zh-CN

自定义或第三方的表单控件，也可以与 Form 组件一起使用。只要该组件遵循以下的约定：

> - *提供受控属性 `value` 或其它与 `valuePropName` 的值同名的属性。*
> - *提供 `onChange` 事件或 `trigger` 的值同名的事件。*

en-US

Customized or third-party form controls can be used in Form, too. Controls must follow these conventions:

> - *It has a controlled property `value` or other name which is equal to the value of `valuePropName`.*
> - *It has event `onChange` or an event which name is equal to the value of `trigger`.*

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

const { Option } = Select;

type Currency = 'rmb' | 'dollar';

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

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

const PriceInput: React.FC<PriceInputProps> = ({ value = {}, onChange }) => {
  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>
      <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 Demo = () => {
  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 () => <Demo />;
```