高性能表单控件，自带数据域管理。包含数据录入、校验以及对应样式。

何时使用

- 用于创建一个实体或收集信息。
- 需要对输入的数据类型进行校验时。

API

Form

参数	说明	类型	默认值	版本
colon	配置 Form.Item 的 `colon` 的默认值。表示是否显示 label 后面的冒号 (只有在属性 layout 为 horizontal 时有效)	boolean	true	
component	设置 Form 渲染元素，为 `false` 则不创建 DOM 节点	ComponentType | false	form	
fields	通过状态管理（如 redux）控制表单字段，如非强需求不推荐使用。查看示例	FieldData[]	-	
form	经 `Form.useForm()` 创建的 form 控制实例，不提供时会自动创建	FormInstance	-	
initialValues	表单默认值，只有初始化以及重置时生效	object	-	
labelAlign	label 标签的文本对齐方式	`left` | `right`	`right`	
labelWrap	label 标签的文本换行方式	boolean	false	4.18.0
labelCol	label 标签布局，同 `<Col>` 组件，设置 `span` `offset` 值，如 `{span: 3, offset: 12}` 或 `sm: {span: 3, offset: 12}`	object	-	
layout	表单布局	`horizontal` | `vertical` | `inline`	`horizontal`	
name	表单名称，会作为表单字段 `id` 前缀使用	string	-	
preserve	当字段被删除时保留字段值	boolean	true	4.4.0
requiredMark	必选样式，可以切换为必选或者可选展示样式。此为 Form 配置，Form.Item 无法单独配置	boolean | `optional`	true	4.6.0
scrollToFirstError	提交失败自动滚动到第一个错误字段	boolean | Options	false	
size	设置字段组件的尺寸（仅限 antd	`small` | `middle` | `large`	-	

	组件)			
validateMessages	验证提示模板，说明见下	ValidateMessages	-	
validateTrigger	统一设置字段触发验证的时机	string | string[]	`onChange`	4.3.0
wrapperCol	需要为输入控件设置布局样式时，使用该属性，用法同 labelCol	object	-	
onFieldsChange	字段更新时触发回调事件	function(changedFields, allFields)	-	
onFinish	提交表单且数据验证成功后回调事件	function(values)	-	
onFinishFailed	提交表单且数据验证失败后回调事件	function({ values, errorFields, outOfDate })	-	
onValuesChange	字段值更新时触发回调事件	function(changedValues, allValues)	-	

validateMessages

Form 为验证提供了默认的错误提示信息，你可以通过配置 `validateMessages` 属性，修改对应的提示模板。一种常见的使用方式，是配置国际化提示信息：

```
const validateMessages = {
  required: "'${name}' 是必选字段",
  // ...
};

<Form validateMessages={validateMessages} />;
```

此外，ConfigProvider 也提供了全局化配置方案，允许统一配置错误提示模板：

```
const validateMessages = {
  required: "'${name}' 是必选字段",
  // ...
};

<ConfigProvider form={{ validateMessages }}>
  <Form />
</ConfigProvider>;
```

Form.Item

表单字段组件，用于数据双向绑定、校验、布局等。

参数	说明	类型	默认值	版本
colon	配合 `label` 属性使用，表示是否显示 `label` 后面的冒号	boolean	true	

dependencies	设置依赖字段，说明见下	NamePath[]	-	
extra	额外的提示信息，和 `help` 类似，当需要错误信息和提示文案同时出现时，可以使用这个。	ReactNode	-	
getValueFromEvent	设置如何将 event 的值转换成字段值	(..args: any[]) => any	-	
getValueProps	为子元素添加额外的属性	(value: any) => any	-	4.2.0
hasFeedback	配合 `validateStatus` 属性使用，展示校验状态图标，建议只配合 Input 组件使用	boolean	false	
help	提示信息，如不设置，则会根据校验规则自动生成	ReactNode	-	
hidden	是否隐藏字段（依然会收集和校验字段）	boolean	false	4.4.0
htmlFor	设置子元素 label `htmlFor` 属性	string	-	
initialValue	设置子元素默认值，如果与 Form 的 `initialValues` 冲突则以 Form 为准	string	-	4.2.0
label	`label` 标签的文本	ReactNode	-	
labelAlign	标签文本对齐方式	`left` | `right`	`right`	
labelCol	`label` 标签布局，同 `<Col>` 组件，设置 `span` `offset` 值，如 `{span: 3, offset: 12}` 或 `sm: {span: 3, offset: 12}`。你可以通过 Form 的 `labelCol` 进行统一设置，，不会作用于嵌套 Item。当和 Form 同时设置时，以 Item 为准	object	-	
messageVariables	默认验证字段的信息	Record<string, string>	-	4.7.0
name	字段名，支持数组	NamePath	-	
normalize	组件获取值后进行转换，再放入 Form 中。不支持异步	(value, prevValue, prevValues) => any	-	
noStyle	为 `true` 时不带样式，作为纯字段控件使用	boolean	false	
preserve	当字段被删除时保留字段值	boolean	true	4.4.0
required	必填样式设置。如不设置，则会根据校验规则自动生成	boolean	false	
rules	校验规则，设置字段的校验逻辑。点击此处查看示例	Rule[]	-	

shouldUpdate	自定义字段更新逻辑，说明见下	boolean | (prevValue, curValue) => boolean	false	
tooltip	配置提示信息	ReactNode | TooltipProps & { icon: ReactNode }	-	4.7.0
trigger	设置收集字段值变更的时机。点击此处查看示例	string	`onChange`	
validateFirst	当某一规则校验不通过时，是否停止剩下的规则的校验。设置 `parallel` 时会并行校验	boolean | `parallel`	false	`parallel`: 4.5.0
validateStatus	校验状态，如不设置，则会根据校验规则自动生成，可选：'success' 'warning' 'error' 'validating'	string	-	
validateTrigger	设置字段校验的时机	string | string[]	`onChange`	
valuePropName	子节点的值的属性，如 Switch 的是 'checked'。该属性为 `getValueProps` 的封装，自定义 `getValueProps` 后会失效	string	`value`	
wrapperCol	需要为输入控件设置布局样式时，使用该属性，用法同 `labelCol`。你可以通过 Form 的 `wrapperCol` 进行统一设置，不会作用于嵌套 Item。当和 Form 同时设置时，以 Item 为准	object	-	

被设置了 `name` 属性的 `Form.Item` 包装的控件，表单控件会自动添加 `value`（或 `valuePropName` 指定的其他属性） `onChange`（或 `trigger` 指定的其他属性），数据同步将被 Form 接管，这会导致以下结果：

1. 你**不再需要也不应该**用 `onChange` 来做数据收集同步（你可以使用 Form 的 `onValuesChange`），但还是可以继续监听 `onChange` 事件。
2. 你不能用控件的 `value` 或 `defaultValue` 等属性来设置表单域的值，默认值可以用 Form 里的 `initialValues` 来设置。注意 `initialValues` 不能被 `setState` 动态更新，你需要用 `setFieldsValue` 来更新。
3. 你不应该用 `setState`，可以使用 `form.setFieldsValue` 来动态改变表单值。

dependencies

当字段间存在依赖关系时使用。如果一个字段设置了 `dependencies` 属性。那么它所依赖的字段更新时，该字段将自动触发更新与校验。一种常见的场景，就是注册用户表单的“密码”与“确认密码”字段。“确认密码”校验依赖于“密码”字段，设置 `dependencies` 后，“密码”字段更新会重新触发“校验密码”的校验逻辑。你可以参考具体例子。

`dependencies` 不应和 `shouldUpdate` 一起使用，因为这可能带来更新逻辑的混乱。

从 `4.5.0` 版本开始，`dependencies` 支持使用 render props 类型 children 的 `Form.Item`。

shouldUpdate

Form 通过增量更新方式，只更新被修改的字段相关组件以达到性能优化目的。大部分场景下，你只需要编写代码或者与 `dependencies` 属性配合校验即可。而在某些特定场景，例如修改某个字段值后出现新的字段选项、或者纯粹希望表单任意变化都对某一个区域进行渲染。你可以通过 `shouldUpdate` 修改 Form.Item 的更新逻辑。

当 `shouldUpdate` 为 `true` 时，Form 的任意变化都会使该 Form.Item 重新渲染。这对于自定义渲染一些区域十分有帮助：

```
<Form.Item shouldUpdate>
  {() => {
    return <pre>{JSON.stringify(form.getFieldsValue(), null, 2)}</pre>;
  }}
</Form.Item>
```

你可以参考示例查看具体使用场景。

当 `shouldUpdate` 为方法时，表单的每次数值更新都会调用该方法，提供原先的值与当前的值以供你比较是否需要更新。这对于是否根据值来渲染额外字段十分有帮助：

```
<Form.Item shouldUpdate={(prevValues, curValues) => prevValues.additional !==
curValues.additional}>
  {() => {
    return (
      <Form.Item name="other">
        <Input />
      </Form.Item>
    );
  }}
</Form.Item>
```

你可以参考示例查看具体使用场景。

messageVariables

你可以通过 `messageVariables` 修改 Form.Item 的默认验证信息。

```
<Form>
  <Form.Item messageVariables={{ another: 'good' }} label="user">
    <Input />
  </Form.Item>
  <Form.Item messageVariables={{ label: 'good' }} label={<span>user</span>}>
    <Input />
  </Form.Item>
</Form>
```

Form.List

为字段提供数组化管理。

参数	说明	类型	默认值	版本
children	渲染函数	(fields: Field[], operation: { add, remove, move }, meta: { errors }) => React.ReactNode	-	
initialValue	设置子元素默认值，如果与 Form 的 `initialValues` 冲突则以 Form 为准	any[]	-	4.9.0
name	字段名，支持数组	NamePath	-	
rules	校验规则，仅支持自定义规则。需要配合 ErrorList 一同使用。	{ validator, message }[]	-	4.7.0

```
<Form.List>
  {fields =>
    fields.map(field => (
      <Form.Item {...field}>
        <Input />
      </Form.Item>
    ))
  }
</Form.List>
```

注意：Form.List 下的字段不应该配置 `initialValue`，你始终应该通过 Form.List 的 `initialValue` 或者 Form 的 `initialValues` 来配置。

operation

Form.List 渲染表单相关操作函数。

参数	说明	类型	默认值	版本
add	新增表单项	(defaultValue?: any, insertIndex?: number) => void	insertIndex	4.6.0
move	移动表单项	(from: number, to: number) => void	-	
remove	删除表单项	(index: number | number[]) => void	number[]	4.5.0

Form.ErrorList

4.7.0 新增。错误展示组件，仅限配合 Form.List 的 rules 一同使用。参考示例。

参数	说明	类型	默认值
errors	错误列表	ReactNode[]	-

Form.Provider

提供表单间联动功能，其下设置 `name` 的 Form 更新时，会自动触发对应事件。查看示例。

参数	说明	类型	默认值
onFormChange	子表单字段更新时触发	function(formName: string, info: { changedFields, forms })	-
onFormFinish	子表单提交时触发	function(formName: string, info: { values, forms })	-

```
<Form.Provider
  onFormFinish={name => {
    if (name === 'form1') {
      // Do something...
    }
  }}
>
  <Form name="form1">...</Form>
  <Form name="form2">...</Form>
</Form.Provider>
```

FormInstance

名称	说明	类型	版本
getFieldError	获取对应字段名的错误信息	(name: NamePath) => string[]	
getFieldInstance	获取对应字段实例	(name: NamePath) => any	4.4.0
getFieldsError	获取一组字段名对应的错误信息，返回为数组形式	(nameList?: NamePath[]) => FieldError[]	
getFieldsValue	获取一组字段名对应的值，会按照对应结构返回。默认返回现存字段值，当调用 `getFieldsValue(true)` 时返回所有值	(nameList?: NamePath[], filterFunc?: (meta: { touched: boolean, validating: boolean }) => boolean) => any	
getFieldValue	获取对应字段名的值	(name: NamePath) => any	
isFieldsTouched	检查一组字段是否被用户操作过，`allTouched` 为 `true` 时检查是否所有字段都被操作过	(nameList?: NamePath[], allTouched?: boolean) => boolean	
isFieldTouched	检查对应字段是否被用户操作过	(name: NamePath) => boolean	
isFieldValidating	检查对应字段是否正在校验	(name: NamePath) => boolean	
resetFields	重置一组字段到 `initialValues`	(fields?: NamePath[]) => void	
scrollToField	滚动到对应字段位置	(name: NamePath, options: ScrollOptions) => void	
setFields	设置一组字段状态	(fields: FieldData[]) => void	
setFieldsValue	设置表单的值（该值将直接传入 form	(values) => void	

	store 中。如果你不希望传入对象被修改，请克隆后传入)		
submit	提交表单，与点击 `submit` 按钮效果相同	() => void	
validateFields	触发表单验证	(nameList?: NamePath[]) => Promise	

validateFields 返回示例

```
validateFields()
  .then(values => {
    /*
  values:
    {
      username: 'username',
      password: 'password',
    }
  */
  })
  .catch(errorInfo => {
    /*
    errorInfo:
      {
        values: {
          username: 'username',
          password: 'password',
        },
        errorFields: [
          { name: ['password'], errors: ['Please input your Password!'] },
        ],
        outOfDate: false,
      }
    */
  });
```

Interface

NamePath

```
string | number | (string | number)[]
```

FieldData

名称	说明	类型
errors	错误信息	string[]
name	字段名称	NamePath[]
touched	是否被用户操作过	boolean
validating	是否正在校验	boolean

value	字段对应值	any

Rule

Rule 支持接收 object 进行配置，也支持 function 来动态获取 form 的数据：

```
type Rule = RuleConfig | ((form: FormInstance) => RuleConfig);
```

名称	说明	类型	版本
defaultField	仅在 `type` 为 `array` 类型时有效，用于指定数组元素的校验规则	rule	
enum	是否匹配枚举中的值（需要将 `type` 设置为 `enum`）	any[]	
fields	仅在 `type` 为 `array` 或 `object` 类型时有效，用于指定子元素的校验规则	Record<string, rule>	
len	string 类型时为字符串长度；number 类型时为确定数字；array 类型时为数组长度	number	
max	必须设置 `type`：string 类型为字符串最大长度；number 类型时为最大值；array 类型时为数组最大长度	number	
message	错误信息，不设置时会通过模板自动生成	string	
min	必须设置 `type`：string 类型为字符串最小长度；number 类型时为最小值；array 类型时为数组最小长度	number	
pattern	正则表达式匹配	RegExp	
required	是否为必选字段	boolean	
transform	将字段值转换成目标值后进行校验	(value) => any	
type	类型，常见有 `string` |`number` |`boolean` |`url` | `email`。更多请参考此处	string	
validateTrigger	设置触发验证时机，必须是 Form.Item 的 `validateTrigger` 的子集	string | string[]	
validator	自定义校验，接收 Promise 作为返回值。示例参考	(rule, value) => Promise	
warningOnly	仅警告，不阻塞表单提交	boolean	4.17.0
whitespace	如果字段仅包含空格则校验不通过，只在 `type: 'string'` 时生效	boolean	

从 v3 升级到 v4

如果你是 antd v3 的用户，你可以参考迁移示例。

FAQ

自定义 validator 没有效果

这是由于你的 `validator` 有错误导致 `callback` 没有执行到。你可以选择通过 `async` 返回一个 promise 或者使用 `try...catch` 进行错误捕获：

```
validator: async (rule, value) => {
  throw new Error('Something wrong!');
}

// or

validator(rule, value, callback) => {
  try {
    throw new Error('Something wrong!');
  } catch (err) {
    callback(err);
  }
}
```

name 为数组时的转换规则?

当 `name` 为数组时，会按照顺序填充路径。当存在数字且 form store 中没有该字段时会自动转变成数组。因而如果需要数组为 key 时请使用 string 如：`['1', 'name']`。

为何在 Modal 中调用 form 控制台会报错?

> *Warning: Instance created by `useForm` is not connect to any Form element. Forget to pass `form` prop?*

这是因为你在调用 form 方法时，Modal 还未初始化导致 form 没有关联任何 Form 组件。你可以通过给 Modal 设置 `forceRender` 将其预渲染。示例点击此处。

为什么 Form.Item 下的子组件 `defaultValue` 不生效?

当你为 Form.Item 设置 `name` 属性后，子组件会转为受控模式。因而 `defaultValue` 不会生效。你需要在 Form 上通过 `initialValues` 设置默认值。

为什么第一次调用 `ref` 的 From 为空?

`ref` 仅在节点被加载时才会被赋值，请参考 React 官方文档：https://reactjs.org/docs/refs-and-the-dom.html#accessing-refs

为什么 `resetFields` 会重新 mount 组件?

`resetFields` 会重置整个 Field，因而其子组件也会重新 mount 从而消除自定义组件可能存在的副作用（例如异步数据、状态等等）。

Form 的 initialValues 与 Item 的 initialValue 区别?

在大部分场景下，我们总是推荐优先使用 Form 的 `initialValues`。只有存在动态字段时你才应该使用 Item 的 `initialValue`。默认值遵循以下规则：

1. Form 的 `initialValues` 拥有最高优先级

2. Field 的 `initialValue` 次之 *. 多个同 `name` Item 都设置 `initialValue` 时，则 Item 的 `initialValue` 不生效

为什么字段设置 `rules` 后更改值 `onFieldsChange` 会触发三次?

字段除了本身的值变化外，校验也是其状态之一。因而在触发字段变化会经历以下几个阶段：

1. Trigger value change
2. Rule validating
3. Rule validated

在触发过程中，调用 `isFieldValidating` 会经历 `false` > `true` > `false` 的变化过程。

为什么 Form.List 不支持 `label` 还需要使用 ErrorList 展示错误?

Form.List 本身是 renderProps，内部样式非常自由。因而默认配置 `label` 和 `error` 节点很难与之配合。如果你需要 antd 样式的 `label`，可以通过外部包裹 Form.Item 来实现。

为什么 Form.Item 的 `dependencies` 对 Form.List 下的字段没有效果?

Form.List 下的字段需要包裹 Form.List 本身的 `name`，比如：

```
<Form.List name="users">
  {fields =>
    fields.map(field => (
      <React.Fragment key={field.key}>
        <Form.Item name={[field.name, 'name']} {...someRest1} />
        <Form.Item name={[field.name, 'age']} {...someRest1} />
      </React.Fragment>
    ))
  }
</Form.List>
```

依赖则是：`['users', 0, 'name']`

为什么 `normalize` 不能是异步方法?

React 中异步更新会导致受控组件交互行为异常。当用户交互触发 `onChange` 后，通过异步改变值会导致组件 `value` 不会立刻更新，使得组件呈现假死状态。如果你需要异步触发变更，请通过自定义组件实现内部异步状态。

自定义表单控件 `scrollToFirstError` 和 `scrollToField` 失效?

类似问题：#28370 #27994

滚动依赖于表单控件元素上绑定的 `id` 字段，如果自定义控件没有将 `id` 赋到正确的元素上，这个功能将失效。你可以参考这个 codesandbox。

`setFieldsValue` 不会触发 `onFieldsChange` 和 `onValuesChange`?

是的，change 事件仅当用户交互才会触发。该设计是为了防止在 change 事件中调用 `setFieldsValue` 导致的循环问题。

有更多参考文档吗?

- 你可以阅读《antd v4 Form 使用心得》获得一些使用帮助以及建议。
- 想在 DatePicker、Switch 也使用 before、after？可以参考《如何优雅的对 Form.Item 的 children 增加 before、after》。