)

[Tree树型控件 23](#_Toc31984)

[Feedback 23](#_Toc6040)

[Alert警告提示 23](#_Toc20966)

[Drawer抽屉 24](#_Toc23259)

[Modal对话框 24](#_Toc21584)

[Message全局提示 24](#_Toc14115)

[Notification通知提示框 25](#_Toc15498)

[Progress进度条 25](#_Toc4197)

[Popconfirm气泡确认框 25](#_Toc24466)

[Spin加载中 26](#_Toc27907)

[Other 26](#_Toc565)

[Anchor锚点 26](#_Toc16061)

[BackTop回到顶部 26](#_Toc14007)

[Divider分割线 26](#_Toc8263)

# 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

# 定制主题

# Component

## General

### Button-按钮

推荐设置顺序：type-shape-size-loading-disable

type，按钮类型；值：primary、不设、dashed、danger

shape，按钮形状；值：circle、不设

icon，按钮图标；可设置icon属性，或使用icon组件(需控制位置时使用)。

size，按钮大小；值：large、不设、small。

disabled，禁用；

ghost，虚化；

loading，加载；

block，和容器同宽。

href，点击跳转地址；使用本属性，最终被渲染为<a>标签。

<ButtonGroup>，类似<RadioGroup>，包裹多个Button。

按钮组合使用时，推荐使用1个主按钮+n个次按钮。

### Icon-图标

语义化的矢量图形。

type，图标类型；类型查询：<https://ant.design/components/icon-cn/>

style，图标样式；图标字体本质上还是文字，可设置fontSize、color

图标默认托管在<http://iconfont.cn/>；[本地部署链接](https://github.com/ant-design/antd-init/tree/master/examples/local-iconfont)。

## Layout

### Grid-栅格

区块布局，使用Row、Col划分栅格。Row 24格。

**普通布局**

Row属性

gutter，栅格间隔；推荐值：16+8n

Col属性

span，栅格格数；

offset，右偏格数；

pull/push，左移/右移栅格；可能导致重叠。

**flex布局**

Row属性

type，布局模式；值：flex

justify，水平排列方式；值：start、center、end、space-arround、space-between

align，垂直对齐方式；值：top、middle、bottom

Col属性

span，栅格格数；

order，栅格顺序；

**响应式布局**

预设6个响应尺寸：xs、sm、md、lg、xl、xxl。

span、push/pull、offset可内嵌到响应式中。

### Layout-布局

页面级布局。布局解析：从上到下、从左至右，依次输出。

Layout组件，可放置于任何父容器中。

Header、Slider、Content、Footer组件，只能放置于Layout容器中。

通过style样式，可固定Header、Slider。

**Slider收缩**

默认收缩

<Slider collapsible collapsed={bool} onCollapse={this.handler}>...</Slider>

collapsed为true收缩，onCollapse绑定收缩标签点击事件

自定义收缩

<Slider collapsible collapsed={bool} trigger={null}>...</Slider>

trigger={null}，隐藏默认设定；而后可通过其他事件改编bool。

**Slider响应式布局**

<Slider breakpoint=”lg” collapsedWidth=”0”>...</Slider>

breakpoint(xs、sm、md、lg、xl、xxl)；collapsedWidth收缩宽度，0时出现特殊trigger。

## Navigation

### Affix固钉

将子元素钉在可视范围。

offsetTop/offsetBottom，距离顶部/底部指定偏移量后触发，number。

target，设置Affix需要监听滚动事件的元素；值:()=>dom，默认window。

onChange，固定状态改变的回调函数；值：Function(affixed)

Affix内的元素不要使用绝对定位，可对Affix设置绝对定位来替代。

### Breadcrumb面包屑

显示当前页面在系统层级结构中的位置，并能向上返回。

<Breadcrumb>+<Breadcrumb.Item>

separator，分隔符；值：string|ReactNode，默认”/”。

params，路由参数；值：object

routes，路由信息；值：object[]

itemRender，自定义链接函数；值：(route,param,routes,params)=>ReactNode

与react-router2配合？？？

### Dropdown下拉菜单

向下弹出列表。

overlay，弹出菜单；值：Menu。

placement，弹出位置；bottomLeft、bottomCenter、bottomRight、topLeft、topCenter、topRight。

trigger，触发方式；值：Array<click | hover | contextMenu>

disabled，禁用

visible，false隐藏菜单；默认点击菜单、移开鼠标后隐藏。

<Dropdown.Button>，单独使用，类似<Dropdown>；带按钮的下拉菜单。

<Dropdown.Button overlay={menu} onButtonClick={handler}>name</Dropdown.Button>

### Menu导航菜单

为页面和功能提供导航的菜单列表。

<Menu>、<Menu.Item>、<SubMenu>、<MenuItemGroup>

**Menu属性**

mode，排布方式；值：vertical、horizontal、inline(内嵌)

inlineCollapsed，inline时收缩；值：true收缩，false展开

onClick，点击事件回调函数

onOpenChange，SubMenu展开关闭回调函数。

selectedKeys/defaultSelectedKeys，选中项；值：string[]

openKeys/defaultOpenKeys，展开的子菜单；值：string[]

**SubMenu属性**

title，子菜单名；值：string | ReactNode

disabled，禁用

onTitleClick，点击子菜单名回调函数

**MenuItemGroup属性**

title，菜单元组名

**MenuItem属性**

disabled，禁用

### Pagination分页

采用分页形式加载长列表，每次只加载一页。

total，总条目数

onChange，页码改变回调函数

current/defaultCurrent，当前页

pageSize/defaultPageSize，每页条目数，默认10

showSizeChanger，展示改变pageSize组件；配合pageSizeOptions。

onShowSizeChange，pageSize改变回调函数

showQuickJumper，展示快速跳转组件

size，大小；值：不设，small(小尺寸)

simple，简单分页；值：boolean

showTotal，展示数据总量、当前数据信息。值：Function(total,range)

### Step步骤条

引导用户按流程完成任务的导航条。

<Steps>+<Step>

**Steps属性**

current，当前步骤，从0开始；可被Step的status覆盖。

status，当前步骤状态；值：wait、process(默认)、finish、error

size，大小；值：不设，small(迷你)

direction，方向；值：vertical、horizontal

**Step属性**

title，标题

icon，图标

description，描述

status，状态；不指定时使用Steps状态。

## Data Entry

### AutoComplete自动完成

输入框自动完成功能。配合ajax使用。

可包裹其他输入组件，实现自定义输入组件。

<AutoComplete>、<AutoComplete.OptGroup>、<AutoComplete.Option>

**AutoComplete属性**

dataSource，自动完成的数据源。

children，自动完成的数据源。

onSelect，选中条目回调函数

onSearch，搜索补全项回调函数

onChange，条目选中、输入变化回调函数

placeholder，输入框提示

allowClear，是否可清除

### Cascader级联选择

级联选择框。

options，数据源

onChange，选择回调函数

allowClear，是否可清除

placeholder，提示

defaultValue，默认选项；值：[]，多级选项每一级的值

expandTrigger，次级菜单展开方式；值：hover、click

size，大小；值：large、不设、small

changeOnSelect，是否允许选择父级；默认只能选择最底层的子级

loadData，动态加载选项；值：(opt)=>void。不能与showSearch一起使用。

showSearch，展示搜索框；值：Boolean、对象。

fieldNames，自定义字段名；值：{label: name1, value: name2, children: name3}

### Checkbox多选框

<Checkbox>+<Checkbox.Group>

**Checkbox属性**

checked/defaultChecked，选中。

onChange，回调函数。

indeterminate，设置indeterminate状态，即半勾状态；值：bool。

disabled，禁用

**CheckBox.Group属性**

options，可选项。

除indeterminate外，其他与Checkbox一致。

### DatePicker日期选择框

<DatePicker>、<MonthPicker>、<RangePicker>、<WeekPicker>(值为数组)

onChange，回调函数；

placeholder，提示；

allowClear，是否可清除

format，格式化；如：YYYY/MM/DD，YYYY/MM

size，大小；值：large，default，small

showTime，时间选择功能。

onOk，点击确定按钮的回调函数；showTime时才有确定按钮。

disabled，禁用

disabledDate，禁用日期

disabledTime，禁用时间；需与showTime联用。

### Form表单

具有数据收集、检验和提交功能的表单。包含单选、复选、下拉框、输入框等元素。

<Form>+<Form.Item>

**Form属性**

layout，布局；值：horizontal、vertical、inline

onSubmit，提交回调函数

**Form.Item属性**

validateStatus，

help，

### Input输入框

<Input>、<InputGroup>、<Input.Search>、<Input.TextArea>

**Input属性**

placeholder，提示

value/defaultValue，值

disabled，禁用

size，大小；值：large、default、small

addonBefore/addonAfter，前置/后置标签；

prefix/suffix，前缀/后缀图标；

type，input类型，同原生input标签；使用Input.TextArea替代type=’textarea’

onPressEnter，回车回调函数

**InputGroup属性**

compact，紧凑模式

size，其内所有input大小

**Input.Search属性**

size，大小；

placeholder，提示；

onSearch，搜索回调函数

enterButton，是否有确认按钮；Boolean、ReactNode

**Input.TextArea属性**

rows，行数；

placeholder，提示

autosize，自适应高度；例：{minRows: 2, maxRows: 6}，指定最小最大行数

onPressEnter，回车回调函数

### InputNumber数字输入框

min/max，最小值/最大值

value/defaultValue，值

onChange，回调函数

size，大小

step，点击图标每次改变步数

precision，精度

formatter，格式化，和parser搭配使用。

parser，指定从formatter转回数字的方法

### **Mention提及**

提及组件。

### Rate评分

allowHalf，支持半星

value/defaultValue，值

count，星总数；默认5

disabled，禁用、只读；

character，自定义图标

allowClear，支持清除

onChange，改变时回调函数

### Radio单选框

<Radio>、<Radio.Group>、<Radio.Button>

**Radio、Radio.Button属性**

checked/defaultChecked，选中

disabled，禁用

value，值

**Radio.Group属性**

value/defaultValue，值

disabled，禁用

onChange，回调函数

options，子元素集；值：string[] | Array<{label: string, value: string, disabled?: boolean}>

name，设置所有子项的name属性；浏览器会将其真正看做一组。

size，大小；值：large，small

buttonStyle，Radio.Button选中时样式；值：outline、solid

### Select选择器

下拉选择器。

<Select>+<Select.Option>、<Select.OptGroup>

**Select属性**

value/defaultValue，值

disabled，禁用

onChange，回调函数

placeholder，提示

showSearch，展示搜索

filterOption，筛选函数

optionFilterProp，搜索时过滤的Select.Option属性；值：value

mode，模式；值：multiple、tags

size，大小；值：large、default、small

labelInValue，将label封装进value；返回{key: value, label: lable}

**Select.Option属性**

key，value

**Select.OptGroup属性**

key，label

### Slider滑动输入条

滑动型输入器，展示当前值和可选范围。

value/defaultValue，值

onChange，回调函数

step，步进量；需被max-min整除；当marks不为空，可设为null，只能拖到marks上。

dots，只能拖到marks上。

min/max，最小值/最大值

disabled，禁用

range，双滑块模式

tipFormatter，格式化提示；null时无提示

included，marks不为空有效；值：true包含，false并列

marks，刻度标记

vertical，垂直；值：Boolean

### Switch开关

开关选择器。

checked/defaultChecked，值

onChange，回调函数

disabled，禁用

checkedChildren/unCheckedChildren，选中/未选中时内嵌图标

size，大小；值：default、small

loading，加载中、开关操作执行中；值：boolean

### TreeSelect树选择

树型选择控件。

<TreeSelect>+<TreeSelect.TreeNode>

**TreeSelect属性**

value/defaultValue，值

onChange，回调函数

placeholder，提示

showSearch，显示搜索框

onSearch，搜索框变化回调

multiple，多选

labelInValue，把label包装进value。

treeData，TreeNode数据；值：array<{value, title, children, disabled...}>

treeDefaultExpandAll/treeDefaultExpandKeys，默认展开项

**TreeSelect.TreeNode属性**

建议使用treeData。

key，value，title，isLeaf，disabled

### TimePicker时间选择框

value/defaultValue，值；moment

placeholder，提示

onChange，回调函数；值：Function(time,timeString)

disabled/disabledHour/disabledMinute/disabledSecond，禁用

defaultOpenValue，无value/defaultValue时，设置选择框初始值。moment

size，大小；值：large，small

format，时间格式；默认：HH:mm:ss

hourStep/minuteStep/secondStep，步长；值：number

addon，选择框底部自定义内容；值：function

use12Hours，12小时制，默认format--h:mm:ss a

allowEmpty，显示清除按钮

clearText，清除按钮内容

open，选择框是否打开

onOpenChange，选择框打开状态回调函数

### Transfer穿梭框

双栏穿梭选择框。

dataSource，数据源；值：TransferItem[]，{key, title, description?, disabled?}

titles，标题集合

targetKeys，选中数据；值：string[]

selectedKeys，设置哪些项应该被选中；

onChange，变化回调函数

onSelectChange，选中回调函数

onScroll，滚动回调函数

render，渲染条目。值：function(record)，返回ReactElement，或{value: ReactElement, label: title}

showSearch，展示搜索框

searchPlaceholder，搜索框提示

filterOption，搜索过滤；值：function(inputValue, option)

### Upload上传

文件选择上传和拖拽上传控件。

name，

action，

header，

onChange，

## Data Display

### Avatar头像

用来代表用户或事物，支持图片、图标、字符。

size，大小；值：large，default，small

shape，形状；值：circle，square

icon，图标；值：参考Icon组件

src，图片类头像链接。文字类头像，<Avatar>User</Avatar>

alt，图片加载失败时替代文字

### Badge徽标数

图标右上角的圆形徽标数字。

count，展示数字，0时隐藏

showZero，显示0

overflowCount，展示封顶的数字值

dot，不展示数字，只展示一个点

status，设置Badge为状态点；值：success，process，default，error，warning

text，status有效时的状态文本。

title，鼠标悬浮时的内容。

可通过包裹<a>标签，实现可点击。

可不包裹其他组件，单独使用。<Badge status={‘error’}/>

### Calendar日历

按日历形式展示数据的容器。

value/defaultValue，默认日期；moment

onSelect，选择回调

onChange，变化回调

onPanelChange，面板变化回调

dateCellRender/dateFullCellRender，自定义日期单元格

monthCellRender/monthFullCellRender，自定义月单元格

fullscreen，全屏显示；默认true。需放置在有限容器中时设为false

mode，初始模式；值：month、year。

### Card卡片

通用卡片容器。

<Card>、<Card.Meta>、<Card.Grid>

**Card属性**

title，标题

extra，右上角额外操作区

bordered，边框

hoverable，鼠标移过时悬浮

cover，封面

loading，加载中

tabList，页签列表；值：Array<[{key: string, tab: ReactNode}]>

activeTabKey/defaultActiveTabKey，页签默认值

onTabChange，页签回调函数

actions，卡片底部展示内容

**Card.Meta属性**

title，description，avatar

**Card.Grid，网格内嵌卡片**

### Carousel走马灯

旋转木马，一组轮播的区域。

autodisplay，自动播放

vertical，垂直显示

effect，动画效果函数；值：fade，scrollx

dots，是否显示面板指示点；默认true

afterChange/beforeChange，切换面板回调函数

### Collapse折叠面板

可以折叠/展开的内容区域。

<Collapse>+<Collapse.Panel>

**Collapse属性**

activeKey/defaultActiceKey，

onChange，切换面板的回调函数。

accordion，手风琴样式，同一时间只有一个打开。

borderd，边界；值：boolean，默认true，有边界。

**Collapse.Panel属性**

header，面板头内容。

key，

showArrow，展示箭头；值：boolean，默认true

disabled，禁用

包裹其他展示组件，作为内容。

### List列表

通用列表。

<List>+<List.Item>+<List.Item.Meta>

**List属性**

dataSource，列表数据源

renderItem，列表节点输出

bordered，展示边框

header/footer，列表头部/底部

size，大小；值：large，small

itemLayout，列表布局；值：horizontal，vertical

loading，加载中

loadMore，加载更多

pagination，分页；值：{onChange: function(pageSize), pageSize: number}

grid，列表栅格配置；

**List.Item属性**

actions，列表操作组

extra，额外内容

**List.Item.Meta属性**

avatar，title，description

### Popover气泡卡片

点击、鼠标移入时，弹出气泡式的卡片浮层。包裹其他元素。

title，标题

content，内容

trigger，触发方式；值：hover，focus，click，contextMenu

placement，弹出位置；值：(top,left,right,bottom) \* (Left,’’,Right)，共12个位置。

visible，可见

onVisibleChange，可见改变回调函数

arrowPointAtCenter，箭头指向子元素中间

确保Popover子元素能接收onMouseEnter、onMouseLeave、onFocus、onClick 事件。

### Tooltip文字提示

简单的文字提示气泡框。

无content属性，其他属性与Popover相同。

### Table表格

展示行列数据。

<Table>、<Table.ColumnGroup>+<Table.Column>(columns的语法糖)

dataSource，数据源；值：Array<{key, ...attrs}>

columns，列定义；值：Array<{key, title, dataIndex, render?}>

rowSelection，

### Tabs标签页

选项卡切换组件。

<Tabs>+<Tabs.TabPane>

**Tabs属性**

activeKey/defaultActiveKey，

onChange，切换页签回调

tabPosition，页签位置；值：top，bottom，left，right

tabBarExtraContent，tab bar上额外元素

size，大小；值：large，default，small

type，页签类型；值：line，card，editable-card

onEdit，增加删除页签回调；值：function(targetKey, action)，action：add、remove

**Tabs.TabPane属性**

tab，页签头内容

key，key

disabled，禁用

closable，可关闭，只有卡片式页签支持。

### Tag标签

进行标记和分类的小标签。包裹其他组件。

<Tag>、<Tag.CheckableTag>

**Tag属性**

closable，可关闭

onClose/afterClose，

color，背景色

onClick，点击回调函数

visible，可见

**Tag.CheckableTag属性**

只支持受控组件用法。

checked，选中；

onChange，回调函数

### Timeline时间轴

垂直展示的时间流信息。

<Timeline>+<Timeline.Item>

**Timeline属性**

pending，指定最后一个幽灵节点是否存在或内容；值：boolean | string | ReactNode

pendingDot，幽灵节点时间图点；

reverse，节点排序；值：boolean

mode，时间轴和内容相对位置；值：left，right，alternate

**Timeline.Item属性**

color，圆圈颜色；

dot，时间图点

### Tree树型控件

<Tree>+<Tree.TreeNode>

**Tree属性**

**Tree.TreeNode属性**

title，key，disabled，

## Feedback

### Alert警告提示

message，提示

description，提示描述

type，类型；值：success，info，warning，error

showIcon，展示类型图标；

closable，可关闭

closeText，替换默认的关闭图标

onClose/afterClose，关闭的回调函数

banner，是否用作顶部公告

### Drawer抽屉

遮罩层，从左侧/右侧弹出。包裹抽屉内容。

title，标题

placement，抽屉弹出位置；值：left，right

closable，显示关闭图标；默认true

maskClosable，点击遮罩层关闭；默认true

mask，显示遮罩层；默认true

onClose，关闭回调函数

visible，可见

通过抽屉嵌套，可实现多层抽屉；用以解决多分支任务的复杂状况。

### Modal对话框

模态对话框。包裹其他组件。

title，标题

visible，可见

closable，显示关闭图标

onOK/onCancel，确定/取消回调函数

confirmLoading，确定按钮loading

mask，显示遮罩层

maskClosable，点击遮罩层关闭

footer，自定义页脚；不需要时设为null。

destroyOnClose，关闭时清空；默认false，<Modal/>默认关闭时不清空子元素。

Modal.confirm()，快速弹出对话框；

Modal.info()，Modal.success()，Modal.error()，Modal.warning()；只有一个关闭按钮的对话框。

参数{title, content, onOk?...}

以上函数调用后，会返回一个引用，可通过引用关闭弹窗。

const ref = Modal.info(); ref.destroy();

### Message全局提示

全局展示操作反馈信息。

import {message} from ‘antd’;

message.success()，message.error()，message.info()，message.warning()/warn()，message.loading()；提示函数，then接口返回值为Promise；参数：(content, [duration], onClose)

message.loading('Action in progress..', 2.5)

.then(() => message.success('Loading finished', 2.5))

.then(() => message.info('Loading finished is finished', 2.5));

message.destroy()；销毁

### Notification通知提示框

全局展示通知提示信息。

import {notification} from ‘antd’;

notification.info()，notification.success()，notification.error()，notification.warn()/warning()，notification.open()；打开通知提示框；参数：{message, description, key...}；两次调用方法key参数相同时，为更新提示框。

notification.close(key: string)；关闭

notification.destroy()；销毁

### Progress进度条

展示操作的当前进度。

percent，百分比

successPercent，已完成的分段百分比

status，状态；值：active，success，exception

showInfo，显示进度信息；默认true

type，类型；值：line，circle，dashboard

size，大小；值：small

width，circle/dashboard控制大小

format，内容模板函数

### Popconfirm气泡确认框

点击元素，弹出气泡式的确认框。包裹组件。

title，弹框内容

onOK/onCancel，确认/取消回调函数

okText/cancelText，确认/取消按钮文字

placement，弹出位置；值：(top, bottom, left, right) \* (‘’, Left,Right)

arrowPointAtCenter，箭头指向元素中心；

visible，可见

onVisibleChange，改变可见性回调函数

icon，图标；值：ReactNode

Popconfirm的子元素必须能接收onMouseEnter、onMouseLeave、onFocus、onClick 事件。

### Spin加载中

用于页面和区块的加载中状态。

size，大小；值：large，small

spinning，加载状态

tip，加载时的自定义文案描述

delay，延迟加载时间，ms；

indicator，加载指示符；值：ReactElement

Spin.setDefaultIndicator(ReactElement)；设置全局默认元素

## Other

### Anchor锚点

用于跳转到页面指定位置。

<Anchor>+<Anchor.Link>

**Anchor属性**

affix，固定模式

**Anchor.Link属性**

href，锚点链接；值：string

title，内容；值：string | ReactNode

### BackTop回到顶部

返回页面顶部的操作按钮。

<BackTop/>；默认按钮样式

<BackTop> <div> TOP</div> </BackTop>；自定义按钮样式，按钮内容为TOP；限制宽高：40\*40。

### Divider分割线

区隔内容的分割线。

dashed，虚线；

orientation，分割线子元素位置；值：left，right，center

type，类型；值：horizontal，vertical