何时使用 {#when-to-use}

- 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。

代码演示

基本使用

```
import React from 'react';
import type { FormProps } from 'antd';
import { Button, Checkbox, Form, Input } from 'antd';

type FieldType = {
  username?: string;
  password?: string;
  remember?: string;
};

const onFinish: FormProps<FieldType>['onFinish'] = (values) => {
  console.log('Success:', values);
};

const onFinishFailed: FormProps<FieldType>['onFinishFailed'] = (errorInfo)
=> {
  console.log('Failed:', errorInfo);
};

const App: React.FC = () => (
  <Form
    name="basic"
    labelCol={{ span: 8 }}
    wrapperCol={{ span: 16 }}
    style={{ maxWidth: 600 }}
    initialValues={{ remember: true }}
    onFinish={onFinish}
    onFinishFailed={onFinishFailed}
    autoComplete="off"
  >
    <Form.Item<FieldType>
      label="Username"
      name="username"
      rules={[{ required: true, message: 'Please input your username!' }]}
    >
      <Input />
    </Form.Item>
    <Form.Item<FieldType>
      label="Password"
      name="password"
      rules={[{ required: true, message: 'Please input your password!' }]}
    >
      <Input.Password />
    </Form.Item>

    <Form.Item<FieldType> name="remember" valuePropName="checked" label=
{null}>
      <Checkbox>Remember me</Checkbox>
    </Form.Item>

    <Form.Item label={null}>
      <Button type="primary" htmlType="submit">
        Submit
      </Button>
    </Form.Item>
  </Form>
);

export default App;
```

表单方法调用

```
import React from 'react';
import { Button, Form, Input, Select, Space } from 'antd';

const { Option } = Select;

const layout = {
  labelCol: { span: 8 },
  wrapperCol: { span: 16 },
};

const tailLayout = {
  wrapperCol: { offset: 8, span: 16 },
};

const App: React.FC = () => {
  const [form] = Form.useForm();

  const onGenderChange = (value: string) => {
    switch (value) {
      case 'male':
        form.setFieldsValue({ note: 'Hi, man!' });
        break;
      case 'female':
        form.setFieldsValue({ note: 'Hi, lady!' });
        break;
      case 'other':
        form.setFieldsValue({ note: 'Hi there!' });
        break;
      default:
    }
  };

  const onFinish = (values: any) => {
    console.log(values);
  };

  const onReset = () => {
    form.resetFields();
  };

  const onFill = () => {
    form.setFieldsValue({ note: 'Hello world!', gender: 'male' });
  };

  return (
    <Form
      {...layout}
      form={form}
      name="control-hooks"
      onFinish={onFinish}
      style={{ maxWidth: 600 }}
    >
      <Form.Item name="note" label="Note" rules={[{ required: true }]}>
        <Input />
      </Form.Item>
      <Form.Item name="gender" label="Gender" rules={[{ required: true }]}>
        <Select
          placeholder="Select a option and change input text above"
          onChange={onGenderChange}
          allowClear
        >
          <Option value="male">male</Option>
          <Option value="female">female</Option>
          <Option value="other">other</Option>
        </Select>
      </Form.Item>
      <Form.Item
```

```
        noStyle
        shouldUpdate={(prevValues, currentValues) => prevValues.gender !==
currentValues.gender}
      >
        {({ getFieldValue }) =>
          getFieldValue('gender') === 'other' ? (
            <Form.Item name="customizeGender" label="Customize Gender"
rules={[{ required: true }]}>
              <Input />
            </Form.Item>
          ) : null
        }
      </Form.Item>
      <Form.Item {...tailLayout}>
        <Space>
          <Button type="primary" htmlType="submit">
            Submit
          </Button>
          <Button htmlType="button" onClick={onReset}>
            Reset
          </Button>
          <Button type="link" htmlType="button" onClick={onFill}>
            Fill form
          </Button>
        </Space>
      </Form.Item>
    </Form>
  );
};

export default App;
```

表单布局

```
import React, { useState } from 'react';
import { Button, Form, Input, Radio } from 'antd';

type LayoutType = Parameters<typeof Form>[0]['layout'];

const App: React.FC = () => {
  const [form] = Form.useForm();
  const [formLayout, setFormLayout] = useState<LayoutType>('horizontal');

  const onFormLayoutChange = ({ layout }: { layout: LayoutType }) => {
    setFormLayout(layout);
  };
  return (
    <Form
      layout={formLayout}
      form={form}
      initialValues={{ layout: formLayout }}
      onValuesChange={onFormLayoutChange}
      style={{ maxWidth: formLayout === 'inline' ? 'none' : 600 }}
    >
      <Form.Item label="Form Layout" name="layout">
        <Radio.Group value={formLayout}>
          <Radio.Button value="horizontal">Horizontal</Radio.Button>
          <Radio.Button value="vertical">Vertical</Radio.Button>
          <Radio.Button value="inline">Inline</Radio.Button>
        </Radio.Group>
      </Form.Item>
      <Form.Item label="Field A">
        <Input placeholder="input placeholder" />
      </Form.Item>
      <Form.Item label="Field B">
        <Input placeholder="input placeholder" />
      </Form.Item>
      <Form.Item>
        <Button type="primary">Submit</Button>
      </Form.Item>
    </Form>
  );
};

export default App;
```

表单混合布局

```
import React from 'react';
import { Form, Input } from 'antd';

const App: React.FC = () => (
  <>
    <Form
      name="layout-multiple-horizontal"
      layout="horizontal"
      labelCol={{ span: 4 }}
      wrapperCol={{ span: 20 }}
    >
      <Form.Item label="horizontal" name="horizontal" rules={[{ required:
true }]}>
```

```
        <Input />
      </Form.Item>
      <Form.Item
        layout="vertical"
        label="vertical"
        name="vertical"
        rules={[{ required: true }]}
        labelCol={{ span: 24 }}
        wrapperCol={{ span: 24 }}
      >
        <Input />
      </Form.Item>
    </Form>
    <br />
    <Form
      name="layout-multiple-vertical"
      layout="vertical"
      labelCol={{ span: 4 }}
      wrapperCol={{ span: 20 }}
    >
      <Form.Item label="vertical" name="vertical" rules={[{ required: true
}]}>
        <Input />
      </Form.Item>
      <Form.Item
        layout="horizontal"
        label="horizontal"
        name="horizontal"
        rules={[{ required: true }]}
      >
        <Input />
      </Form.Item>
    </Form>
  </>
);

export default App;
```

表单禁用

```
import React, { useState } from 'react';
import { PlusOutlined } from '@ant-design/icons';
import {
  Button,
  Cascader,
  Checkbox,
  ColorPicker,
  DatePicker,
  Form,
  Input,
  InputNumber,
  Radio,
  Rate,
  Select,
  Slider,
  Switch,
  TreeSelect,
  Upload,
} from 'antd';

const { RangePicker } = DatePicker;
const { TextArea } = Input;

const normFile = (e: any) => {
  if (Array.isArray(e)) {
    return e;
  }
  return e?.fileList;
};

const FormDisabledDemo: React.FC = () => {
  const [componentDisabled, setComponentDisabled] = useState<boolean>
(true);

  return (
    <>
      <Checkbox
        checked={componentDisabled}
        onChange={(e) => setComponentDisabled(e.target.checked)}
      >
        Form disabled
      </Checkbox>
      <Form
        labelCol={{ span: 4 }}
        wrapperCol={{ span: 14 }}
        layout="horizontal"
        disabled={componentDisabled}
        style={{ maxWidth: 600 }}
      >
        <Form.Item label="Checkbox" name="disabled"
valuePropName="checked">
          <Checkbox>Checkbox</Checkbox>
        </Form.Item>
        <Form.Item label="Radio">
          <Radio.Group>
            <Radio value="apple"> Apple </Radio>
            <Radio value="pear"> Pear </Radio>
          </Radio.Group>
        </Form.Item>
        <Form.Item label="Input">
          <Input />
        </Form.Item>
        <Form.Item label="Select">
          <Select>
            <Select.Option value="demo">Demo</Select.Option>
          </Select>
        </Form.Item>
        <Form.Item label="TreeSelect">
          <TreeSelect
            treeData={[
              { title: 'Light', value: 'light', children: [{ title:
'Bamboo', value: 'bamboo' }] },
            ]}
          />
        </Form.Item>
        <Form.Item label="Cascader">
          <Cascader
            options={[
              {
                value: 'zhejiang',
                label: 'Zhejiang',
                children: [
                  {
                    value: 'hangzhou',
                    label: 'Hangzhou',
                  },
                ],
              },
            ]}
          />
        </Form.Item>
        <Form.Item label="DatePicker">
          <DatePicker />
        </Form.Item>
        <Form.Item label="RangePicker">
          <RangePicker />
        </Form.Item>
        <Form.Item label="InputNumber">
```

```
          <InputNumber />
        </Form.Item>
        <Form.Item label="TextArea">
          <TextArea rows={4} />
        </Form.Item>
        <Form.Item label="Switch" valuePropName="checked">
          <Switch />
        </Form.Item>
        <Form.Item label="Upload" valuePropName="fileList"
getValueFromEvent={normFile}>
          <Upload action="/upload.do" listType="picture-card">
            <button
              style={{ color: 'inherit', cursor: 'inherit', border: 0,
background: 'none' }}
              type="button"
            >
              <PlusOutlined />
              <div style={{ marginTop: 8 }}>Upload</div>
            </button>
          </Upload>
        </Form.Item>
        <Form.Item label="Button">
          <Button>Button</Button>
        </Form.Item>
        <Form.Item label="Slider">
          <Slider />
        </Form.Item>
        <Form.Item label="ColorPicker">
          <ColorPicker />
        </Form.Item>
        <Form.Item label="Rate">
          <Rate />
        </Form.Item>
      </Form>
    </>
  );
};

export default () => <FormDisabledDemo />;
```

表单变体

v5.13.0

```
import React from 'react';
import {
  Button,
  Cascader,
  DatePicker,
  Form,
  Input,
  InputNumber,
  Mentions,
  Segmented,
  Select,
  TreeSelect,
} from 'antd';

const { RangePicker } = DatePicker;

const formItemLayout = {
  labelCol: {
    xs: { span: 24 },
    sm: { span: 6 },
  },
  wrapperCol: {
    xs: { span: 24 },
    sm: { span: 14 },
  },
};

const App: React.FC = () => {
  const [form] = Form.useForm();
  const variant = Form.useWatch('variant', form);
  return (
    <Form
      {...formItemLayout}
      form={form}
      variant={variant || 'filled'}
      style={{ maxWidth: 600 }}
      initialValues={{ variant: 'filled' }}
    >
      <Form.Item label="Form variant" name="variant">
        <Segmented options={['outlined', 'filled', 'borderless',
'underlined']} />
      </Form.Item>

      <Form.Item label="Input" name="Input" rules={[{ required: true,
message: 'Please input!' }]}>
        <Input />
      </Form.Item>
      <Form.Item
        label="InputNumber"
        name="InputNumber"
        rules={[{ required: true, message: 'Please input!' }]}
      >
        <InputNumber style={{ width: '100%' }} />
      </Form.Item>

      <Form.Item
        label="TextArea"
        name="TextArea"
        rules={[{ required: true, message: 'Please input!' }]}
      >
        <Input.TextArea />
      </Form.Item>

      <Form.Item
        label="Mentions"
        name="Mentions"
        rules={[{ required: true, message: 'Please input!' }]}
      >
        <Mentions />
      </Form.Item>

      <Form.Item
        label="Select"
        name="Select"
        rules={[{ required: true, message: 'Please input!' }]}
      >
        <Select />
      </Form.Item>

      <Form.Item
        label="Cascader"
        name="Cascader"
        rules={[{ required: true, message: 'Please input!' }]}
      >
        <Cascader />
      </Form.Item>

      <Form.Item
        label="TreeSelect"
        name="TreeSelect"
        rules={[{ required: true, message: 'Please input!' }]}
      >
        <TreeSelect />
      </Form.Item>

      <Form.Item
        label="DatePicker"
        name="DatePicker"
        rules={[{ required: true, message: 'Please input!' }]}
      >
        <DatePicker />
      </Form.Item>

      <Form.Item
        label="RangePicker"
        name="RangePicker"
        rules={[{ required: true, message: 'Please input!' }]}
      >
        <RangePicker />
      </Form.Item>

      <Form.Item wrapperCol={{ offset: 6, span: 16 }}>
        <Button type="primary" htmlType="submit">
          Submit
        </Button>
      </Form.Item>
    </Form>
  );
};

export default App;
```

必选样式

```
import React, { useState } from 'react';
import { InfoCircleOutlined } from '@ant-design/icons';
import { Button, Form, Input, Radio, Tag } from 'antd';

type RequiredMark = boolean | 'optional' | 'customize';

const customizeRequiredMark = (label: React.ReactNode, { required }: {
required: boolean }) => (
  <>
    {required ? <Tag color="error">Required</Tag> : <Tag
color="warning">optional</Tag>}
    {label}
  </>
);
```

```
const App: React.FC = () => {
  const [form] = Form.useForm();
  const [requiredMark, setRequiredMarkType] = useState<RequiredMark>
('optional');

  const onRequiredTypeChange = ({ requiredMarkValue }: { requiredMarkValue:
RequiredMark }) => {
    setRequiredMarkType(requiredMarkValue);
  };

  return (
    <Form
      form={form}
      layout="vertical"
      initialValues={{ requiredMarkValue: requiredMark }}
      onValuesChange={onRequiredTypeChange}
      requiredMark={requiredMark === 'customize' ? customizeRequiredMark :
requiredMark}
    >
      <Form.Item label="Required Mark" name="requiredMarkValue">
        <Radio.Group>
          <Radio.Button value>Default</Radio.Button>
          <Radio.Button value="optional">Optional</Radio.Button>
          <Radio.Button value={false}>Hidden</Radio.Button>
          <Radio.Button value="customize">Customize</Radio.Button>
        </Radio.Group>
      </Form.Item>
      <Form.Item label="Field A" required tooltip="This is a required
field">
        <Input placeholder="input placeholder" />
      </Form.Item>
      <Form.Item
        label="Field B"
        tooltip={{ title: 'Tooltip with customize icon', icon:
<InfoCircleOutlined /> }}
      >
        <Input placeholder="input placeholder" />
      </Form.Item>
      <Form.Item>
        <Button type="primary">Submit</Button>
      </Form.Item>
    </Form>
  );
};

export default App;
```

表单尺寸

```
import React, { useState } from 'react';
import {
  Button,
  Cascader,
  DatePicker,
  Form,
  Input,
  InputNumber,
  Radio,
  Select,
  Switch,
  TreeSelect,
} from 'antd';

type SizeType = Parameters<typeof Form>[0]['size'];

const App: React.FC = () => {
  const [componentSize, setComponentSize] = useState<SizeType | 'default'>
('default');

  const onFormLayoutChange = ({ size }: { size: SizeType }) => {
    setComponentSize(size);
  };

  return (
    <Form
      labelCol={{ span: 4 }}
      wrapperCol={{ span: 14 }}
      layout="horizontal"
      initialValues={{ size: componentSize }}
      onValuesChange={onFormLayoutChange}
      size={componentSize as SizeType}
      style={{ maxWidth: 600 }}
    >
      <Form.Item label="Form Size" name="size">
        <Radio.Group>
          <Radio.Button value="small">Small</Radio.Button>
          <Radio.Button value="default">Default</Radio.Button>
          <Radio.Button value="large">Large</Radio.Button>
        </Radio.Group>
      </Form.Item>
      <Form.Item label="Input">
        <Input />
      </Form.Item>
      <Form.Item label="Select">
        <Select>
          <Select.Option value="demo">Demo</Select.Option>
        </Select>
      </Form.Item>
      <Form.Item label="TreeSelect">
        <TreeSelect
          treeData={[
            { title: 'Light', value: 'light', children: [{ title: 'Bamboo', 
value: 'bamboo' }] },
          ]}
        />
      </Form.Item>
      <Form.Item label="Cascader">
        <Cascader
          options={[
            {
              value: 'zhejiang',
              label: 'Zhejiang',
              children: [{ value: 'hangzhou', label: 'Hangzhou' }],
            },
          ]}
        />
      </Form.Item>
      <Form.Item label="DatePicker">
        <DatePicker />
      </Form.Item>
      <Form.Item label="InputNumber">
        <InputNumber />
      </Form.Item>
      <Form.Item label="Switch" valuePropName="checked">
        <Switch />
      </Form.Item>
      <Form.Item label="Button">
        <Button>Button</Button>
      </Form.Item>
    </Form>
  );
};

export default App;
```

表单标签可换行

```
import React from 'react';
import { Button, Form, Input } from 'antd';
```

```
const App: React.FC = () => (
  <Form
    name="wrap"
    labelCol={{ flex: '110px' }}
    labelAlign="left"
    labelWrap
    wrapperCol={{ flex: 1 }}
    colon={false}
    style={{ maxWidth: 600 }}
  >
    <Form.Item label="Normal label" name="username" rules={[{ required:
true }]}>
      <Input />
    </Form.Item>

    <Form.Item label="A super long label text" name="password" rules={[{
required: true }]}>
      <Input />
    </Form.Item>

    <Form.Item label=" ">
      <Button type="primary" htmlType="submit">
        Submit
      </Button>
    </Form.Item>
  </Form>
);

export default App;
```

非阻塞校验

```
import React from 'react';
import { Button, Form, Input, message, Space } from 'antd';

const App: React.FC = () => {
  const [form] = Form.useForm();

  const onFinish = () => {
    message.success('Submit success!');
  };

  const onFinishFailed = () => {
    message.error('Submit failed!');
  };
  const onFill = () => {
    form.setFieldsValue({
      url: 'https://taobao.com/',
    });
  };

  return (
    <Form
      form={form}
      layout="vertical"
      onFinish={onFinish}
      onFinishFailed={onFinishFailed}
      autoComplete="off"
    >
      <Form.Item
        name="url"
        label="URL"
        rules={[{ required: true }, { type: 'url', warningOnly: true }, {
type: 'string', min: 6 }]}
      >
        <Input placeholder="input placeholder" />
      </Form.Item>
      <Form.Item>
        <Space>
          <Button type="primary" htmlType="submit">
            Submit
          </Button>
          <Button htmlType="button" onClick={onFill}>
            Fill
          </Button>
        </Space>
      </Form.Item>
    </Form>
  );
};

export default App;
```

字段监听 Hooks

```
import React from 'react';
import { Form, Input, InputNumber, Typography } from 'antd';

const Demo: React.FC = () => {
  const [form] = Form.useForm<{ name: string; age: number }>();
  const nameValue = Form.useWatch('name', form);
  // The selector is static and does not support closures.
  const customValue = Form.useWatch((values) => `name: ${values.name ||
''}`, form);

  return (
    <>
      <Form form={form} layout="vertical" autoComplete="off">
        <Form.Item name="name" label="Name (Watch to trigger rerender)">
          <Input />
        </Form.Item>
        <Form.Item name="age" label="Age (Not Watch)">
          <InputNumber />
        </Form.Item>
      </Form>

      <Typography>
        <pre>Name Value: {nameValue}</pre>
        <pre>Custom Value: {customValue}</pre>
      </Typography>
    </>
  );
};

export default Demo;
```

校验时机

```
import React from 'react';
import { Alert, Form, Input } from 'antd';

const App: React.FC = () => (
  <Form name="trigger" style={{ maxWidth: 600 }} layout="vertical"
autoComplete="off">
    <Alert message="Use 'max' rule, continue type chars to see it" />

    <Form.Item
      hasFeedback
      label="Field A"
      name="field_a"
      validateTrigger="onBlur"
      rules={[{ max: 3 }]}
    >
      <Input placeholder="Validate required onBlur" />
    </Form.Item>
    <Form.Item
      hasFeedback
      label="Field B"
      name="field_b"
      validateDebounce={1000}
      rules={[{ max: 3 }]}
    >
      <Input placeholder="Validate required debounce after 1s" />
    </Form.Item>

    <Form.Item
      hasFeedback
      label="Field C"
      name="field_c"
      validateFirst
      rules={[{ max: 6 }, { max: 3, message: 'Continue input to exceed 6
chars' }]}
    >
      <Input placeholder="Validate one by one" />
    </Form.Item>
  </Form>
);

export default App;
```

仅校验

```
import React from 'react';
import type { FormInstance } from 'antd';
import { Button, Form, Input, Space } from 'antd';

interface SubmitButtonProps {
  form: FormInstance;
}

const SubmitButton: React.FC<React.PropsWithChildren<SubmitButtonProps>> =
({ form, children }) => {
  const [submittable, setSubmittable] = React.useState<boolean>(false);

  // Watch all values
  const values = Form.useWatch([], form);

  React.useEffect(() => {
    form
      .validateFields({ validateOnly: true })
      .then(() => setSubmittable(true))
      .catch(() => setSubmittable(false));
  }, [form, values]);

  return (
    <Button type="primary" htmlType="submit" disabled={!submittable}>
      {children}
    </Button>
  );
};

const App: React.FC = () => {
  const [form] = Form.useForm();
  return (
    <Form form={form} name="validateOnly" layout="vertical"
autoComplete="off">
      <Form.Item name="name" label="Name" rules={[{ required: true }]}>
        <Input />
      </Form.Item>
      <Form.Item name="age" label="Age" rules={[{ required: true }]}>
        <Input />
      </Form.Item>
      <Form.Item>
        <Space>
          <SubmitButton form={form}>Submit</SubmitButton>
          <Button htmlType="reset">Reset</Button>
        </Space>
      </Form.Item>
    </Form>
  );
};

export default App;
```

字段路径前缀

```
import React from 'react';
import { Button, Form, Input } from 'antd';
import type { FormItemProps } from 'antd';

const MyFormItemContext = React.createContext<(string | number)[]>([]);

interface MyFormItemGroupProps {
  prefix: string | number | (string | number)[];
}

function toArr(str: string | number | (string | number)[]): (string |
number)[] {
  return Array.isArray(str) ? str : [str];
}

const MyFormItemGroup:
React.FC<React.PropsWithChildren<MyFormItemGroupProps>> = ({
  prefix,
  children,
}) => {
  const prefixPath = React.useContext(MyFormItemContext);
  const concatPath = React.useMemo(() => [...prefixPath, ...toArr(prefix)],
[prefixPath, prefix]);

  return <MyFormItemContext.Provider value={concatPath}>{children}
</MyFormItemContext.Provider>;
};

const MyFormItem = ({ name, ...props }: FormItemProps) => {
  const prefixPath = React.useContext(MyFormItemContext);
  const concatName = name !== undefined ? [...prefixPath, ...toArr(name)] :
undefined;

  return <Form.Item name={concatName} {...props} />;
};

const App: React.FC = () => {
  const onFinish = (value: object) => {
    console.log(value);
  };

  return (
    <Form name="form_item_path" layout="vertical" onFinish={onFinish}>
      <MyFormItemGroup prefix={['user']}>
        <MyFormItemGroup prefix={['name']}>
          <MyFormItem name="firstName" label="First Name">
            <Input />
          </MyFormItem>
          <MyFormItem name="lastName" label="Last Name">
            <Input />
          </MyFormItem>
        </MyFormItemGroup>

        <MyFormItem name="age" label="Age">
          <Input />
        </MyFormItem>
      </MyFormItemGroup>
      <Button type="primary" htmlType="submit">
        Submit
      </Button>
    </Form>
  );
};

export default App;
```

动态增减表单项

```
import React from 'react';
import { MinusCircleOutlined, PlusOutlined } from '@ant-design/icons';
import { Button, Form, Input } from 'antd';

const formItemLayout = {
  labelCol: {
    xs: { span: 24 },
    sm: { span: 4 },
  },
  wrapperCol: {
    xs: { span: 24 },
    sm: { span: 20 },
  },
};

const formItemLayoutWithOutLabel = {
  wrapperCol: {
    xs: { span: 24, offset: 0 },
    sm: { span: 20, offset: 4 },
  },
};

const App: React.FC = () => {
  const onFinish = (values: any) => {
    console.log('Received values of form:', values);
  };

  return (
    <Form
      name="dynamic_form_item"
      {...formItemLayoutWithOutLabel}
      onFinish={onFinish}
      style={{ maxWidth: 600 }}
    >
```

```
      <Form.List
        name="names"
        rules={[
          {
            validator: async (_, names) => {
              if (!names || names.length < 2) {
                return Promise.reject(new Error('At least 2 passengers'));
              }
            },
          },
        ]}
      >
        {(fields, { add, remove }, { errors }) => (
          <>
            {fields.map((field, index) => (
              <Form.Item
                {...(index === 0 ? formItemLayout :
formItemLayoutWithOutLabel)}
                label={index === 0 ? 'Passengers' : ''}
                required={false}
                key={field.key}
              >
                <Form.Item
                  {...field}
                  validateTrigger={['onChange', 'onBlur']}
                  rules={[
                    {
                      required: true,
                      whitespace: true,
                      message: "Please input passenger's name or delete
this field.",
                    },
                  ]}
                  noStyle
                >
                  <Input placeholder="passenger name" style={{ width: '60%'
}} />
                </Form.Item>
                {fields.length > 1 ? (
                  <MinusCircleOutlined
                    className="dynamic-delete-button"
                    onClick={() => remove(field.name)}
                  />
                ) : null}
              </Form.Item>
            ))}
```

```
              <Form.Item>
                <Button
                  type="dashed"
                  onClick={() => add()}
                  style={{ width: '60%' }}
                  icon={<PlusOutlined />}
                >
                  Add field
                </Button>
                <Button
                  type="dashed"
                  onClick={() => {
                    add('The head item', 0);
                  }}
                  style={{ width: '60%', marginTop: '20px' }}
                  icon={<PlusOutlined />}
                >
                  Add field at head
                </Button>
                <Form.ErrorList errors={errors} />
              </Form.Item>
            </>
          )}
        </Form.List>
        <Form.Item>
          <Button type="primary" htmlType="submit">
            Submit
          </Button>
        </Form.Item>
      </Form>
    );
};

export default App;
```

动态增减嵌套字段

```
import React from 'react';
import { MinusCircleOutlined, PlusOutlined } from '@ant-design/icons';
import { Button, Form, Input, Space } from 'antd';

const onFinish = (values: any) => {
  console.log('Received values of form:', values);
};

const App: React.FC = () => (
  <Form
    name="dynamic_form_nest_item"
    onFinish={onFinish}
    style={{ maxWidth: 600 }}
    autoComplete="off"
  >
    <Form.List name="users">
      {(fields, { add, remove }) => (
        <>
          {fields.map(({ key, name, ...restField }) => (
            <Space key={key} style={{ display: 'flex', marginBottom: 8 }}
align="baseline">
              <Form.Item
                {...restField}
                name={[name, 'first']}
                rules={[{ required: true, message: 'Missing first name' }]}
              >
                <Input placeholder="First Name" />
              </Form.Item>
              <Form.Item
                {...restField}
                name={[name, 'last']}
                rules={[{ required: true, message: 'Missing last name' }]}
              >
                <Input placeholder="Last Name" />
              </Form.Item>
              <MinusCircleOutlined onClick={() => remove(name)} />
            </Space>
          ))}
          <Form.Item>
            <Button type="dashed" onClick={() => add()} block icon=
{<PlusOutlined />}>
              Add field
            </Button>
          </Form.Item>
        </>
      )}
    </Form.List>
    <Form.Item>
      <Button type="primary" htmlType="submit">
        Submit
      </Button>
    </Form.Item>
  </Form>
);
```

```
export default App;
```

动态增减嵌套纯字段

Debug

```
import React from 'react';
import { MinusCircleOutlined, PlusOutlined } from '@ant-design/icons';
import { Button, Form, Input, Space } from 'antd';

const onFinish = (values: any) => {
  console.log('Received values of form:', values);
};

const App: React.FC = () => (
  <Form
    name="dynamic_form_no_style"
    onFinish={onFinish}
    style={{ maxWidth: 600 }}
    autoComplete="off"
  >
    <Form.Item label="Users">
      <Form.List name="users">
        {(fields, { add, remove }) => (
          <>
            {fields.map((field) => (
              <Space key={field.key} style={{ marginBottom: 16 }}>
                <Form.Item noStyle name={[field.name, 'lastName']} rules=
{[{ required: true }]}>
                  <Input placeholder="Last Name" />
                </Form.Item>
                <Form.Item noStyle name={[field.name, 'firstName']} rules=
{[{ required: true }]}>
                  <Input placeholder="First Name" />
                </Form.Item>
                <MinusCircleOutlined
                  onClick={() => {
                    remove(field.name);
                  }}
                />
              </Space>
            ))}
            <Form.Item>
              <Button type="dashed" onClick={() => add()} block icon=
{<PlusOutlined />}>
```

```
                Add field
              </Button>
            </Form.Item>
          </>
        )}
      </Form.List>
    </Form.Item>
    <Form.Item>
      <Button type="primary" htmlType="submit">
        Submit
      </Button>
    </Form.Item>
  </Form>
);

export default App;
```

复杂的动态增减表单项

```
import React from 'react';
import { CloseOutlined } from '@ant-design/icons';
import { Button, Card, Form, Input, Space, Typography } from 'antd';

const App: React.FC = () => {
  const [form] = Form.useForm();

  return (
    <Form
      labelCol={{ span: 6 }}
      wrapperCol={{ span: 18 }}
      form={form}
      name="dynamic_form_complex"
      style={{ maxWidth: 600 }}
      autoComplete="off"
      initialValues={{ items: [{}] }}
    >
      <Form.List name="items">
        {(fields, { add, remove }) => (
          <div style={{ display: 'flex', rowGap: 16, flexDirection:
'column' }}>
            {fields.map((field) => (
              <Card
                size="small"
                title={`Item ${field.name + 1}`}
                key={field.key}
                extra={
                  <CloseOutlined
                    onClick={() => {
                      remove(field.name);
                    }}
                  />
                }
              >
                <Form.Item label="Name" name={[field.name, 'name']}>
                  <Input />
                </Form.Item>

                {/* Nest Form.List */}
                <Form.Item label="List">
                  <Form.List name={[field.name, 'list']}>
                    {(subFields, subOpt) => (
                      <div style={{ display: 'flex', flexDirection:
'column', rowGap: 16 }}>
                        {subFields.map((subField) => (
                          <Space key={subField.key}>
                            <Form.Item noStyle name={[subField.name,
'first']}>
                              <Input placeholder="first" />
                            </Form.Item>
                            <Form.Item noStyle name={[subField.name,
'second']}>
                              <Input placeholder="second" />
                            </Form.Item>
                            <CloseOutlined
                              onClick={() => {
                                subOpt.remove(subField.name);
                              }}
                            />
                          </Space>
                        ))}
                        <Button type="dashed" onClick={() => subOpt.add()}
block>
                          + Add Sub Item
                        </Button>
                      </div>
                    )}
                  </Form.List>
                </Form.Item>
              </Card>
            ))}

            <Button type="dashed" onClick={() => add()} block>
```

```
              + Add Item
            </Button>
          </div>
        )}
      </Form.List>

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

export default App;
```

嵌套结构与校验信息

```
import React from 'react';
import { Button, Form, Input, InputNumber } from 'antd';

const layout = {
  labelCol: { span: 8 },
  wrapperCol: { span: 16 },
};

const validateMessages = {
  required: '${label} is required!',
  types: {
    email: '${label} is not a valid email!',
    number: '${label} is not a valid number!',
  },
  number: {
    range: '${label} must be between ${min} and ${max}',
  },
};

const onFinish = (values: any) => {
  console.log(values);
};

const App: React.FC = () => (
  <Form
```

```
    {...layout}
    name="nest-messages"
    onFinish={onFinish}
    style={{ maxWidth: 600 }}
    validateMessages={validateMessages}
  >
    <Form.Item name={['user', 'name']} label="Name" rules={[{ required:
true }]}>
      <Input />
    </Form.Item>
    <Form.Item name={['user', 'email']} label="Email" rules={[{ type:
'email' }]}>
      <Input />
    </Form.Item>
    <Form.Item name={['user', 'age']} label="Age" rules={[{ type: 'number',
min: 0, max: 99 }]}>
      <InputNumber />
    </Form.Item>
    <Form.Item name={['user', 'website']} label="Website">
      <Input />
    </Form.Item>
    <Form.Item name={['user', 'introduction']} label="Introduction">
      <Input.TextArea />
    </Form.Item>
    <Form.Item label={null}>
      <Button type="primary" htmlType="submit">
        Submit
      </Button>
    </Form.Item>
  </Form>
);

export default App;
```

复杂一点的控件

```
import React from 'react';
import { Button, Form, Input, Select, Space, Tooltip, Typography } from
'antd';

const { Option } = Select;

const onFinish = (values: any) => {
  console.log('Received values of form: ', values);
};
```

```
const App: React.FC = () => (
  <Form
    name="complex-form"
    onFinish={onFinish}
    labelCol={{ span: 8 }}
    wrapperCol={{ span: 16 }}
    style={{ maxWidth: 600 }}
  >
    <Form.Item label="Username">
      <Space>
        <Form.Item
          name="username"
          noStyle
          rules={[{ required: true, message: 'Username is required' }]}
        >
          <Input style={{ width: 160 }} placeholder="Please input" />
        </Form.Item>
        <Tooltip title="Useful information">
          <Typography.Link href="#API">Need Help?</Typography.Link>
        </Tooltip>
      </Space>
    </Form.Item>
    <Form.Item label="Address">
      <Space.Compact>
        <Form.Item
          name={['address', 'province']}
          noStyle
          rules={[{ required: true, message: 'Province is required' }]}
        >
          <Select placeholder="Select province">
            <Option value="Zhejiang">Zhejiang</Option>
            <Option value="Jiangsu">Jiangsu</Option>
          </Select>
        </Form.Item>
        <Form.Item
          name={['address', 'street']}
          noStyle
          rules={[{ required: true, message: 'Street is required' }]}
        >
          <Input style={{ width: '50%' }} placeholder="Input street" />
        </Form.Item>
      </Space.Compact>
    </Form.Item>
    <Form.Item label="BirthDate" style={{ marginBottom: 0 }}>
      <Form.Item
        name="year"
        rules={[{ required: true }]}
        style={{ display: 'inline-block', width: 'calc(50% - 8px)' }}
      >
        <Input placeholder="Input birth year" />
      </Form.Item>
      <Form.Item
        name="month"
        rules={[{ required: true }]}
        style={{ display: 'inline-block', width: 'calc(50% - 8px)', margin:
'0 8px' }}
      >
        <Input placeholder="Input birth month" />
      </Form.Item>
    </Form.Item>
    <Form.Item label={null}>
      <Button type="primary" htmlType="submit">
        Submit
      </Button>
    </Form.Item>
  </Form>
);

export default App;
```

自定义表单控件

```
import React, { useState } from 'react';
import { Button, Form, Input, Select } from 'antd';

const { Option } = Select;

type Currency = 'rmb' | 'dollar';

interface PriceValue {
  number?: number;
  currency?: Currency;
}

interface PriceInputProps {
  id?: string;
  value?: PriceValue;
  onChange?: (value: PriceValue) => void;
}

const PriceInput: React.FC<PriceInputProps> = (props) => {
  const { id, value = {}, onChange } = props;
  const [number, setNumber] = useState(0);
  const [currency, setCurrency] = useState<Currency>('rmb');

  const triggerChange = (changedValue: { number?: number; currency?:
Currency }) => {
    onChange?.({ number, currency, ...value, ...changedValue });
  };

  const onNumberChange = (e: React.ChangeEvent<HTMLInputElement>) => {
    const newNumber = parseInt(e.target.value || '0', 10);
    if (Number.isNaN(number)) {
      return;
    }
    if (!('number' in value)) {
      setNumber(newNumber);
    }
    triggerChange({ number: newNumber });
  };

  const onCurrencyChange = (newCurrency: Currency) => {
    if (!('currency' in value)) {
      setCurrency(newCurrency);
    }
    triggerChange({ currency: newCurrency });
  };

  return (
    <span id={id}>
      <Input
        type="text"
        value={value.number || number}
        onChange={onNumberChange}
        style={{ width: 100 }}
      />
      <Select
        value={value.currency || currency}
        style={{ width: 80, margin: '0 8px' }}
        onChange={onCurrencyChange}
      >
        <Option value="rmb">RMB</Option>
        <Option value="dollar">Dollar</Option>
      </Select>
    </span>
  );
};
```

```
const App: React.FC = () => {
  const onFinish = (values: any) => {
    console.log('Received values from form: ', values);
  };

  const checkPrice = (_: any, value: { number: number }) => {
    if (value.number > 0) {
      return Promise.resolve();
    }
    return Promise.reject(new Error('Price must be greater than zero!'));
  };

  return (
    <Form
      name="customized_form_controls"
      layout="inline"
      onFinish={onFinish}
      initialValues={{
        price: {
          number: 0,
          currency: 'rmb',
        },
      }}
    >
      <Form.Item name="price" label="Price" rules={[{ validator: checkPrice
}]}>
        <PriceInput />
      </Form.Item>
      <Form.Item>
        <Button type="primary" htmlType="submit">
          Submit
        </Button>
      </Form.Item>
    </Form>
  );
};

export default App;
```

表单数据存储于上层组件

```
import React, { useState } from 'react';
import { Form, Input, Typography } from 'antd';

const { Paragraph } = Typography;
```

```
interface FieldData {
  name: string | number | (string | number)[];
  value?: any;
  touched?: boolean;
  validating?: boolean;
  errors?: string[];
}

interface CustomizedFormProps {
  onChange: (fields: FieldData[]) => void;
  fields: FieldData[];
}

const CustomizedForm: React.FC<CustomizedFormProps> = ({ onChange, fields
}) => (
  <Form
    name="global_state"
    layout="inline"
    fields={fields}
    onFieldsChange={(_, allFields) => {
      onChange(allFields);
    }}
  >
    <Form.Item
      name="username"
      label="Username"
      rules={[{ required: true, message: 'Username is required!' }]}
    >
      <Input />
    </Form.Item>
  </Form>
);

const App: React.FC = () => {
  const [fields, setFields] = useState<FieldData[]>([{ name: ['username'],
value: 'Ant Design' }]);

  return (
    <>
      <CustomizedForm
        fields={fields}
        onChange={(newFields) => {
          setFields(newFields);
        }}
      />
      <Paragraph style={{ maxWidth: 440, marginTop: 24 }}>
```

```
        <pre style={{ border: 'none' }}>{JSON.stringify(fields, null, 2)}
</pre>
      </Paragraph>
    </>
  );
};

export default App;
```

多表单联动

```
import React, { useEffect, useRef, useState } from 'react';
import { SmileOutlined, UserOutlined } from '@ant-design/icons';
import { Avatar, Button, Flex, Form, Input, InputNumber, Modal, Space,
Typography } from 'antd';
import type { GetRef } from 'antd';

type FormInstance = GetRef<typeof Form>;

const layout = {
  labelCol: { span: 8 },
  wrapperCol: { span: 16 },
};

const tailLayout = {
  wrapperCol: { offset: 8, span: 16 },
};

interface UserType {
  name: string;
  age: string;
}

interface ModalFormProps {
  open: boolean;
  onCancel: () => void;
}

// reset form fields when modal is form, closed
const useResetFormOnCloseModal = ({ form, open }: { form: FormInstance;
open: boolean }) => {
  const prevOpenRef = useRef<boolean>(null);
  useEffect(() => {
    prevOpenRef.current = open;
  }, [open]);
  const prevOpen = prevOpenRef.current;
  useEffect(() => {
    if (!open && prevOpen) {
      form.resetFields();
    }
  }, [form, prevOpen, open]);
};

const ModalForm: React.FC<ModalFormProps> = ({ open, onCancel }) => {
  const [form] = Form.useForm();

  useResetFormOnCloseModal({
    form,
    open,
  });

  const onOk = () => {
    form.submit();
  };

  return (
    <Modal title="Basic Drawer" open={open} onOk={onOk} onCancel=
{onCancel}>
      <Form form={form} layout="vertical" name="userForm">
        <Form.Item name="name" label="User Name" rules={[{ required: true
}]}>
          <Input />
        </Form.Item>
        <Form.Item name="age" label="User Age" rules={[{ required: true
}]}>
          <InputNumber />
        </Form.Item>
      </Form>
    </Modal>
  );
};

const App: React.FC = () => {
  const [open, setOpen] = useState(false);

  const showUserModal = () => {
    setOpen(true);
  };

  const hideUserModal = () => {
    setOpen(false);
  };

  const onFinish = (values: any) => {
    console.log('Finish:', values);
  };

  return (
    <Form.Provider
      onFormFinish={(name, { values, forms }) => {
        if (name === 'userForm') {
          const { basicForm } = forms;
          const users = basicForm.getFieldValue('users') || [];
          basicForm.setFieldsValue({ users: [...users, values] });
          setOpen(false);
        }
      }}
    >
      <Form {...layout} name="basicForm" onFinish={onFinish} style={{
maxWidth: 600 }}>
        <Form.Item name="group" label="Group Name" rules={[{ required: true
}]}>
          <Input />
        </Form.Item>

        {/* Create a hidden field to make Form instance record this */}
        <Form.Item name="users" noStyle />

        <Form.Item
          label="User List"
          shouldUpdate={(prevValues, curValues) => prevValues.users !==
curValues.users}
        >
          {({ getFieldValue }) => {
            const users: UserType[] = getFieldValue('users') || [];
            return users.length ? (
              <Flex vertical gap={8}>
                {users.map((user) => (
                  <Space key={user.name}>
                    <Avatar icon={<UserOutlined />} />
                    {`${user.name} - ${user.age}`}
                  </Space>
                ))}
              </Flex>
            ) : (
              <Typography.Text className="ant-form-text" type="secondary">
                ( <SmileOutlined /> No user yet. )
              </Typography.Text>
            );
          }}
        </Form.Item>
        <Form.Item {...tailLayout}>
          <Button htmlType="submit" type="primary">
            Submit
          </Button>
          <Button htmlType="button" style={{ margin: '0 8px' }} onClick=
{showUserModal}>
            Add User
          </Button>
        </Form.Item>
      </Form>

      <ModalForm open={open} onCancel={hideUserModal} />
    </Form.Provider>
  );
};

export default App;
```

内联登录栏

```
import React, { useEffect, useState } from 'react';
import { LockOutlined, UserOutlined } from '@ant-design/icons';
import { Button, Form, Input } from 'antd';

const App: React.FC = () => {
  const [form] = Form.useForm();
  const [clientReady, setClientReady] = useState<boolean>(false);

  // To disable submit button at the beginning.
  useEffect(() => {
    setClientReady(true);
  }, []);

  const onFinish = (values: any) => {
    console.log('Finish:', values);
  };

  return (
    <Form form={form} name="horizontal_login" layout="inline" onFinish=
{onFinish}>
      <Form.Item
        name="username"
        rules={[{ required: true, message: 'Please input your username!'
}]}
      >
        <Input prefix={<UserOutlined />} placeholder="Username" />
      </Form.Item>
      <Form.Item
        name="password"
        rules={[{ required: true, message: 'Please input your password!'
}]}
      >
        <Input prefix={<LockOutlined />} type="password"
placeholder="Password" />
      </Form.Item>
      <Form.Item shouldUpdate>
        {() => (
          <Button
            type="primary"
            htmlType="submit"
            disabled={
              !clientReady ||
              !form.isFieldsTouched(true) ||
              !!form.getFieldsError().filter(({ errors }) =>
errors.length).length
            }
          >
            Log in
          </Button>
        )}
      </Form.Item>
    </Form>
  );
};

export default App;
```

登录框

```
import React from 'react';
import { LockOutlined, UserOutlined } from '@ant-design/icons';
import { Button, Checkbox, Form, Input, Flex } from 'antd';

const App: React.FC = () => {
  const onFinish = (values: any) => {
    console.log('Received values of form: ', values);
  };
  return (
    <Form
      name="login"
      initialValues={{ remember: true }}
      style={{ maxWidth: 360 }}
      onFinish={onFinish}
    >
      <Form.Item
        name="username"
        rules={[{ required: true, message: 'Please input your Username!'
}]}
      >
        <Input prefix={<UserOutlined />} placeholder="Username" />
      </Form.Item>
      <Form.Item
        name="password"
        rules={[{ required: true, message: 'Please input your Password!'
}]}
      >
        <Input prefix={<LockOutlined />} type="password"
placeholder="Password" />
      </Form.Item>
      <Form.Item>
        <Flex justify="space-between" align="center">
          <Form.Item name="remember" valuePropName="checked" noStyle>
            <Checkbox>Remember me</Checkbox>
          </Form.Item>
          <a href="">Forgot password</a>
        </Flex>
      </Form.Item>

      <Form.Item>
        <Button block type="primary" htmlType="submit">
          Log in
        </Button>
        or <a href="">Register now!</a>
      </Form.Item>
    </Form>
  );
};

export default App;
```

注册新用户

```
import React, { useState } from 'react';
import type { CascaderProps } from 'antd';
import {
  AutoComplete,
  Button,
  Cascader,
  Checkbox,
  Col,
  Form,
  Input,
  InputNumber,
  Row,
  Select,
} from 'antd';

const { Option } = Select;

interface DataNodeType {
  value: string;
  label: string;
  children?: DataNodeType[];
}

const residences: CascaderProps<DataNodeType>['options'] = [
  {
    value: 'zhejiang',
    label: 'Zhejiang',
    children: [
      {
        value: 'hangzhou',
        label: 'Hangzhou',
        children: [
          {
            value: 'xihu',
            label: 'West Lake',
          },
        ],
      },
    ],
  },
  {
    value: 'jiangsu',
    label: 'Jiangsu',
    children: [
      {
        value: 'nanjing',
        label: 'Nanjing',
        children: [
          {
            value: 'zhonghuamen',
            label: 'Zhong Hua Men',
          },
        ],
      },
    ],
  },
];

const formItemLayout = {
  labelCol: {
    xs: { span: 24 },
    sm: { span: 8 },
  },
  wrapperCol: {
    xs: { span: 24 },
    sm: { span: 16 },
  },
};

const tailFormItemLayout = {
  wrapperCol: {
    xs: {
      span: 24,
      offset: 0,
    },
    sm: {
      span: 16,
      offset: 8,
    },
  },
};

const App: React.FC = () => {
  const [form] = Form.useForm();

  const onFinish = (values: any) => {
    console.log('Received values of form: ', values);
  };

  const prefixSelector = (
    <Form.Item name="prefix" noStyle>
      <Select style={{ width: 70 }}>
```

```
        <Option value="86">+86</Option>
        <Option value="87">+87</Option>
      </Select>
    </Form.Item>
  );

  const suffixSelector = (
    <Form.Item name="suffix" noStyle>
      <Select style={{ width: 70 }}>
        <Option value="USD">$</Option>
        <Option value="CNY">¥</Option>
      </Select>
    </Form.Item>
  );

  const [autoCompleteResult, setAutoCompleteResult] = useState<string[]>
([]);

  const onWebsiteChange = (value: string) => {
    if (!value) {
      setAutoCompleteResult([]);
    } else {
      setAutoCompleteResult(['.com', '.org', '.net'].map((domain) =>
`${value}${domain}`));
    }
  };

  const websiteOptions = autoCompleteResult.map((website) => ({
    label: website,
    value: website,
  }));

  return (
    <Form
      {...formItemLayout}
      form={form}
      name="register"
      onFinish={onFinish}
      initialValues={{ residence: ['zhejiang', 'hangzhou', 'xihu'], prefix:
'86' }}
      style={{ maxWidth: 600 }}
      scrollToFirstError
    >
      <Form.Item
        name="email"
        label="E-mail"
```

```
          rules={[
            {
              type: 'email',
              message: 'The input is not valid E-mail!',
            },
            {
              required: true,
              message: 'Please input your E-mail!',
            },
          ]}
        >
          <Input />
        </Form.Item>

        <Form.Item
          name="password"
          label="Password"
          rules={[
            {
              required: true,
              message: 'Please input your password!',
            },
          ]}
          hasFeedback
        >
          <Input.Password />
        </Form.Item>

        <Form.Item
          name="confirm"
          label="Confirm Password"
          dependencies={['password']}
          hasFeedback
          rules={[
            {
              required: true,
              message: 'Please confirm your password!',
            },
            ({ getFieldValue }) => ({
              validator(_, value) {
                if (!value || getFieldValue('password') === value) {
                  return Promise.resolve();
                }
                return Promise.reject(new Error('The new password that you
entered do not match!'));
              },
            }),
          ]}
        >
          <Input.Password />
        </Form.Item>

        <Form.Item
          name="nickname"
          label="Nickname"
          tooltip="What do you want others to call you?"
          rules={[{ required: true, message: 'Please input your nickname!',
  whitespace: true }]}
        >
          <Input />
        </Form.Item>

        <Form.Item
          name="residence"
          label="Habitual Residence"
          rules={[
            { type: 'array', required: true, message: 'Please select your
  habitual residence!' },
          ]}
        >
          <Cascader options={residences} />
        </Form.Item>

        <Form.Item
          name="phone"
          label="Phone Number"
          rules={[{ required: true, message: 'Please input your phone
  number!' }]}
        >
          <Input addonBefore={prefixSelector} style={{ width: '100%' }} />
        </Form.Item>

        <Form.Item
          name="donation"
          label="Donation"
          rules={[{ required: true, message: 'Please input donation amount!'
  }]}
        >
          <InputNumber addonAfter={suffixSelector} style={{ width: '100%' }}
  />
        </Form.Item>
        <Form.Item
          name="website"
          label="Website"
          rules={[{ required: true, message: 'Please input website!' }]}
        >
          <AutoComplete options={websiteOptions} onChange={onWebsiteChange}
  placeholder="website">
            <Input />
          </AutoComplete>
        </Form.Item>

        <Form.Item
          name="intro"
          label="Intro"
          rules={[{ required: true, message: 'Please input Intro' }]}
        >
          <Input.TextArea showCount maxLength={100} />
        </Form.Item>

        <Form.Item
          name="gender"
          label="Gender"
          rules={[{ required: true, message: 'Please select gender!' }]}
        >
          <Select placeholder="select your gender">
            <Option value="male">Male</Option>
            <Option value="female">Female</Option>
            <Option value="other">Other</Option>
          </Select>
        </Form.Item>

        <Form.Item label="Captcha" extra="We must make sure that your are a
  human.">
          <Row gutter={8}>
            <Col span={12}>
              <Form.Item
                name="captcha"
                noStyle
                rules={[{ required: true, message: 'Please input the captcha
  you got!' }]}
              >
                <Input />
              </Form.Item>
            </Col>
            <Col span={12}>
              <Button>Get captcha</Button>
            </Col>
          </Row>
        </Form.Item>

        <Form.Item
          name="agreement"
          valuePropName="checked"
          rules={[
            {
              validator: (_, value) =>
                value ? Promise.resolve() : Promise.reject(new Error('Should
  accept agreement')),
            },
          ]}
          {...tailFormItemLayout}
        >
          <Checkbox>
            I have read the <a href="">agreement</a>
          </Checkbox>
        </Form.Item>
        <Form.Item {...tailFormItemLayout}>
          <Button type="primary" htmlType="submit">
            Register
          </Button>
        </Form.Item>
      </Form>
    );
  };

  export default App;
```

高级搜索

```
import React, { useState } from 'react';
import { DownOutlined } from '@ant-design/icons';
import { Button, Col, Form, Input, Row, Select, Space, theme } from 'antd';

const { Option } = Select;

const AdvancedSearchForm = () => {
  const { token } = theme.useToken();
  const [form] = Form.useForm();
  const [expand, setExpand] = useState(false);

  const formStyle: React.CSSProperties = {
    maxWidth: 'none',
    background: token.colorFillAlter,
    borderRadius: token.borderRadiusLG,
    padding: 24,
  };

  const getFields = () => {
    const count = expand ? 10 : 6;
    const children = [];
    for (let i = 0; i < count; i++) {
      children.push(
        <Col span={8} key={i}>
          {i % 3 !== 1 ? (
            <Form.Item
              name={`field-${i}`}
              label={`Field ${i}`}
              rules={[
                {
                  required: true,
                  message: 'Input something!',
                },
              ]}
            >
              <Input placeholder="placeholder" />
            </Form.Item>
          ) : (
            <Form.Item
              name={`field-${i}`}
              label={`Field ${i}`}
              rules={[
                {
                  required: true,
                  message: 'Select something!',
                },
              ]}
              initialValue="1"
            >
              <Select>
                <Option value="1">

longlonglonglonglonglonglonglonglonglonglonglonglonglonglonglonglonglonglonglongl
                </Option>
                <Option value="2">222</Option>
              </Select>
            </Form.Item>
          )}
        </Col>,
      );
    }
    return children;
  };

  const onFinish = (values: any) => {
    console.log('Received values of form: ', values);
  };

  return (
    <Form form={form} name="advanced_search" style={formStyle} onFinish=
{onFinish}>
      <Row gutter={24}>{getFields()}</Row>
      <div style={{ textAlign: 'right' }}>
        <Space size="small">
          <Button type="primary" htmlType="submit">
            Search
          </Button>
          <Button
            onClick={() => {
              form.resetFields();
            }}
          >
            Clear
          </Button>
          <a
            style={{ fontSize: 12 }}
            onClick={() => {
              setExpand(!expand);
            }}
          >
            <DownOutlined rotate={expand ? 180 : 0} /> Collapse
          </a>
        </Space>
      </div>
    </Form>
  );
};

const App: React.FC = () => {
  const { token } = theme.useToken();

  const listStyle: React.CSSProperties = {
    lineHeight: '200px',
    textAlign: 'center',
    background: token.colorFillAlter,
    borderRadius: token.borderRadiusLG,
    marginTop: 16,
  };

  return (
    <>
      <AdvancedSearchForm />
      <div style={listStyle}>Search Result List</div>
    </>
  );
};

export default App;
```

弹出层中的新建表单

```
import React, { useState } from 'react';
import { Button, Form, Input, Modal, Radio } from 'antd';

interface Values {
  title?: string;
  description?: string;
  modifier?: string;
}

const App: React.FC = () => {
  const [form] = Form.useForm();
  const [formValues, setFormValues] = useState<Values>();
  const [open, setOpen] = useState(false);

  const onCreate = (values: Values) => {
    console.log('Received values of form: ', values);
    setFormValues(values);
    setOpen(false);
  };

  return (
    <>
      <Button type="primary" onClick={() => setOpen(true)}>
        New Collection
      </Button>
      <pre>{JSON.stringify(formValues, null, 2)}</pre>
      <Modal
        open={open}
        title="Create a new collection"
        okText="Create"
        cancelText="Cancel"
        okButtonProps={{ autoFocus: true, htmlType: 'submit' }}
        onCancel={() => setOpen(false)}
        destroyOnClose
        modalRender={(dom) => (
          <Form
            layout="vertical"
            form={form}
            name="form_in_modal"
            initialValues={{ modifier: 'public' }}
            clearOnDestroy
            onFinish={(values) => onCreate(values)}
          >
            {dom}
          </Form>
        )}
      >
        <Form.Item
          name="title"
          label="Title"
          rules={[{ required: true, message: 'Please input the title of
collection!' }]}
        >
          <Input />
        </Form.Item>
        <Form.Item name="description" label="Description">
          <Input type="textarea" />
        </Form.Item>
        <Form.Item name="modifier" className="collection-create-form_last-
form-item">
          <Radio.Group>
            <Radio value="public">Public</Radio>
            <Radio value="private">Private</Radio>
          </Radio.Group>
        </Form.Item>
      </Modal>
    </>
  );
};

export default App;
```

时间类控件

```
import React from 'react';
import { Button, DatePicker, Form, TimePicker } from 'antd';
```

```
const { RangePicker } = DatePicker;

const formItemLayout = {
  labelCol: {
    xs: { span: 24 },
    sm: { span: 8 },
  },
  wrapperCol: {
    xs: { span: 24 },
    sm: { span: 16 },
  },
};

const config = {
  rules: [{ type: 'object' as const, required: true, message: 'Please
select time!' }],
};

const rangeConfig = {
  rules: [{ type: 'array' as const, required: true, message: 'Please select
time!' }],
};

const onFinish = (fieldsValue: any) => {
  // Should format date value before submit.
  const rangeValue = fieldsValue['range-picker'];
  const rangeTimeValue = fieldsValue['range-time-picker'];
  const values = {
    ...fieldsValue,
    'date-picker': fieldsValue['date-picker'].format('YYYY-MM-DD'),
    'date-time-picker': fieldsValue['date-time-picker'].format('YYYY-MM-DD
HH:mm:ss'),
    'month-picker': fieldsValue['month-picker'].format('YYYY-MM'),
    'range-picker': [rangeValue[0].format('YYYY-MM-DD'),
rangeValue[1].format('YYYY-MM-DD')],
    'range-time-picker': [
      rangeTimeValue[0].format('YYYY-MM-DD HH:mm:ss'),
      rangeTimeValue[1].format('YYYY-MM-DD HH:mm:ss'),
    ],
    'time-picker': fieldsValue['time-picker'].format('HH:mm:ss'),
  };
  console.log('Received values of form: ', values);
};

const App: React.FC = () => (
  <Form
    name="time_related_controls"
    {...formItemLayout}
    onFinish={onFinish}
    style={{ maxWidth: 600 }}
  >
    <Form.Item name="date-picker" label="DatePicker" {...config}>
      <DatePicker />
    </Form.Item>
    <Form.Item name="date-time-picker" label="DatePicker[showTime]"
{...config}>
      <DatePicker showTime format="YYYY-MM-DD HH:mm:ss" />
    </Form.Item>
    <Form.Item name="month-picker" label="MonthPicker" {...config}>
      <DatePicker picker="month" />
    </Form.Item>
    <Form.Item name="range-picker" label="RangePicker" {...rangeConfig}>
      <RangePicker />
    </Form.Item>
    <Form.Item name="range-time-picker" label="RangePicker[showTime]"
{...rangeConfig}>
      <RangePicker showTime format="YYYY-MM-DD HH:mm:ss" />
    </Form.Item>
    <Form.Item name="time-picker" label="TimePicker" {...config}>
      <TimePicker />
    </Form.Item>
    <Form.Item label={null}>
      <Button type="primary" htmlType="submit">
        Submit
      </Button>
    </Form.Item>
  </Form>
);

export default App;
```

自行处理表单数据

```
import React, { useState } from 'react';
import type { InputNumberProps } from 'antd';
import { Form, InputNumber } from 'antd';

type ValidateStatus = Parameters<typeof Form.Item>[0]['validateStatus'];

const validatePrimeNumber = (
  number: number,
): {
  validateStatus: ValidateStatus;
  errorMsg: string | null;
} => {
  if (number === 11) {
    return {
      validateStatus: 'success',
      errorMsg: null,
    };
  }
  return {
    validateStatus: 'error',
    errorMsg: 'The prime between 8 and 12 is 11!',
  };
};

const formItemLayout = {
  labelCol: { span: 7 },
  wrapperCol: { span: 12 },
};

const tips =
  'A prime is a natural number greater than 1 that has no positive divisors
other than 1 and itself.';

const App: React.FC = () => {
  const [number, setNumber] = useState<{
    value: number;
    validateStatus?: ValidateStatus;
    errorMsg?: string | null;
  }>({ value: 11 });

  const onNumberChange: InputNumberProps['onChange'] = (value) => {
    setNumber({
      ...validatePrimeNumber(value as number),
      value: value as number,
    });
  };

  return (
    <Form style={{ maxWidth: 600 }}>
      <Form.Item
        {...formItemLayout}
        label="Prime between 8 & 12"
        validateStatus={number.validateStatus}
        help={number.errorMsg || tips}
      >
        <InputNumber min={8} max={12} value={number.value} onChange=
{onNumberChange} />
      </Form.Item>
    </Form>
  );
};

export default App;
```

自定义校验

```
import React from 'react';
import { SmileOutlined } from '@ant-design/icons';
import {
  Cascader,
  DatePicker,
  Form,
  Input,
  InputNumber,
  Mentions,
  Select,
  TimePicker,
  TreeSelect,
} from 'antd';

const { Option } = Select;

const formItemLayout = {
  labelCol: {
    xs: { span: 24 },
    sm: { span: 6 },
  },
  wrapperCol: {
    xs: { span: 24 },
    sm: { span: 14 },
  },
};

const App: React.FC = () => (
  <Form {...formItemLayout} style={{ maxWidth: 600 }}>
    <Form.Item
      label="Fail"
      validateStatus="error"
      help="Should be combination of numbers & alphabets"
    >
```

```
      <Input placeholder="unavailable choice" id="error" />
    </Form.Item>

    <Form.Item label="Warning" validateStatus="warning">
      <Input placeholder="Warning" id="warning" prefix={<SmileOutlined />}
/>
    </Form.Item>

    <Form.Item
      label="Validating"
      hasFeedback
      validateStatus="validating"
      help="The information is being validated..."
    >
      <Input placeholder="I'm the content is being validated"
id="validating" />
    </Form.Item>

    <Form.Item label="Success" hasFeedback validateStatus="success">
      <Input placeholder="I'm the content" id="success" />
    </Form.Item>

    <Form.Item label="Warning" hasFeedback validateStatus="warning">
      <Input placeholder="Warning" id="warning2" />
    </Form.Item>

    <Form.Item
      label="Fail"
      hasFeedback
      validateStatus="error"
      help="Should be combination of numbers & alphabets"
    >
      <Input placeholder="unavailable choice" id="error2" />
    </Form.Item>

    <Form.Item label="Success" hasFeedback validateStatus="success">
      <DatePicker style={{ width: '100%' }} />
    </Form.Item>

    <Form.Item label="Warning" hasFeedback validateStatus="warning">
      <TimePicker style={{ width: '100%' }} />
    </Form.Item>

    <Form.Item label="Error" hasFeedback validateStatus="error">
      <DatePicker.RangePicker style={{ width: '100%' }} />
    </Form.Item>
    <Form.Item label="Error" hasFeedback validateStatus="error">
      <Select placeholder="I'm Select" allowClear>
        <Option value="1">Option 1</Option>
        <Option value="2">Option 2</Option>
        <Option value="3">Option 3</Option>
      </Select>
    </Form.Item>

    <Form.Item
      label="Validating"
      hasFeedback
      validateStatus="error"
      help="Something breaks the rule."
    >
      <Cascader placeholder="I'm Cascader" options={[{ value: 'xx', label:
'xx' }]} allowClear />
    </Form.Item>

    <Form.Item label="Warning" hasFeedback validateStatus="warning"
help="Need to be checked">
      <TreeSelect
        placeholder="I'm TreeSelect"
        treeData={[{ value: 'xx', label: 'xx' }]}
        allowClear
      />
    </Form.Item>

    <Form.Item label="inline" style={{ marginBottom: 0 }}>
      <Form.Item
        validateStatus="error"
        help="Please select right date"
        style={{ display: 'inline-block', width: 'calc(50% - 12px)' }}
      >
        <DatePicker />
      </Form.Item>
      <span
        style={{ display: 'inline-block', width: '24px', lineHeight:
'32px', textAlign: 'center' }}
      >
        -
      </span>
      <Form.Item style={{ display: 'inline-block', width: 'calc(50% -
12px)' }}>
        <DatePicker />
      </Form.Item>
    </Form.Item>

    <Form.Item label="Success" hasFeedback validateStatus="success">
      <InputNumber style={{ width: '100%' }} />
    </Form.Item>

    <Form.Item label="Success" hasFeedback validateStatus="success">
      <Input allowClear placeholder="with allowClear" />
    </Form.Item>

    <Form.Item label="Warning" hasFeedback validateStatus="warning">
      <Input.Password placeholder="with input password" />
    </Form.Item>

    <Form.Item label="Error" hasFeedback validateStatus="error">
      <Input.Password allowClear placeholder="with input password and
allowClear" />
    </Form.Item>

    <Form.Item label="Success" hasFeedback validateStatus="success">
      <Input.OTP />
    </Form.Item>
    <Form.Item label="Warning" hasFeedback validateStatus="warning">
      <Input.OTP />
    </Form.Item>

    <Form.Item label="Error" hasFeedback validateStatus="error">
      <Input.OTP />
    </Form.Item>

    <Form.Item label="Fail" validateStatus="error" hasFeedback>
      <Mentions />
    </Form.Item>

    <Form.Item label="Fail" validateStatus="error" hasFeedback help="Should
have something">
      <Input.TextArea allowClear showCount />
    </Form.Item>
  </Form>
);

export default App;
```

动态校验规则

```
import React, { useEffect, useState } from 'react';
import { Button, Checkbox, Form, Input } from 'antd';

const formItemLayout = {
  labelCol: { span: 4 },
  wrapperCol: { span: 8 },
};

const formTailLayout = {
  labelCol: { span: 4 },
  wrapperCol: { span: 8, offset: 4 },
};

const App: React.FC = () => {
  const [form] = Form.useForm();
  const [checkNick, setCheckNick] = useState(false);

  useEffect(() => {
    form.validateFields(['nickname']);
  }, [checkNick, form]);

  const onCheckboxChange = (e: { target: { checked: boolean } }) => {
    setCheckNick(e.target.checked);
  };

  const onCheck = async () => {
    try {
      const values = await form.validateFields();
      console.log('Success:', values);
    } catch (errorInfo) {
      console.log('Failed:', errorInfo);
    }
  };

  return (
    <Form form={form} name="dynamic_rule" style={{ maxWidth: 600 }}>
      <Form.Item
        {...formItemLayout}
        name="username"
        label="Name"
        rules={[{ required: true, message: 'Please input your name' }]}
      >
        <Input placeholder="Please input your name" />
      </Form.Item>
      <Form.Item
        {...formItemLayout}
        name="nickname"
        label="Nickname"
        rules={[{ required: checkNick, message: 'Please input your
nickname' }]}
      >
        <Input placeholder="Please input your nickname" />
      </Form.Item>
      <Form.Item {...formTailLayout}>
        <Checkbox checked={checkNick} onChange={onCheckboxChange}>
          Nickname is required
        </Checkbox>
      </Form.Item>
      <Form.Item {...formTailLayout}>
        <Button type="primary" onClick={onCheck}>
          Check
        </Button>
      </Form.Item>
    </Form>
  );
};

export default App;
```

校验与更新依赖

```
import React from 'react';
import { Alert, Form, Input, Typography } from 'antd';

const App: React.FC = () => {
  const [form] = Form.useForm();
  return (
    <Form
      form={form}
      name="dependencies"
      autoComplete="off"
      style={{ maxWidth: 600 }}
      layout="vertical"
    >
      <Alert message=" Try modify `Password2` and then modify `Password`"
type="info" showIcon />

      <Form.Item label="Password" name="password" rules={[{ required: true
}]}>
        <Input />
      </Form.Item>
      {/* Field */}
      <Form.Item
        label="Confirm Password"
        name="password2"
        dependencies={['password']}
        rules={[
          {
            required: true,
          },
          ({ getFieldValue }) => ({
            validator(_, value) {
              if (!value || getFieldValue('password') === value) {
                return Promise.resolve();
              }
              return Promise.reject(new Error('The new password that you
entered do not match!'));
            },
          }),
        ]}
      >
        <Input />
      </Form.Item>

      {/* Render Props */}
      <Form.Item noStyle dependencies={['password2']}>
        {() => (
          <Typography>
            <p>
              Only Update when <code>password2</code> updated:
            </p>
            <pre>{JSON.stringify(form.getFieldsValue(), null, 2)}</pre>
          </Typography>
        )}
      </Form.Item>
    </Form>
  );
};

export default App;
```

滑动到错误字段

```
import React from 'react';
import { Button, Flex, Form, Input, Select } from 'antd';

const App = () => {
  const [form] = Form.useForm();

  return (
    <Form
      form={form}
      scrollToFirstError={{ behavior: 'instant', block: 'end', focus: true
}}
      style={{ paddingBlock: 32 }}
      labelCol={{ span: 6 }}
      wrapperCol={{ span: 14 }}
    >
      <Form.Item wrapperCol={{ offset: 6 }}>
        <Button onClick={() => form.scrollToField('bio')}>Scroll to
Bio</Button>
      </Form.Item>

      <Form.Item name="username" label="UserName" rules={[{ required: true
}]}>
        <Input />
      </Form.Item>

      <Form.Item label="Occupation" name="occupation">
        <Select
          options={[
            { label: 'Designer', value: 'designer' },
            { label: 'Developer', value: 'developer' },
            { label: 'Product Manager', value: 'product-manager' },
          ]}
        />
      </Form.Item>

      <Form.Item name="motto" label="Motto">
        <Input.TextArea rows={4} />
      </Form.Item>

      <Form.Item name="bio" label="Bio" rules={[{ required: true }]}>
        <Input.TextArea rows={6} />
      </Form.Item>

      <Form.Item wrapperCol={{ offset: 6 }}>
        <Flex gap="small">
          <Button type="primary" htmlType="submit">
            Submit
          </Button>
          <Button danger onClick={() => form.resetFields()}>
            Reset
          </Button>
        </Flex>
      </Form.Item>
    </Form>
  );
};

export default App;
```

校验其他组件

```
import React from 'react';
import { InboxOutlined, UploadOutlined } from '@ant-design/icons';
import {
  Button,
  Checkbox,
  Col,
  ColorPicker,
  Form,
  InputNumber,
  Radio,
  Rate,
  Row,
  Select,
  Slider,
  Space,
  Switch,
  Upload,
} from 'antd';

const { Option } = Select;

const formItemLayout = {
  labelCol: { span: 6 },
  wrapperCol: { span: 14 },
};

const normFile = (e: any) => {
  console.log('Upload event:', e);
  if (Array.isArray(e)) {
    return e;
  }
  return e?.fileList;
};

const onFinish = (values: any) => {
  console.log('Received values of form: ', values);
};

const App: React.FC = () => (
  <Form
    name="validate_other"
    {...formItemLayout}
    onFinish={onFinish}
    initialValues={{
      'input-number': 3,
      'checkbox-group': ['A', 'B'],
      rate: 3.5,
      'color-picker': null,
    }}
    style={{ maxWidth: 600 }}
  >
    <Form.Item label="Plain Text">
      <span className="ant-form-text">China</span>
    </Form.Item>
    <Form.Item
      name="select"
      label="Select"
      hasFeedback
      rules={[{ required: true, message: 'Please select your country!' }]}
    >
      <Select placeholder="Please select a country">
        <Option value="china">China</Option>
        <Option value="usa">U.S.A</Option>
      </Select>
    </Form.Item>

    <Form.Item
      name="select-multiple"
      label="Select[multiple]"
      rules={[{ required: true, message: 'Please select your favourite
colors!', type: 'array' }]}
    >
      <Select mode="multiple" placeholder="Please select favourite colors">
        <Option value="red">Red</Option>
        <Option value="green">Green</Option>
        <Option value="blue">Blue</Option>
      </Select>
    </Form.Item>

    <Form.Item label="InputNumber">
      <Form.Item name="input-number" noStyle>
        <InputNumber min={1} max={10} />
      </Form.Item>
      <span className="ant-form-text" style={{ marginInlineStart: 8 }}>
        machines
      </span>
    </Form.Item>

    <Form.Item name="switch" label="Switch" valuePropName="checked">
      <Switch />
    </Form.Item>

    <Form.Item name="slider" label="Slider">
      <Slider
        marks={{
          0: 'A',
          20: 'B',
          40: 'C',
          60: 'D',
          80: 'E',
          100: 'F',
        }}
      />
    </Form.Item>

    <Form.Item name="radio-group" label="Radio.Group">
      <Radio.Group>
        <Radio value="a">item 1</Radio>
        <Radio value="b">item 2</Radio>
        <Radio value="c">item 3</Radio>
      </Radio.Group>
    </Form.Item>

    <Form.Item
      name="radio-button"
      label="Radio.Button"
      rules={[{ required: true, message: 'Please pick an item!' }]}
    >
      <Radio.Group>
        <Radio.Button value="a">item 1</Radio.Button>
        <Radio.Button value="b">item 2</Radio.Button>
        <Radio.Button value="c">item 3</Radio.Button>
      </Radio.Group>
    </Form.Item>

    <Form.Item name="checkbox-group" label="Checkbox.Group">
      <Checkbox.Group>
        <Row>
          <Col span={8}>
            <Checkbox value="A" style={{ lineHeight: '32px' }}>
              A
            </Checkbox>
          </Col>
          <Col span={8}>
            <Checkbox value="B" style={{ lineHeight: '32px' }} disabled>
              B
            </Checkbox>
          </Col>
          <Col span={8}>
            <Checkbox value="C" style={{ lineHeight: '32px' }}>
              C
            </Checkbox>
          </Col>
          <Col span={8}>
            <Checkbox value="D" style={{ lineHeight: '32px' }}>
              D
            </Checkbox>
          </Col>
          <Col span={8}>
            <Checkbox value="E" style={{ lineHeight: '32px' }}>
              E
            </Checkbox>
          </Col>
          <Col span={8}>
            <Checkbox value="F" style={{ lineHeight: '32px' }}>
              F
            </Checkbox>
          </Col>
        </Row>
      </Checkbox.Group>
    </Form.Item>

    <Form.Item name="rate" label="Rate">
      <Rate />
    </Form.Item>

    <Form.Item
      name="upload"
      label="Upload"
      valuePropName="fileList"
      getValueFromEvent={normFile}
      extra="longgggggggggggggggggggggggggggggggggg"
    >
```

```
      <Upload name="logo" action="/upload.do" listType="picture">
        <Button icon={<UploadOutlined />}>Click to upload</Button>
      </Upload>
    </Form.Item>
    <Form.Item label="Dragger">
      <Form.Item name="dragger" valuePropName="fileList" getValueFromEvent=
{normFile} noStyle>
        <Upload.Dragger name="files" action="/upload.do">
          <p className="ant-upload-drag-icon">
            <InboxOutlined />
          </p>
          <p className="ant-upload-text">Click or drag file to this area to
upload</p>
          <p className="ant-upload-hint">Support for a single or bulk
upload.</p>
        </Upload.Dragger>
      </Form.Item>
    </Form.Item>
    <Form.Item
      name="color-picker"
      label="ColorPicker"
      rules={[{ required: true, message: 'color is required!' }]}
    >
      <ColorPicker />
    </Form.Item>

    <Form.Item wrapperCol={{ span: 12, offset: 6 }}>
      <Space>
        <Button type="primary" htmlType="submit">
          Submit
        </Button>
        <Button htmlType="reset">reset</Button>
      </Space>
    </Form.Item>
  </Form>
);

export default App;
```

getValueProps + normalize

```
import React from 'react';
import type { FormProps } from 'antd';
import { Button, DatePicker, Form } from 'antd';
import dayjs from 'dayjs';
```

```
const dateTimestamp = dayjs('2024-01-01').valueOf();

type FieldType = {
  date?: string;
};

const onFinish: FormProps<FieldType>['onFinish'] = (values) => {
  console.log('Success:', values);
};

const App: React.FC = () => (
  <Form
    name="getValueProps"
    labelCol={{ span: 8 }}
    wrapperCol={{ span: 16 }}
    style={{ maxWidth: 600 }}
    initialValues={{ date: dateTimestamp }}
    onFinish={onFinish}
    autoComplete="off"
  >
    <Form.Item<FieldType>
      label="Date"
      name="date"
      rules={[{ required: true }]}
      getValueProps={(value) => ({ value: value && dayjs(Number(value)) })}
      normalize={(value) => value && `${dayjs(value).valueOf()}`}
    >
      <DatePicker />
    </Form.Item>

    <Form.Item label={null}>
      <Button type="primary" htmlType="submit">
        Submit
      </Button>
    </Form.Item>
  </Form>
);

export default App;
```

Disabled Input Debug

Debug

```
import React from 'react';
import { Form, Input } from 'antd';

const App: React.FC = () => (
  <Form style={{ maxWidth: 600 }}>
    <Form.Item label="Normal0">
      <Input placeholder="unavailable choice" value="Buggy!" />
    </Form.Item>
    <Form.Item label="Fail0" validateStatus="error" help="Buggy!">
      <Input placeholder="unavailable choice" value="Buggy!" />
    </Form.Item>
    <Form.Item label="FailDisabled0" validateStatus="error" help="Buggy!">
      <Input placeholder="unavailable choice" disabled value="Buggy!" />
    </Form.Item>
    <Form.Item label="Normal1">
      <Input placeholder="unavailable choice" value="Buggy!" />
    </Form.Item>
    <Form.Item label="Fail1" validateStatus="error" help="Buggy!">
      <Input placeholder="unavailable choice" value="Buggy!" />
    </Form.Item>
    <Form.Item label="FailDisabled1" validateStatus="error" help="Buggy!">
      <Input placeholder="unavailable choice" disabled value="Buggy!" />
    </Form.Item>
    <Form.Item label="Normal2">
      <Input placeholder="unavailable choice" addonBefore="Buggy!" />
    </Form.Item>
    <Form.Item label="Fail2" validateStatus="error" help="Buggy!">
      <Input placeholder="unavailable choice" addonBefore="Buggy!" />
    </Form.Item>
    <Form.Item label="FailDisabled2" validateStatus="error" help="Buggy!">
      <Input placeholder="unavailable choice" disabled addonBefore="Buggy!"
/>
    </Form.Item>
    <Form.Item label="Normal3">
      <Input placeholder="unavailable choice" prefix="人民币" value="50" />
    </Form.Item>
    <Form.Item label="Fail3" validateStatus="error" help="Buggy!">
      <Input placeholder="unavailable choice" prefix="人民币" value="50" />
    </Form.Item>
    <Form.Item label="FailDisabled3" validateStatus="error" help="Buggy!">
      <Input placeholder="unavailable choice" disabled prefix="人民币"
value="50" />
    </Form.Item>
  </Form>
);
```

```
export default App;
```

测试 label 省略

Debug

```
import React from 'react';
import { Form, Input, Typography } from 'antd';

const App: React.FC = () => (
  <Form
    name="label-ellipsis"
    labelCol={{ span: 8 }}
    wrapperCol={{ span: 16 }}
    style={{ maxWidth: 600 }}
  >
    <Form.Item
      label={
        <Typography.Text ellipsis>
          longtextlongtextlongtextlongtextlongtextlongtextlongtext
        </Typography.Text>
      }
      name="username"
    >
      <Input />
    </Form.Item>

    <Form.Item
      label={
        <Typography.Text ellipsis>
          longtext longtext longtext longtext longtext longtext longtext
        </Typography.Text>
      }
      name="password"
    >
      <Input.Password />
    </Form.Item>
  </Form>
);

export default App;
```

测试特殊 col 24 用法

Debug

```tsx
import React from 'react';
import { Button, Divider, Form, Input, Select } from 'antd';

const sharedItem = (
  <Form.Item
    label={
      <a
        href="https://github.com/ant-design/ant-design/issues/36459"
        target="_blank"
        rel="noreferrer"
      >
        #36459
      </a>
    }
    initialValue={['bamboo']}
    name="select"
    style={{ boxShadow: '0 0 3px red' }}
  >
    <Select
      style={{ width: '70%' }}
      mode="multiple"
      options={[
        { label: 'Bamboo', value: 'bamboo' },
        { label: 'Little', value: 'little' },
        { label: 'Light', value: 'light' },
      ]}
    />
  </Form.Item>
);

const App: React.FC = () => {
  const onFinish = (values: any) => {
    console.log('Success:', values);
  };

  const onFinishFailed = (errorInfo: any) => {
    console.log('Failed:', errorInfo);
  };

  return (
    <>
      <Form
        name="col-24-debug"
        labelCol={{ span: 24 }}
        wrapperCol={{ span: 24 }}
        initialValues={{ remember: true }}
```

```
          onFinish={onFinish}
          onFinishFailed={onFinishFailed}
          style={{ maxWidth: 600 }}
          autoComplete="off"
        >
          <Form.Item
            label="Username"
            name="username"
            rules={[{ required: true, message: 'Please input your username!'
}]}
          >
            <Input />
          </Form.Item>

          <Form.Item
            label="Password"
            name="password"
            rules={[{ required: true, message: 'Please input your password!'
}]}
          >
            <Input.Password />
          </Form.Item>

          {sharedItem}

          <Form.Item>
            <Button type="primary" htmlType="submit">
              Submit
            </Button>
          </Form.Item>
        </Form>
        <Form
          name="responsive"
          labelCol={{ sm: 24, xl: 24 }}
          wrapperCol={{ sm: 24, xl: 24 }}
          initialValues={{ remember: true }}
          onFinish={onFinish}
          onFinishFailed={onFinishFailed}
          autoComplete="off"
        >
          <Form.Item
            label="Username"
            name="username"
            rules={[{ required: true, message: 'Please input your username!'
}]}
          >
            <Input />
          </Form.Item>

          <Form.Item
            label="Password"
            name="password"
            rules={[{ required: true, message: 'Please input your password!'
  }]}
          >
            <Input.Password />
          </Form.Item>

          <Form.Item>
            <Button type="primary" htmlType="submit">
              Submit
            </Button>
          </Form.Item>
        </Form>

        <Divider />

        <Form layout="vertical">
          {sharedItem}

          <Form.Item label="col12" name="col12" labelCol={{ span: 12 }}
  wrapperCol={{ span: 12 }}>
            <Input />
          </Form.Item>
        </Form>
      </>
    );
  };

  export default App;
```

引用字段

Debug

```
import React from 'react';
import type { InputRef } from 'antd';
import { Button, Form, Input } from 'antd';

const App: React.FC = () => {
  const [form] = Form.useForm();
  const ref = React.useRef<InputRef>(null);
  return (
    <Form form={form} initialValues={{ list: ['light'] }} style={{
maxWidth: 600 }}>
      <Form.Item name="test" label="test">
        <Input ref={ref} />
      </Form.Item>

      <Form.List name="list">
        {(fields) =>
          fields.map((field) => (
            <Form.Item {...field} key={field.key}>
              <Input ref={ref} />
            </Form.Item>
          ))
        }
      </Form.List>

      <Button
        htmlType="button"
        onClick={() => {
          form.getFieldInstance('test').focus();
        }}
      >
        Focus Form.Item
      </Button>
      <Button
        onClick={() => {
          form.getFieldInstance(['list', 0]).focus();
        }}
      >
        Focus Form.List
      </Button>
    </Form>
  );
};

export default App;
```

Custom feedback icons

Debug

```
import React from 'react';
import { AlertFilled, CloseSquareFilled } from '@ant-design/icons';
import { Button, Form, Input, Tooltip } from 'antd';
import { createStyles, css } from 'antd-style';
import uniqueId from 'lodash/uniqueId';

const useStyle = createStyles(() => ({
  'custom-feedback-icons': css`
    .ant-form-item-feedback-icon {
      pointer-events: all;
    }
  `,
}));

const App: React.FC = () => {
  const [form] = Form.useForm();
  const { styles } = useStyle();

  return (
    <Form
      name="custom-feedback-icons"
      form={form}
      style={{ maxWidth: 600 }}
      feedbackIcons={({ errors }) => ({
        error: (
          <Tooltip
            key="tooltipKey"
            title={errors?.map((error) => <div key={uniqueId()}>{error}
</div>)}
            color="red"
          >
            <CloseSquareFilled />
          </Tooltip>
        ),
      })}
    >
      <Form.Item
        name="custom-feedback-test-item"
        label="Test"
        className={styles['custom-feedback-icons']}
        rules={[{ required: true, type: 'email' }, { min: 10 }]}
        help=""
        hasFeedback
      >
        <Input />
      </Form.Item>
      <Form.Item
        name="custom-feedback-test-item2"
        label="Test"
```

```
          className={styles['custom-feedback-icons']}
          rules={[{ required: true, type: 'email' }, { min: 10 }]}
          help=""
          hasFeedback={{
            icons: ({ errors }) => ({
              error: (
                <Tooltip
                  key="tooltipKey"
                  title={errors?.map((error) => <div key={uniqueId()}>{error}
</div>)}
                  color="pink"
                >
                  <AlertFilled />
                </Tooltip>
              ),
              success: false,
            }),
          }}
        >
          <Input />
        </Form.Item>
        <Form.Item>
          <Button htmlType="submit">Submit</Button>
        </Form.Item>
      </Form>
    );
};

export default App;
```

组件 Token

Debug

```
import React from 'react';
import { ConfigProvider, Form, Input } from 'antd';

const App: React.FC = () => (
  <ConfigProvider
    theme={{
      components: {
        Form: {
          labelRequiredMarkColor: 'pink',
          labelColor: 'green',
          labelFontSize: 16,
          labelHeight: 34,
          labelColonMarginInlineStart: 4,
          labelColonMarginInlineEnd: 12,
          itemMarginBottom: 18,
          inlineItemMarginBottom: 18,
        },
      },
    }}
  >
    <Form
      name="component-token"
      labelCol={{ span: 8 }}
      wrapperCol={{ span: 16 }}
      style={{ maxWidth: 600 }}
      initialValues={{ remember: true }}
      autoComplete="off"
    >
      <Form.Item
        label="Username"
        name="username"
        rules={[{ required: true, message: 'Please input your username!'
}]}
      >
        <Input />
      </Form.Item>

      <Form.Item
        label="Password"
        name="password"
        rules={[{ required: true, message: 'Please input your password!'
}]}
      >
        <Input.Password />
      </Form.Item>
    </Form>
  </ConfigProvider>
);

export default App;
```

API

通用属性参考：通用属性

Form

参数	说明	类型	默认值	

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	
disabled	设置表单组件禁用，仅对 antd 组件有效	boolean	false	4
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	-	
feedbackIcons	当 `Form.Item` 有 `hasFeedback` 属性时可以自定义图标	FeedbackIcons	-	5
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
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。你可以通过 `getFieldsValue(true)` 来获取保留字段值	boolean	true	4

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` | ((label: ReactNode, info: { required: boolean }) => ReactNode)	true	r 5
scrollToFirstError	提交失败自动滚动到第一个错误字段	boolean | [Options](#) | { focus: boolean }	false	f
size	设置字段组件的尺寸（仅限 antd 组件）	`small` | `middle` | `large`	-	
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
variant	表单内控件变体	`outlined` | `borderless` | `filled` | `underlined`	`outlined`	5 u 5
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)	-	
clearOnDestroy	当表单被卸载时清空表单值	boolean	false	5

> *支持原生 form 除 `onSubmit` 外的所有属性。*

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 (不建议通过 `getValueProps` 生成动态函数 prop，请直接将其传递给子组件)	(value: any) => Record<string, 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 此外，它还可以通过 Icons 属性获取反馈图标。	boolean | { icons: FeedbackIcons }	false	icons: 5.9.0

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，当不需要 label 又需要与冒号对齐，可以设为 null	ReactNode	-	null: 5.22.0
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，查看详情	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。当自身没有 `validateStatus` 而父元素存在有 `validateStatus` 的	boolean	false	

	Form.Item 会继承父元素的 `validateStatus`			
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
validateDebounce	设置防抖，延迟毫秒数后进行校验	number	-	5.9.0
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。注意：Switch、Checkbox 的 valuePropName 应该是 checked，否则无法获取这个两个组件的	string	`value`	

	值。该属性为 getValueProps 的封装，自定义 getValueProps 后会失效			
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	-	
layout	表单项布局	horizontal | vertical	-	5.18.0

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。

FeedbackIcons

`({ status: ValidateStatus, errors: ReactNode, warnings: ReactNode }) => Record<ValidateStatus, ReactNode>`

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，要注意 Form.Item 里包裹的子组件必须由函数返回，否则 `shouldUpdate` 不会起作用：

相关issue：[#34500](#34500)

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

你可以参考[示例](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](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"
    rules={[{ required: true, message: '${another} is required' }]}
  >
    <Input />
  </Form.Item>
  <Form.Item
    messageVariables={{ label: 'good' }}
    label={<span>user</span>}
    rules={[{ required: true, message: '${label} is required' }]}
  >
    <Input />
  </Form.Item>
</Form>
```

自 `5.20.2` 起，当你希望不要转译 `${}` 时，你可以通过 `\\${}` 来略过：

```
{ required: true, message: '${label} is convert, \\${label} is not convert' }

// good is convert, ${label} is not convert
```

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。List 本身也是字段，因而 `getFieldsValue()` 默认会返回 List 下所有值，你可以通过参数改变这一行为	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	GetFieldsValue	
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 | { focus: boolean }) => void	focus: 5.24.0
setFields	设置一组字段状态	(fields: FieldData[]) => void	

setFieldValue	设置表单的值（该值将直接传入 form store 中并且**重置错误信息**。如果你不希望传入对象被修改，请克隆后传入）	(name: [NamePath](), value: any) => void	4.22.0
setFieldsValue	设置表单的值（该值将直接传入 form store 中并且**重置错误信息**。如果你不希望传入对象被修改，请克隆后传入）。如果你只想修改 Form.List 中单项值，请通过 `setFieldValue` 进行指定	(values) => void	
submit	提交表单，与点击 `submit` 按钮效果相同	() => void	
validateFields	触发表单验证，设置 `recursive` 时会递归校验所有包含的路径	(nameList?: [NamePath]()[], config?: [ValidateConfig]()) => Promise	

validateFields

```
export interface ValidateConfig {
  // 5.5.0 新增。仅校验内容而不会将错误信息展示到 UI 上。
  validateOnly?: boolean;
  // 5.9.0 新增。对提供的 `nameList` 与其子路径进行递归校验。
  recursive?: boolean;
  // 5.11.0 新增。校验 dirty 的字段（touched + validated）。
  // 使用 `dirty` 可以很方便的仅校验用户操作过和被校验过的字段。
  dirty?: boolean;
}
```

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,
      }
    */
  });
```

Hooks

Form.useForm

`type Form.useForm = (): [FormInstance]`

创建 Form 实例，用于管理所有数据状态。

Form.useFormInstance

`type Form.useFormInstance = (): FormInstance`

`4.20.0` 新增，获取当前上下文正在使用的 Form 实例，常见于封装子组件消费无需透传 Form 实例：

```
const Sub = () => {
  const form = Form.useFormInstance();

  return <Button onClick={() => form.setFieldsValue({})} />;
};

export default () => {
  const [form] = Form.useForm();

  return (
    <Form form={form}>
      <Sub />
    </Form>
  );
};
```

Form.useWatch

`type Form.useWatch = (namePath: NamePath | (selector: (values: Store)) => any, formInstance?: FormInstance | WatchOptions): Value`

`5.12.0` 新增 `selector`

用于直接获取 form 中字段对应的值。通过该 Hooks 可以与诸如 `useSWR` 进行联动从而降低维护成本：

```
const Demo = () => {
  const [form] = Form.useForm();
  const userName = Form.useWatch('username', form);

  const { data: options } = useSWR(`/api/user/${userName}`, fetcher);

  return (
    <Form form={form}>
      <Form.Item name="username">
        <AutoComplete options={options} />
      </Form.Item>
    </Form>
  );
};
```

如果你的组件被包裹在 `Form.Item` 内部，你可以省略第二个参数， `Form.useWatch` 会自动找到上层最近的 `FormInstance` 。

`useWatch` 默认只监听在 Form 中注册的字段，如果需要监听非注册字段，可以通过配置 `preserve` 进行监听：

```
const Demo = () => {
  const [form] = Form.useForm();

  const age = Form.useWatch('age', { form, preserve: true });
  console.log(age);

  return (
    <div>
      <Button onClick={() => form.setFieldValue('age', 2)}>Update</Button>
      <Form form={form}>
        <Form.Item name="name">
          <Input />
        </Form.Item>
      </Form>
    </div>
  );
};
```

Form.Item.useStatus

`type Form.Item.useStatus = (): { status: ValidateStatus | undefined, errors: ReactNode[], warnings: ReactNode[] }`

`4.22.0` 新增，可用于获取当前 Form.Item 的校验状态，如果上层没有 Form.Item， `status` 将会返回 `undefined` 。 `5.4.0` 新增 `errors` 和 `warnings` ，可用于获取当前 Form.Item 的错误信息和警告信息：

```
const CustomInput = ({ value, onChange }) => {
  const { status, errors } = Form.Item.useStatus();
  return (
    <input
      value={value}
      onChange={onChange}
      className={`custom-input-${status}`}
      placeholder={(errors.length && errors[0]) || ''}
    />
  );
};

export default () => (
  <Form>
    <Form.Item name="username">
      <CustomInput />
    </Form.Item>
  </Form>
);
```

与其他获取数据的方式的区别

Form 仅会对变更的 Field 进行刷新，从而避免完整的组件刷新可能引发的性能问题。因而你无法在 render 阶段通过 `form.getFieldsValue` 来实时获取字段值，而 `useWatch` 提供了一种特定字段访问的方式，从而使得在当前组件中可以直接消费字段的值。同时，如果为了更好的渲染性能，你可以通过 Field 的 renderProps 仅更新需要更新的部分。而当当前组件更新或者 effect 都不需要消费字段值时，则可以通过 `onValuesChange` 将数据抛出，从而避免组件更新。

Interface

NamePath

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

GetFieldsValue

`getFieldsValue` 提供了多种重载方法：

getFieldsValue(nameList?: true | NamePath[], filterFunc?: FilterFunc)

当不提供 `nameList` 时，返回所有注册字段，这也包含 List 下所有的值（即便 List 下没有绑定 Item）。

当 `nameList` 为 `true` 时，返回 store 中所有的值，包含未注册字段。例如通过 `setFieldsValue` 设置了不存在的 Item 的值，也可以通过 `true` 全部获取。

当 `nameList` 为数组时，返回规定路径的值。需要注意的是， `nameList` 为嵌套数组。例如你需要某路径值应该如下：

```
// 单个路径
form.getFieldsValue([['user', 'age']]);

// 多个路径
form.getFieldsValue([
  ['user', 'age'],
  ['preset', 'account'],
]);
```

getFieldsValue({ strict?: boolean, filter?: FilterFunc })

`5.8.0` 新增接受配置参数。当 `strict` 为 `true` 时会仅匹配 Item 的值。例如 `{ list: [{ bamboo: 1, little: 2 }] }` 中，如果 List 仅绑定了 `bamboo` 字段，那么 `getFieldsValue({ strict: true })` 会只获得 `{ list: [{ bamboo: 1 }] }`。

FilterFunc

用于过滤一些字段值， `meta` 会返回字段相关信息。例如可以用来获取仅被用户修改过的值等等。

```
type FilterFunc = (meta: { touched: boolean; validating: boolean }) =>
boolean;
```

FieldData

名称	说明	类型
errors	错误信息	string[]
warnings	警告信息	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 | ReactElement	
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	

WatchOptions

名称	说明	类型	默认值	版本
form	指定 Form 实例	FormInstance	当前 context 中的 Form	5.4.0

preserve	是否监视没有对应的 `Form.Item` 的字段	boolean	false	5.4.0

主题变量（Design Token）

FAQ

Switch、Checkbox 为什么不能绑定数据?

Form.Item 默认绑定值属性到 `value` 上，而 Switch、Checkbox 等组件的值属性为 `checked` 。你可以通过 `valuePropName` 来修改绑定的值属性。

```
<Form.Item name="fieldA" valuePropName="checked">
  <Switch />
</Form.Item>
```

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 Form 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 `getFieldsValue` 在初次渲染的时候拿不到值?

`getFieldsValue` 默认返回收集的字段数据，而在初次渲染时 Form.Item 节点尚未渲染，因而无法收集到数据。你可以通过 `getFieldsValue(true)` 来获取所有字段数据。

为什么 `setFieldsValue` 设置字段为 `undefined` 时，有的组件不会重置为空?

在 React 中，`value` 从确定值改为 `undefined` 表示从受控变为非受控，因而不会重置展示值（但是 Form 中的值确实已经改变）。你可以通过 HOC 改变这一逻辑：

```
const MyInput = ({
  // 强制保持受控逻辑
  value = '',
  ...rest
}) => <input value={value} {...rest} />;

<Form.Item name="my">
  <MyInput />
</Form.Item>;
```

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。

`scrollToFirstError` 和 `scrollToField` 失效?

1. 使用了自定义表单控件

类似问题：#28370 #27994

从 `5.17.0` 版本开始，滑动操作将优先使用表单控件元素所转发的 ref 元素。因此，在考虑自定义组件支持校验滚动时，请优先考虑将其转发给表单控件元素。

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。

2. 页面内有多个表单

页面内如果有多个表单，且存在表单项 `name` 重复，表单滚动定位可能会查找到另一个表单的同名表单项上。需要给表单 `Form` 组件设置不同的 `name` 以区分。

继上，为何不通过 `ref` 绑定元素?

当自定义组件不支持 `ref` 时，Form 无法获取子元素真实 DOM 节点，而通过包裹 Class Component 调用 `findDOMNode` 会在 React Strict Mode 下触发警告。因而我们使用 id 来进行元素定位。

`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。如果仅仅需要组件内消费，可以通过 `useWatch` 或者 `Field.renderProps` 来实现。

为什么 Form.Item 嵌套子组件后，不更新表单值?

Form.Item 在渲染时会注入 `value` 与 `onChange` 事件给子元素，当你的字段组件被包裹时属性将无法传递。所以以下代码是不会生效的：

```
<Form.Item name="input">
  <div>
    <h3>I am a wrapped Input</h3>
    <Input />
  </div>
</Form.Item>
```

你可以通过 HOC 自定义组件形式来解决这个问题：

```
const MyInput = (props) => (
  <div>
    <h3>I am a wrapped Input</h3>
    <Input {...props} />
  </div>
);

<Form.Item name="input">
  <MyInput />
</Form.Item>;
```

为什么表单点击 label 会更改组件状态？

> *相关 issue：#47031,#43175, #52152*

表单 label 使用 HTML label 元素来包裹表单控件，从而实现点击 label 时聚焦到对应控件。这是 label 元素的原生行为，用于提升可访问性和用户体验，这种标准交互模式能让用户更容易操作表单控件。如果你不希望这种行为，可通过 `htmlFor={null}` 属性解除关联，通常不建议这样做。

```diff
- <Form.Item name="switch" label="Switch">
+ <Form.Item name="switch" label="Switch" htmlFor={null}>
    <Switch />
  </Form.Item>
```

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》。
- 优雅的 Form + Modal 结合使用方案《如何优雅的使用 Form + Modal》。