High performance Form component with data scope management. Including data collection, verification, and styles.

When to use

- When you need to create an instance or collect information.
- When you need to validate fields in certain rules.

API

Form

Property	Description	Type	Default	Version
colon	Configure the default value of `colon` for Form.Item. Indicates whether the colon after the label is displayed (only effective when prop layout is horizontal)	boolean	true	
component	Set the Form rendering element. Do not create a DOM node for `false`	ComponentType | false	form	
fields	Control of form fields through state management (such as redux). Not recommended for non-strong demand. View example	FieldData[]	-	
form	Form control instance created by `Form.useForm()`. Automatically created when not provided	FormInstance	-	
initialValues	Set value by Form initialization or reset	object	-	
labelAlign	The text align of label of all items	`left` | `right`	`right`	
labelWrap	whether label can be wrap	boolean	false	4.18.0
labelCol	Label layout, like `<Col>` component. Set `span` `offset` value like `{span: 3, offset: 12}` or `sm: {span: 3, offset: 12}`	object	-	
layout	Form layout	`horizontal` | `vertical` | `inline`	`horizontal`	
name	Form name. Will be the prefix of Field `id`	string	-	
preserve	Keep field value even when field	boolean	true	4.4.0

	removed			
requiredMark	Required mark style. Can use required mark or optional mark. You can not config to single Form.Item since this is a Form level config	boolean | `optional`	true	4.6.0
scrollToFirstError	Auto scroll to first failed field when submit	boolean | [Options](#)	false	
size	Set field component size (antd components only)	`small` | `middle` | `large`	-	
validateMessages	Validation prompt template, description [see below](#)	[ValidateMessages](#)	-	
validateTrigger	Config field validate trigger	string | string[]	`onChange`	4.3.0
wrapperCol	The layout for input controls, same as `labelCol`	[object](#)	-	
onFieldsChange	Trigger when field updated	function(changedFields, allFields)	-	
onFinish	Trigger after submitting the form and verifying data successfully	function(values)	-	
onFinishFailed	Trigger after submitting the form and verifying data failed	function({ values, errorFields, outOfDate })	-	
onValuesChange	Trigger when value updated	function(changedValues, allValues)	-	

validateMessages

Form provides [default verification error messages](#). You can modify the template by configuring `validateMessages` property. A common usage is to configure localization:

```
const validateMessages = {
  required: "'${name}' is required!",
  // ...
};

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

Besides, [ConfigProvider](#) also provides a global configuration scheme that allows for uniform configuration error notification templates:

```
const validateMessages = {
  required: "'${name}' is Required!",
  // ...
};
```

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

Form.Item

Form field component for data bidirectional binding, validation, layout, and so on.

Property	Description	Type	Default	Version
colon	Used with `label`, whether to display `:` after label text.	boolean	true	
dependencies	Set the dependency field. See below	NamePath[]	-	
extra	The extra prompt message. It is similar to help. Usage example: to display error message and prompt message at the same time	ReactNode	-	
getValueFromEvent	Specify how to get value from event or other onChange arguments	(..args: any[]) => any	-	
getValueProps	Additional props with sub component	(value: any) => any	-	4.2.0
hasFeedback	Used with `validateStatus`, this option specifies the validation status icon. Recommended to be used only with `Input`	boolean	false	
help	The prompt message. If not provided, the prompt message will be generated by the validation rule.	ReactNode	-	
hidden	Whether to hide Form.Item (still collect and validate value)	boolean	false	4.4.0
htmlFor	Set sub label `htmlFor`	string	-	
initialValue	Config sub default value. Form `initialValues` get higher priority when conflict	string	-	4.2.0
label	Label text	ReactNode	-	
labelAlign	The text align of label	`left` | `right`	`right`	
labelCol	The layout of label. You can set `span` `offset` to something like `{span: 3, offset: 12}` or `sm: {span: 3, offset: 12}` same as with `<Col>`. You can set	object	-	

	`labelCol` on Form which will not affect nest Item. If both exists, use Item first			
messageVariables	The default validate field info	Record<string, string>	-	4.7.0
name	Field name, support array	[NamePath]()	-	
normalize	Normalize value from component value before passing to Form instance. Do not support async	(value, prevValue, prevValues) => any	-	
noStyle	No style for `true`, used as a pure field control	boolean	false	
preserve	Keep field value even when field removed	boolean	true	4.4.0
required	Display required style. It will be generated by the validation rule	boolean	false	
rules	Rules for field validation. Click [here]() to see an example	[Rule]()[]	-	
shouldUpdate	Custom field update logic. See [below]()	boolean | (prevValue, curValue) => boolean	false	
tooltip	Config tooltip info	ReactNode | [TooltipProps & { icon: ReactNode }]()	-	4.7.0
trigger	When to collect the value of children node. Click [here]() to see an example	string	`onChange`	
validateFirst	Whether stop validate on first rule of error for this field. Will parallel validate when `parallel` cofigured	boolean | `parallel`	false	`parallel`: 4.5.0
validateStatus	The validation status. If not provided, it will be generated by validation rule. options: `success` `warning` `error` `validating`	string	-	
validateTrigger	When to validate the value of children node	string | string[]	`onChange`	
valuePropName	Props of children node, for example, the prop of Switch is 'checked'. This prop is an encapsulation of `getValueProps`, which will be invalid after customizing `getValueProps`	string	`value`	

wrapperCol	The layout for input controls, same as `labelCol`. You can set `wrapperCol` on Form which will not affect nest Item. If both exists, use Item first	[object](#)	-	

After wrapped by `Form.Item` with `name` property, `value` (or other property defined by `valuePropName`) `onChange` (or other property defined by `trigger`) props will be added to form controls, the flow of form data will be handled by Form which will cause:

1. You shouldn't use `onChange` on each form control to **collect data**(use `onValuesChange` of Form), but you can still listen to `onChange` .
2. You cannot set value for each form control via `value` or `defaultValue` prop, you should set default value with `initialValues` of Form. Note that `initialValues` cannot be updated by `setState` dynamically, you should use `setFieldsValue` in that situation.
3. You shouldn't call `setState` manually, please use `form.setFieldsValue` to change value programmatically.

dependencies

Used when there are dependencies between fields. If a field has the `dependencies` prop, this field will automatically trigger updates and validations when upstream is updated. A common scenario is a user registration form with "password" and "confirm password" fields. The "Confirm Password" validation depends on the "Password" field. After setting `dependencies` , the "Password" field update will re-trigger the validation of "Check Password". You can refer [examples](#).

`dependencies` shouldn't be used together with `shouldUpdate` , since it may result in conflicting update logic.

`dependencies` supports `Form.Item` with render props children since `4.5.0` .

shouldUpdate

Form updates only the modified field-related components for performance optimization purposes by incremental update. In most cases, you only need to write code or do validation with the [`dependencies`](#) property. In some specific cases, such as when a new field option appears with a field value changed, or you just want to keep some area updating by form update, you can modify the update logic of Form.Item via the `shouldUpdate` .

When `shouldUpdate` is `true` , any Form update will cause the Form.Item to be re-rendered. This is very helpful for custom rendering some areas:

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

You can ref [example](#) to see detail.

When `shouldUpdate` is a function, it will be called by form values update. Providing original values and current value to compare. This is very helpful for rendering additional fields based on values:

```
<Form.Item shouldUpdate={(prevValues, curValues) => prevValues.additional !==
curValues.additional}>
```

```
  {() => {
    return (
      <Form.Item name="other">
        <Input />
      </Form.Item>
    );
  }}
</Form.Item>
```

You can ref example to see detail.

messageVariables

You can modify the default verification information of Form.Item through `messageVariables`.

```
<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

Provides array management for fields.

Property	Description	Type	Default	Version
children	Render function	(fields: Field[], operation: { add, remove, move }, meta: { errors }) => React.ReactNode	-	
initialValue	Config sub default value. Form `initialValues` get higher priority when conflict	any[]	-	4.9.0
name	Field name, support array	NamePath	-	
rules	Validate rules, only support customize validator. Should work with ErrorList	{ validator, message }[]	-	4.7.0

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

Note: You should not configure Form.Item `initialValue` under Form.List. It always should be configured by Form.List `initialValue` or Form `initialValues`.

operation

Some operator functions in render form of Form.List.

Property	Description	Type	Default	Version
add	add form item	(defaultValue?: any, insertIndex?: number) => void	insertIndex	4.6.0
move	move form item	(from: number, to: number) => void	-	
remove	remove form item	(index: number | number[]) => void	number[]	4.5.0

Form.ErrorList

New in 4.7.0. Show error messages, should only work with `rules` of Form.List. See [example](#).

Property	Description	Type	Default
errors	Error list	ReactNode[]	-

Form.Provider

Provide linkage between forms. If a sub form with `name` prop update, it will auto trigger Provider related events. See [example](#).

Property	Description	Type	Default
onFormChange	Triggered when a sub form field updates	function(formName: string, info: { changedFields, forms })	-
onFormFinish	Triggered when a sub form submits	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

Name	Description	Type	Version
getFieldError	Get the error messages by the field name	(name: NamePath) => string[]	
getFieldInstance	Get field instance	(name: NamePath) => any	4.4.0
getFieldsError	Get the error messages by the fields name. Return as an array	(nameList?: NamePath[]) => FieldError[]	
getFieldsValue	Get values by a set of field names. Return according to the corresponding structure. Default return mounted field value, but you can use `getFieldsValue(true)` to get all values	(nameList?: NamePath[], filterFunc?: (meta: { touched: boolean, validating: boolean }) => boolean) => any	
getFieldValue	Get the value by the field name	(name: NamePath) => any	
isFieldsTouched	Check if fields have been operated. Check if all fields is touched when `allTouched` is `true`	(nameList?: NamePath[], allTouched?: boolean) => boolean	
isFieldTouched	Check if a field has been operated	(name: NamePath) => boolean	
isFieldValidating	Check field if is in validating	(name: NamePath) => boolean	
resetFields	Reset fields to `initialValues`	(fields?: NamePath[]) => void	
scrollToField	Scroll to field position	(name: NamePath, options: [ScrollOptions]) => void	
setFields	Set fields status	(fields: FieldData[]) => void	
setFieldsValue	Set fields value(Will directly pass to form store. If you do not want to modify passed object, please clone first)	(values) => void	
submit	Submit the form. It's same as click `submit` button	() => void	
validateFields	Validate fields	(nameList?: NamePath[]) => Promise	

validateFields return sample

```
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

Name	Description	Type
errors	Error messages	string[]
name	Field name path	NamePath[]
touched	Whether is operated	boolean
validating	Whether is in validating	boolean
value	Field value	any

Rule

Rule supports a config object, or a function returning config object:

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

Name	Description	Type	Version
defaultField	Validate rule for all array elements, valid when `type` is `array`	rule	

enum	Match enum value. You need to set `type` to `enum` to enable this	any[]	
fields	Validate rule for child elements, valid when `type` is `array` or `object`	Record<string, rule>	
len	Length of string, number, array	number	
max	`type` required: max length of `string`, `number`, `array`	number	
message	Error message. Will auto generate by template if not provided	string	
min	`type` required: min length of `string`, `number`, `array`	number	
pattern	Regex pattern	RegExp	
required	Required field	boolean	
transform	Transform value to the rule before validation	(value) => any	
type	Normally `string` |`number` |`boolean` |`url` | `email`. More type to ref here	string	
validateTrigger	Set validate trigger event. Must be the sub set of `validateTrigger` in Form.Item	string | string[]	
validator	Customize validation rule. Accept Promise as return. See example	(rule, value) => Promise	
warningOnly	Warning only. Not block form submit	boolean	4.17.0
whitespace	Failed if only has whitespace, only work with `type: 'string'` rule	boolean	

Migrate to v4

If you are a user of v3, you can ref migrate doc。

FAQ

Custom validator not working

It may be caused by your `validator` if it has some errors that prevents `callback` to be called. You can use `async` instead or use `try...catch` to catch the error:

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

// or

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

How does `name` fill value when it's an array?

`name` will fill value by array order. When there exists number in it and no related field in form store, it will auto convert field to array. If you want to keep it as object, use string like: `['1', 'name']`.

Why is there a form warning when used in Modal?

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

Before Modal opens, children elements do not exist in the view. You can set `forceRender` on Modal to pre-render its children. Click here to view an example.

Why is component `defaultValue` not working when inside Form.Item?

Components inside Form.Item with name property will turn into controlled mode, which makes `defaultValue` not work anymore. Please try `initialValues` of Form to set default value.

Why can not call `ref` of Form at first time?

`ref` only receives the mounted instance. please ref React official doc: https://reactjs.org/docs/refs-and-the-dom.html#accessing-refs

Why will `resetFields` re-mount component?

`resetFields` will re-mount component under Field to clean up customize component side effects (like async data, cached state, etc.). It's by design.

Difference between Form initialValues and Item initialValue?

In most case, we always recommend to use Form `initialValues`. Use Item `initialValue` only with dynamic field usage. Priority follows the rules:

1. Form `initialValues` is the first priority
2. Field `initialValue` is secondary *. Does not work when multiple Item with same `name` setting the `initialValue`

Why does `onFieldsChange` trigger three times on change when field sets `rules`?

Validating is also part of the value updating. It pass follow steps:

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

In each `onFieldsChange`, you will get `false` > `true` > `false` with `isFieldValidating`.

Why doesn't Form.List support `label` and need ErrorList to show errors?

Form.List use renderProps which mean internal structure is flexible. Thus `label` and `error` can not have best place. If you want to use antd `label`, you can wrap with Form.Item instead.

Why can't Form.Item `dependencies` work on Form.List field?

Your name path should also contain Form.List `name`:

```
<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>
```

dependencies should be `['users', 0, 'name']`

Why doesn't `normalize` support async?

React can not get correct interaction of controlled component with async value update. When user trigger `onChange`, component will do no response since `value` update is async. If you want to trigger value update async, you should use customize component to handle value state internal and pass sync value control to Form instead.

`scrollToFirstError` and `scrollToField` not working on custom form control?

See similar issues: #28370 #27994

`scrollToFirstError` and `scrollToField` deps on `id` attribute passed to form control, please mark sure that it hasn't been ignored in your custom form control. Check codesandbox for solution.

`setFieldsValue` do not trigger `onFieldsChange` or `onValuesChange`?

It's by design. Only user interactive can trigger the change event. This design is aim to avoid call `setFieldsValue` in change event which may makes loop calling.