何时使用 {#when-to-use}

当用户需要输入一个日期，可以点击标准输入框，弹出日期面板进行选择。

代码演示

基本

```
import React from 'react';
import type { DatePickerProps } from 'antd';
import { DatePicker, Space } from 'antd';

const onChange: DatePickerProps['onChange'] = (date, dateString) => {
  console.log(date, dateString);
};

const App: React.FC = () => (
  <Space direction="vertical">
    <DatePicker onChange={onChange} />
    <DatePicker onChange={onChange} picker="week" />
    <DatePicker onChange={onChange} picker="month" />
    <DatePicker onChange={onChange} picker="quarter" />
    <DatePicker onChange={onChange} picker="year" />
  </Space>
);

export default App;
```

范围选择器

```
import React from 'react';
import { DatePicker, Space } from 'antd';

const { RangePicker } = DatePicker;

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <RangePicker />
    <RangePicker showTime />
    <RangePicker picker="week" />
    <RangePicker picker="month" />
    <RangePicker picker="quarter" />
    <RangePicker
      picker="year"
      id={{
        start: 'startInput',
        end: 'endInput',
      }}
      onFocus={(_, info) => {
        console.log('Focus:', info.range);
      }}
      onBlur={(_, info) => {
        console.log('Blur:', info.range);
      }}
    />
  </Space>
);

export default App;
```

多选

v5.14.0

```
import React from 'react';
import type { DatePickerProps } from 'antd';
import { DatePicker, Flex } from 'antd';
import dayjs from 'dayjs';
import type { Dayjs } from 'dayjs';

const onChange: DatePickerProps<Dayjs[]>['onChange'] = (date, dateString)
=> {
  console.log(date, dateString);
};

const defaultValue = [dayjs('2000-01-01'), dayjs('2000-01-03'),
dayjs('2000-01-05')];

const App: React.FC = () => (
  <Flex vertical gap="small">
    <DatePicker
      multiple
      onChange={onChange}
      maxTagCount="responsive"
      defaultValue={defaultValue}
      size="small"
    />
    <DatePicker multiple onChange={onChange} maxTagCount="responsive"
defaultValue={defaultValue} />
    <DatePicker
      multiple
      onChange={onChange}
      maxTagCount="responsive"
      defaultValue={defaultValue}
      size="large"
    />
  </Flex>
);

export default App;
```

多选 Debug

Debug

```
import React from 'react';
import { DatePicker, Flex } from 'antd';
import dayjs from 'dayjs';

const defaultValue = Array.from({ length: 10 }).map((_, i) => dayjs('2000-
01-01').add(i, 'day'));

const App: React.FC = () => (
  <Flex vertical gap="small">
    <DatePicker multiple placeholder="Bamboo" />
    <DatePicker multiple defaultValue={defaultValue} size="small" />
    <DatePicker multiple defaultValue={defaultValue} />
    <DatePicker multiple defaultValue={defaultValue} size="large" />
  </Flex>
);

export default App;
```

选择确认

v5.14.0

```
import React from 'react';
import type { DatePickerProps } from 'antd';
import { DatePicker } from 'antd';
import type { Dayjs } from 'dayjs';

const onChange: DatePickerProps<Dayjs[]>['onChange'] = (date, dateString)
=> {
  console.log(date, dateString);
};

const App: React.FC = () => <DatePicker onChange={onChange} needConfirm />;
```

```
export default App;
```

切换不同的选择器

```
import React, { useState } from 'react';
import type { DatePickerProps, TimePickerProps } from 'antd';
import { DatePicker, Select, Space, TimePicker } from 'antd';

const { Option } = Select;

type PickerType = 'time' | 'date';

const PickerWithType = ({
  type,
  onChange,
}: {
  type: PickerType;
  onChange: TimePickerProps['onChange'] | DatePickerProps['onChange'];
}) => {
  if (type === 'time') return <TimePicker onChange={onChange} />;
  if (type === 'date') return <DatePicker onChange={onChange} />;
  return <DatePicker picker={type} onChange={onChange} />;
};

const App: React.FC = () => {
  const [type, setType] = useState<PickerType>('time');

  return (
    <Space>
      <Select aria-label="Picker Type" value={type} onChange={setType}>
        <Option value="time">Time</Option>
        <Option value="date">Date</Option>
        <Option value="week">Week</Option>
        <Option value="month">Month</Option>
        <Option value="quarter">Quarter</Option>
        <Option value="year">Year</Option>
      </Select>
      <PickerWithType type={type} onChange={(value) => console.log(value)}
/>
    </Space>
  );
};

export default App;
```

日期格式

```
import React from 'react';
import type { DatePickerProps } from 'antd';
import { DatePicker, Space } from 'antd';
import dayjs from 'dayjs';
import customParseFormat from 'dayjs/plugin/customParseFormat';

dayjs.extend(customParseFormat);

const { RangePicker } = DatePicker;

const dateFormat = 'YYYY/MM/DD';
const weekFormat = 'MM/DD';
const monthFormat = 'YYYY/MM';

/** Manually entering any of the following formats will perform date
parsing */
const dateFormatList = ['DD/MM/YYYY', 'DD/MM/YY', 'DD-MM-YYYY', 'DD-MM-
YY'];

const customFormat: DatePickerProps['format'] = (value) =>
  `custom format: ${value.format(dateFormat)}`;

const customWeekStartEndFormat: DatePickerProps['format'] = (value) =>
  `${dayjs(value).startOf('week').format(weekFormat)} ~ ${dayjs(value)
    .endOf('week')
    .format(weekFormat)}`;

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <DatePicker defaultValue={dayjs('2015/01/01', dateFormat)} format=
{dateFormat} />
    <DatePicker defaultValue={dayjs('01/01/2015', dateFormatList[0])}
format={dateFormatList} />
    <DatePicker defaultValue={dayjs('2015/01', monthFormat)} format=
{monthFormat} picker="month" />
    <DatePicker defaultValue={dayjs()} format={customWeekStartEndFormat}
picker="week" />
    <RangePicker
      defaultValue={[dayjs('2015/01/01', dateFormat), dayjs('2015/01/01',
dateFormat)]}
      format={dateFormat}
    />
    <DatePicker defaultValue={dayjs('2015/01/01', dateFormat)} format=
{customFormat} />
```

```
  </Space>
);

export default App;
```

日期时间选择

```
import React from 'react';
import { DatePicker, Space } from 'antd';
import type { DatePickerProps, GetProps } from 'antd';

type RangePickerProps = GetProps<typeof DatePicker.RangePicker>;

const { RangePicker } = DatePicker;

const onOk = (value: DatePickerProps['value'] | RangePickerProps['value'])
=> {
  console.log('onOk: ', value);
};

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <DatePicker
      showTime
      onChange={(value, dateString) => {
        console.log('Selected Time: ', value);
        console.log('Formatted Selected Time: ', dateString);
      }}
      onOk={onOk}
    />
    <RangePicker
      showTime={{ format: 'HH:mm' }}
      format="YYYY-MM-DD HH:mm"
      onChange={(value, dateString) => {
        console.log('Selected Time: ', value);
        console.log('Formatted Selected Time: ', dateString);
      }}
      onOk={onOk}
    />
  </Space>
);

export default App;
```

格式对齐

v5.14.0

```
import React from 'react';
import type { DatePickerProps } from 'antd';
import { DatePicker, Space } from 'antd';

const onChange: DatePickerProps['onChange'] = (date, dateString) => {
  console.log(date, dateString);
};

const App: React.FC = () => (
  <Space direction="vertical">
    <DatePicker
      format={{
        format: 'YYYY-MM-DD',
        type: 'mask',
      }}
      onChange={onChange}
    />
    <DatePicker
      format={{
        format: 'YYYY-MM-DD HH:mm:ss',
        type: 'mask',
      }}
      onChange={onChange}
    />
  </Space>
);

export default App;
```

日期限定范围

v5.14.0

```
import React from 'react';
import { DatePicker } from 'antd';
import dayjs from 'dayjs';
import customParseFormat from 'dayjs/plugin/customParseFormat';

dayjs.extend(customParseFormat);

const dateFormat = 'YYYY-MM-DD';

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

```
    defaultValue={dayjs('2019-09-03', dateFormat)}
    minDate={dayjs('2019-08-01', dateFormat)}
    maxDate={dayjs('2020-10-31', dateFormat)}
  />
);

export default App;
```

禁用

```
import React from 'react';
import { DatePicker, Space } from 'antd';
import dayjs from 'dayjs';
import customParseFormat from 'dayjs/plugin/customParseFormat';

dayjs.extend(customParseFormat);

const { RangePicker } = DatePicker;

const dateFormat = 'YYYY-MM-DD';

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <DatePicker defaultValue={dayjs('2015-06-06', dateFormat)} disabled />
    <DatePicker picker="month" defaultValue={dayjs('2015-06', 'YYYY-MM')}
disabled />
    <RangePicker
      defaultValue={[dayjs('2015-06-06', dateFormat), dayjs('2015-06-06',
dateFormat)]}
      disabled
    />
    <RangePicker
      defaultValue={[dayjs('2019-09-03', dateFormat), dayjs('2019-11-22',
dateFormat)]}
      disabled={[false, true]}
    />
    <DatePicker
      defaultValue={dayjs('2019-09-03', dateFormat)}
      minDate={dayjs('2019-06-01', dateFormat)}
      maxDate={dayjs('2020-06-30', dateFormat)}
    />
  </Space>
);

export default App;
```

不可选择日期和时间

```
import React from 'react';
import { DatePicker, Space } from 'antd';
import type { GetProps } from 'antd';
import dayjs from 'dayjs';
import customParseFormat from 'dayjs/plugin/customParseFormat';

type RangePickerProps = GetProps<typeof DatePicker.RangePicker>;

dayjs.extend(customParseFormat);

const { RangePicker } = DatePicker;

const range = (start: number, end: number) => {
  const result = [];
  for (let i = start; i < end; i++) {
    result.push(i);
  }
  return result;
};

const disabledDate: RangePickerProps['disabledDate'] = (current) => {
  // Can not select days before today and today
  return current && current < dayjs().endOf('day');
};

const disabledDateTime = () => ({
  disabledHours: () => range(0, 24).splice(4, 20),
  disabledMinutes: () => range(30, 60),
  disabledSeconds: () => [55, 56],
});

const disabledRangeTime: RangePickerProps['disabledTime'] = (_, type) => {
  if (type === 'start') {
    return {
      disabledHours: () => range(0, 60).splice(4, 20),
      disabledMinutes: () => range(30, 60),
      disabledSeconds: () => [55, 56],
    };
  }
  return {
    disabledHours: () => range(0, 60).splice(20, 4),
    disabledMinutes: () => range(0, 31),
    disabledSeconds: () => [55, 56],
  };
};

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <DatePicker
      format="YYYY-MM-DD HH:mm:ss"
      disabledDate={disabledDate}
      disabledTime={disabledDateTime}
      showTime={{ defaultValue: dayjs('00:00:00', 'HH:mm:ss') }}
    />
    <DatePicker picker="month" disabledDate={disabledDate} />
    <RangePicker disabledDate={disabledDate} />
    <RangePicker
      disabledDate={disabledDate}
      disabledTime={disabledRangeTime}
      showTime={{
        hideDisabledOptions: true,
        defaultValue: [dayjs('00:00:00', 'HH:mm:ss'), dayjs('11:59:59',
'HH:mm:ss')],
      }}
      format="YYYY-MM-DD HH:mm:ss"
    />
  </Space>
);

export default App;
```

允许留空

```
import React from 'react';
import { DatePicker } from 'antd';

const App: React.FC = () => (
  <DatePicker.RangePicker
    placeholder={['Allow Empty', 'Till Now']}
    allowEmpty={[false, true]}
    onChange={(date, dateString) => {
      console.log(date, dateString);
    }}
  />
);

export default App;
```

选择不超过一定的范围

```
import React from 'react';
import { DatePicker, Space, Typography } from 'antd';
import type { DatePickerProps } from 'antd';
import type { Dayjs } from 'dayjs';

const { RangePicker } = DatePicker;

const getYearMonth = (date: Dayjs) => date.year() * 12 + date.month();

// Disabled 7 days from the selected date
const disabled7DaysDate: DatePickerProps['disabledDate'] = (current, {
from, type }) => {
  if (from) {
    const minDate = from.add(-6, 'days');
    const maxDate = from.add(6, 'days');

    switch (type) {
      case 'year':
        return current.year() < minDate.year() || current.year() >
maxDate.year();

      case 'month':
        return (
          getYearMonth(current) < getYearMonth(minDate) ||
          getYearMonth(current) > getYearMonth(maxDate)
        );

      default:
        return Math.abs(current.diff(from, 'days')) >= 7;
    }
  }

  return false;
};

// Disabled 6 months from the selected date
const disabled6MonthsDate: DatePickerProps['disabledDate'] = (current, {
from, type }) => {
  if (from) {
    const minDate = from.add(-5, 'months');
    const maxDate = from.add(5, 'months');

    switch (type) {
      case 'year':
        return current.year() < minDate.year() || current.year() >
maxDate.year();
```

```
      default:
        return (
          getYearMonth(current) < getYearMonth(minDate) ||
          getYearMonth(current) > getYearMonth(maxDate)
        );
    }
  }

  return false;
};

const App: React.FC = () => (
  <Space direction="vertical">
    <Typography.Title level={5}>7 days range</Typography.Title>
    <RangePicker disabledDate={disabled7DaysDate} />

    <Typography.Title level={5}>6 months range</Typography.Title>
    <RangePicker disabledDate={disabled6MonthsDate} picker="month" />
  </Space>
);

export default App;
```

预设范围

```
import React from 'react';
import type { TimeRangePickerProps } from 'antd';
import { DatePicker, Space } from 'antd';
import dayjs from 'dayjs';
import type { Dayjs } from 'dayjs';

const { RangePicker } = DatePicker;

const onChange = (date: Dayjs) => {
  if (date) {
    console.log('Date: ', date);
  } else {
    console.log('Clear');
  }
};
const onRangeChange = (dates: null | (Dayjs | null)[], dateStrings:
string[]) => {
  if (dates) {
    console.log('From: ', dates[0], ', to: ', dates[1]);
    console.log('From: ', dateStrings[0], ', to: ', dateStrings[1]);
  } else {
    console.log('Clear');
  }
};

const rangePresets: TimeRangePickerProps['presets'] = [
  { label: 'Last 7 Days', value: [dayjs().add(-7, 'd'), dayjs()] },
  { label: 'Last 14 Days', value: [dayjs().add(-14, 'd'), dayjs()] },
  { label: 'Last 30 Days', value: [dayjs().add(-30, 'd'), dayjs()] },
  { label: 'Last 90 Days', value: [dayjs().add(-90, 'd'), dayjs()] },
];

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <DatePicker
      presets={[
        { label: 'Yesterday', value: dayjs().add(-1, 'd') },
        { label: 'Last Week', value: dayjs().add(-7, 'd') },
        { label: 'Last Month', value: dayjs().add(-1, 'month') },
      ]}
      onChange={onChange}
    />
    <RangePicker presets={rangePresets} onChange={onRangeChange} />
    <RangePicker
      presets={[
        {
          label: <span aria-label="Current Time to End of Day">Now ~
EOD</span>,
          value: () => [dayjs(), dayjs().endOf('day')], // 5.8.0+ support
function
        },
        ...rangePresets,
      ]}
      showTime
      format="YYYY/MM/DD HH:mm:ss"
      onChange={onRangeChange}
    />
  </Space>
);

export default App;
```

额外的页脚

```
import React from 'react';
import { DatePicker, Space } from 'antd';
```

```
const { RangePicker } = DatePicker;

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <DatePicker renderExtraFooter={() => 'extra footer'} />
    <DatePicker renderExtraFooter={() => 'extra footer'} showTime />
    <RangePicker renderExtraFooter={() => 'extra footer'} />
    <RangePicker renderExtraFooter={() => 'extra footer'} showTime />
    <DatePicker renderExtraFooter={() => 'extra footer'} picker="month" />
  </Space>
);

export default App;
```

三种大小

```
import React, { useState } from 'react';
import type { ConfigProviderProps, RadioChangeEvent } from 'antd';
import { DatePicker, Radio, Space } from 'antd';

type SizeType = ConfigProviderProps['componentSize'];

const { RangePicker } = DatePicker;

const App: React.FC = () => {
  const [size, setSize] = useState<SizeType>('middle');

  const handleSizeChange = (e: RadioChangeEvent) => {
    setSize(e.target.value);
  };

  return (
    <Space direction="vertical" size={12}>
      <Radio.Group value={size} onChange={handleSizeChange}>
        <Radio.Button value="large">Large</Radio.Button>
        <Radio.Button value="middle">middle</Radio.Button>
        <Radio.Button value="small">Small</Radio.Button>
      </Radio.Group>
      <DatePicker size={size} />
      <DatePicker size={size} picker="month" />
      <RangePicker size={size} />
      <DatePicker size={size} picker="week" />
    </Space>
  );
};
```

```
export default App;
```

定制单元格

```
import React from 'react';
import type { DatePickerProps } from 'antd';
import { DatePicker, Space, theme } from 'antd';
import type { Dayjs } from 'dayjs';

const App: React.FC = () => {
  const { token } = theme.useToken();
  const style: React.CSSProperties = {
    border: `1px solid ${token.colorPrimary}`,
    borderRadius: '50%',
  };
  const cellRender: DatePickerProps<Dayjs>['cellRender'] = (current, info)
=> {
    if (info.type !== 'date') {
      return info.originNode;
    }
    if (typeof current === 'number' || typeof current === 'string') {
      return <div className="ant-picker-cell-inner">{current}</div>;
    }
    return (
      <div className="ant-picker-cell-inner" style={current.date() === 1 ?
style : {}}>
        {current.date()}
      </div>
    );
  };
  return (
    <Space size={12} direction="vertical">
      <DatePicker cellRender={cellRender} />
      <DatePicker.RangePicker cellRender={cellRender} />
    </Space>
  );
};

export default App;
```

定制面板

v5.14.0

```
import React from 'react';
import type { DatePickerProps } from 'antd';
import { Button, DatePicker, Flex, Slider, Space, Typography } from 'antd';
import dayjs from 'dayjs';
import type { Dayjs } from 'dayjs';

const onChange: DatePickerProps['onChange'] = (date, dateString) => {
  console.log(date, dateString);
};

type DateComponent = Required<NonNullable<DatePickerProps<Dayjs>
['components']>>['date'];
type GetProps<T> = T extends React.ComponentType<infer P> ? P : never;

const MyDatePanel = (props: GetProps<DateComponent>) => {
  const { value, onSelect, onHover } = props;

  // Value
  const startDate = React.useMemo(() => dayjs().date(1).month(0), []);
  const [innerValue, setInnerValue] = React.useState(value || startDate);

  React.useEffect(() => {
    if (value) {
      setInnerValue(value);
    }
  }, [value]);

  // Range
  const dateCount = React.useMemo(() => {
    const endDate = startDate.add(1, 'year').add(-1, 'day');
    return endDate.diff(startDate, 'day');
  }, [startDate]);

  const sliderValue = Math.min(Math.max(0, innerValue.diff(startDate,
'day')), dateCount);

  // Render
  return (
    <Flex vertical gap="small" style={{ padding: 16 }}>
      <Typography.Title level={4} style={{ margin: 0 }} title="no, it's
not">
        The BEST Picker Panel
      </Typography.Title>
      <Slider
        min={0}
        max={dateCount}
        value={sliderValue}
        onChange={(nextValue) => {
          const nextDate = startDate.add(nextValue, 'day');
          setInnerValue(nextDate);
          onHover?.(nextDate);
        }}
        tooltip={{
          formatter: (nextValue) => startDate.add(nextValue || 0,
'day').format('YYYY-MM-DD'),
        }}
      />
      <Button
        type="primary"
        onClick={() => {
          onSelect(innerValue);
        }}
      >{`That's It!`}</Button>
    </Flex>
  );
};

const App: React.FC = () => (
  <Space direction="vertical">
    <DatePicker
      showNow={false}
      onChange={onChange}
      components={{
        date: MyDatePanel,
      }}
    />
  </Space>
);

export default App;
```

佛历格式

v5.14.0

```
import React from 'react';
import { ConfigProvider, DatePicker, Space, Typography } from 'antd';
import type { DatePickerProps } from 'antd';
import en from 'antd/es/date-picker/locale/en_US';
import enUS from 'antd/es/locale/en_US';
import dayjs from 'dayjs';
import buddhistEra from 'dayjs/plugin/buddhistEra';
```

```tsx
dayjs.extend(buddhistEra);

const { Title } = Typography;

// Component level locale
const buddhistLocale: typeof en = {
  ...en,
  lang: {
    ...en.lang,
    fieldDateFormat: 'BBBB-MM-DD',
    fieldDateTimeFormat: 'BBBB-MM-DD HH:mm:ss',
    yearFormat: 'BBBB',
    cellYearFormat: 'BBBB',
  },
};

// ConfigProvider level locale
const globalBuddhistLocale: typeof enUS = {
  ...enUS,
  DatePicker: {
    ...enUS.DatePicker!,
    lang: buddhistLocale.lang,
  },
};

const defaultValue = dayjs('2024-01-01');

const App: React.FC = () => {
  const onChange: DatePickerProps['onChange'] = (_, dateStr) => {
    console.log('onChange:', dateStr);
  };

  return (
    <Space direction="vertical">
      <Title level={4}>By locale props</Title>
      <DatePicker defaultValue={defaultValue} locale={buddhistLocale}
onChange={onChange} />
      <DatePicker
        defaultValue={defaultValue}
        showTime
        locale={buddhistLocale}
        onChange={onChange}
      />

      <Title level={4}>By ConfigProvider</Title>
      <ConfigProvider locale={globalBuddhistLocale}>
        <Space direction="vertical">
          <DatePicker defaultValue={defaultValue} onChange={onChange} />
          <DatePicker defaultValue={defaultValue} showTime onChange=
{onChange} />
        </Space>
      </ConfigProvider>
    </Space>
  );
};

export default App;
```

自定义状态

```
import React from 'react';
import { DatePicker, Space } from 'antd';

const App: React.FC = () => (
  <Space direction="vertical" style={{ width: '100%' }}>
    <DatePicker status="error" style={{ width: '100%' }} />
    <DatePicker status="warning" style={{ width: '100%' }} />
    <DatePicker.RangePicker status="error" style={{ width: '100%' }} />
    <DatePicker.RangePicker status="warning" style={{ width: '100%' }} />
  </Space>
);

export default App;
```

形态变体

v5.13.0

```
import React from 'react';
import { DatePicker, Flex } from 'antd';

const { RangePicker } = DatePicker;

const App: React.FC = () => (
  <Flex vertical gap={12}>
    <Flex gap={8}>
      <DatePicker placeholder="Outlined" />
      <RangePicker placeholder={['Outlined Start', 'Outlined End']} />
    </Flex>
    <Flex gap={8}>
      <DatePicker placeholder="Filled" variant="filled" />
      <RangePicker placeholder={['Filled Start', 'Filled End']}
variant="filled" />
    </Flex>
    <Flex gap={8}>
      <DatePicker placeholder="Borderless" variant="borderless" />
      <RangePicker placeholder={['Borderless Start', 'Borderless End']}
variant="borderless" />
    </Flex>
    <Flex gap={8}>
      <DatePicker placeholder="Underlined" variant="underlined" />
      <RangePicker placeholder={['Underlined Start', 'Underlined End']}
variant="underlined" />
    </Flex>
  </Flex>
);

export default App;
```

Filled Debug

Debug

```
import React from 'react';
import { DatePicker, Space } from 'antd';
import dayjs from 'dayjs';
import customParseFormat from 'dayjs/plugin/customParseFormat';

dayjs.extend(customParseFormat);

const { RangePicker } = DatePicker;

const dateFormat = 'YYYY-MM-DD';

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <DatePicker variant="filled" defaultValue={dayjs('2015-06-06',
dateFormat)} disabled />
    <DatePicker
      variant="filled"
      picker="month"
      defaultValue={dayjs('2015-06', 'YYYY-MM')}
      disabled
    />
    <RangePicker
      variant="filled"
      defaultValue={[dayjs('2015-06-06', dateFormat), dayjs('2015-06-06',
dateFormat)]}
      disabled
    />
    <RangePicker
      variant="filled"
      defaultValue={[dayjs('2019-09-03', dateFormat), dayjs('2019-11-22',
dateFormat)]}
      disabled={[false, true]}
    />
    <DatePicker
      defaultValue={dayjs('2023-12-25')}
      variant="filled"
      status="error"
      style={{ width: '100%' }}
    />
    <DatePicker variant="filled" status="warning" style={{ width: '100%' }}
/>
    <RangePicker variant="filled" status="error" style={{ width: '100%' }}
/>
    <RangePicker variant="filled" status="warning" style={{ width: '100%'
}} />
  </Space>
);

export default App;
```

弹出位置

```
import React, { useState } from 'react';
import type { DatePickerProps, RadioChangeEvent } from 'antd';
import { DatePicker, Radio } from 'antd';

const { RangePicker } = DatePicker;

const App: React.FC = () => {
  const [placement, SetPlacement] = useState<DatePickerProps['placement']>
('topLeft');

  const placementChange = (e: RadioChangeEvent) => {
    SetPlacement(e.target.value);
  };

  return (
    <>
      <Radio.Group value={placement} onChange={placementChange}>
        <Radio.Button value="topLeft">topLeft</Radio.Button>
        <Radio.Button value="topRight">topRight</Radio.Button>
        <Radio.Button value="bottomLeft">bottomLeft</Radio.Button>
        <Radio.Button value="bottomRight">bottomRight</Radio.Button>
      </Radio.Group>
      <br />
      <br />
      <DatePicker placement={placement} />
      <br />
      <br />
      <RangePicker placement={placement} />
    </>
  );
};

export default App;
```

受控面板

Debug

```
import React, { useState } from 'react';
import { DatePicker, Space } from 'antd';
import type { DatePickerProps, GetProps } from 'antd';
import type { Dayjs } from 'dayjs';

type RangePickerProps = GetProps<typeof DatePicker.RangePicker>;

const { RangePicker } = DatePicker;

type RangeValue = [Dayjs | null | undefined, Dayjs | null | undefined] |
null;

const ControlledDatePicker = () => {
  const [mode, setMode] = useState<DatePickerProps['mode']>('time');

  const handleOpenChange = (open: boolean) => {
    if (open) {
      setMode('time');
    }
  };

  const handlePanelChange: DatePickerProps['onPanelChange'] = (_, newMode)
=> {
    setMode(newMode);
  };
  return (
    <DatePicker
      mode={mode}
      showTime
      onOpenChange={handleOpenChange}
      onPanelChange={handlePanelChange}
    />
  );
};

const ControlledRangePicker = () => {
  const [mode, setMode] = useState<RangePickerProps['mode']>(['month',
'month']);
  const [value, setValue] = useState<RangeValue>([null, null]);

  const handlePanelChange: RangePickerProps['onPanelChange'] = (newValue,
newModes) => {
    setValue(newValue);
    setMode([
      newModes[0] === 'date' ? 'month' : newModes[0],
      newModes[1] === 'date' ? 'month' : newModes[1],
    ]);
  };

  return (
    <RangePicker
      placeholder={['Start month', 'End month']}
      format="YYYY-MM"
      value={value}
      mode={mode}
      onChange={setValue}
      onPanelChange={handlePanelChange}
    />
  );
};

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <ControlledDatePicker />
    <ControlledRangePicker />
  </Space>
);

export default App;
```

自定义日期范围选择

Debug

```
import React, { useState } from 'react';
import { DatePicker, Space } from 'antd';
import type { Dayjs } from 'dayjs';

const App: React.FC = () => {
  const [startValue, setStartValue] = useState<Dayjs | null>(null);
  const [endValue, setEndValue] = useState<Dayjs | null>(null);
  const [endOpen, setEndOpen] = useState(false);

  const disabledStartDate = (startDate: Dayjs) => {
    if (!startDate || !endValue) {
      return false;
    }
    return startDate.valueOf() > endValue.valueOf();
  };

  const disabledEndDate = (endDate: Dayjs) => {
    if (!endDate || !startValue) {
      return false;
    }
    return endDate.valueOf() <= startValue.valueOf();
  };

  const handleStartOpenChange = (open: boolean) => {
    if (!open) {
      setEndOpen(true);
    }
  };

  const handleEndOpenChange = (open: boolean) => {
    setEndOpen(open);
  };

  return (
    <Space>
      <DatePicker
        disabledDate={disabledStartDate}
        showTime
        format="YYYY-MM-DD HH:mm:ss"
        value={startValue}
        placeholder="Start"
        onChange={setStartValue}
        onOpenChange={handleStartOpenChange}
      />
      <DatePicker
        disabledDate={disabledEndDate}
        showTime
        format="YYYY-MM-DD HH:mm:ss"
        value={endValue}
        placeholder="End"
        onChange={setEndValue}
        open={endOpen}
        onOpenChange={handleEndOpenChange}
      />
    </Space>
  );
};

export default App;
```

前后缀

```
import React from 'react';
import { SmileOutlined } from '@ant-design/icons';
import { DatePicker, Space } from 'antd';
import type { Dayjs } from 'dayjs';

const smileIcon = <SmileOutlined />;
const { RangePicker } = DatePicker;

const onChange = (date: Dayjs | (Dayjs | null)[] | null, dateString: string
| string[]) => {
  console.log(date, dateString);
};

const App: React.FC = () => (
  <Space direction="vertical" size={12}>
    <DatePicker suffixIcon={smileIcon} onChange={onChange} />
    <DatePicker suffixIcon={smileIcon} onChange={onChange} picker="month"
/>
    <RangePicker suffixIcon={smileIcon} onChange={onChange} />
    <DatePicker suffixIcon={smileIcon} onChange={onChange} picker="week" />
    <DatePicker suffixIcon="ab" onChange={onChange} />
    <DatePicker suffixIcon="ab" onChange={onChange} picker="month" />
    <RangePicker suffixIcon="ab" onChange={onChange} />
    <DatePicker suffixIcon="ab" onChange={onChange} picker="week" />
    <DatePicker prefix={smileIcon} onChange={onChange} picker="week" />
    <DatePicker prefix="Event Period" onChange={onChange} picker="week" />
    <RangePicker prefix={smileIcon} onChange={onChange} picker="week" />
    <RangePicker prefix="Event Period" onChange={onChange} picker="week" />
  </Space>
);

export default App;
```

_InternalPanelDoNotUseOrYouWillBeFired

Debug

```
import React from 'react';
import { DatePicker } from 'antd';

const { _InternalPanelDoNotUseOrYouWillBeFired: InternalDatePicker } =
DatePicker;

const App: React.FC = () => <InternalDatePicker />;

export default App;
```

组件 Token

Debug

```
import React from 'react';
import { ConfigProvider, DatePicker, Divider, Flex, Space, TimePicker }
from 'antd';
import dayjs from 'dayjs';

/** Test usage. Do not use in your production. */

const { RangePicker } = DatePicker;

const App: React.FC = () => (
  <>
    <ConfigProvider
      theme={{
        components: {
          DatePicker: {
            presetsWidth: 160,
            zIndexPopup: 888,
            cellHoverWithRangeBg: '#f0f0f0',
            cellActiveWithRangeBg: '#e6bbff',
            cellRangeBorderColor: 'green',
            timeColumnWidth: 80,
            timeColumnHeight: 250,
            timeCellHeight: 30,
            cellWidth: 64,
            cellHeight: 40,
            textHeight: 45,
            withoutTimeCellHeight: 70,
          },
        },
      }}
    >
      <Space direction="vertical">
        <DatePicker
          presets={[
            { label: 'Yesterday', value: dayjs().add(-1, 'd') },
            { label: 'Last Week', value: dayjs().add(-7, 'd') },
            { label: 'Last Month', value: dayjs().add(-1, 'month') },
          ]}
        />
        <RangePicker />
        <TimePicker />
        <DatePicker picker="month" />
      </Space>
    </ConfigProvider>

    <Divider />

    <ConfigProvider
      theme={{
        components: {
          DatePicker: {
            controlHeightSM: 32,
            controlHeight: 40,
          },
        },
      }}
    >
      <Flex vertical gap={8}>
        <DatePicker multiple size="small" />
        <DatePicker multiple />
        <DatePicker multiple size="large" />
      </Flex>
    </ConfigProvider>
  </>
);

export default App;
```

API

通用属性参考：通用属性

日期类组件包括以下五种形式。

- DatePicker
- DatePicker[picker="month"]
- DatePicker[picker="week"]
- DatePicker[picker="year"]
- DatePicker[picker="quarter"] (4.1.0 新增)
- RangePicker

国际化配置

默认配置为 en-US，如果你需要设置其他语言，推荐在入口处使用我们提供的国际化组件，详见：ConfigProvider 国际化。

如有特殊需求（仅修改单一组件的语言），请使用 locale 参数，参考：默认配置。

```
// 默认语言为 en-US，如果你需要设置其他语言，推荐在入口文件全局设置 locale
// 确保还导入相关的 dayjs 文件，否则所有文本的区域设置都不会更改（例如范围选择器月份）
import locale from 'antd/locale/zh_CN';
import dayjs from 'dayjs';

import 'dayjs/locale/zh-cn';

dayjs.locale('zh-cn');

<ConfigProvider locale={locale}>
  <DatePicker defaultValue={dayjs('2015-01-01', 'YYYY-MM-DD')} />
</ConfigProvider>;
```

:::warning 在搭配 Next.js 的 App Router 使用时，注意在引入 dayjs 的 locale 文件时加上 `'use client'`。这是由于 Ant Design 的组件都是客户端组件，在 RSC 中引入 dayjs 的 locale 文件将不会在客户端生效。 :::

共同的 API

以下 API 为 DatePicker、 RangePicker 共享的 API。

参数	说明	类型	默认值	版
allowClear	自定义清除按钮	boolean | { clearIcon?: ReactNode }	true	5.8.0: 象类型
autoFocus	自动获取焦点	boolean	false	
className	选择器 className	string	-	

dateRender	自定义日期单元格的内容，5.4.0 起用 `cellRender` 代替	function(currentDate: dayjs, today: dayjs) => React.ReactNode	-	< 5.4.
cellRender	自定义单元格的内容	(current: dayjs, info: { originNode: React.ReactElement,today: DateType, range?: 'start' | 'end', type: PanelMode, locale?: Locale, subType?: 'hour' | 'minute' | 'second' | 'meridiem' }) => React.ReactNode	-	5.4.0
components	自定义面板	Record<Panel | 'input', React.ComponentType>	-	5.14.0
defaultOpen	是否默认展开控制弹层	boolean	-	
disabled	禁用	boolean	false	
disabledDate	不可选择的日期	(currentDate: dayjs, info: { from?: dayjs, type: Picker }) => boolean	-	`info:`
format	设置日期格式，为数组时支持多格式匹配，展示以第一个为准。配置参考 [dayjs#format](#)。示例：[自定义格式](#)	[formatType](#)	[rc-picker](#)	
order	多选、范围时是否自动排序	boolean	true	5.14.0
preserveInvalidOnBlur	失去焦点是否要清空输入框内无效内容	boolean	false	5.14.0
popupClassName	额外的弹出日历 className	string	-	4.23.0
getPopupContainer	定义浮层的容器，默认为	function(trigger)	-	

	body 上新建 div			
inputReadOnly	设置输入框为只读（避免在移动设备上打开虚拟键盘）	boolean	false	
locale	国际化配置	object	[默认配置]()	
minDate	最小日期，同样会限制面板的切换范围	dayjs	-	5.14.0
maxDate	最大日期，同样会限制面板的切换范围	dayjs	-	5.14.0
mode	日期面板的状态（[设置后无法选择年份/月份？]()）	`time` | `date` | `month` | `year` | `decade`	-	
needConfirm	是否需要确认按钮，为 `false` 时失去焦点即代表选择。当设置 `multiple` 时默认为 `false`	boolean	-	5.14.0
nextIcon	自定义下一个图标	ReactNode	-	4.17.0
open	控制弹层是否展开	boolean	-	
panelRender	自定义渲染面板	(panelNode) => ReactNode	-	4.5.0
picker	设置选择器类型	`date` | `week` | `month` | `quarter` | `year`	`date`	`quart` 4.1.0
placeholder	输入框提示文字	string | [string, string]	-	
placement	选择框弹出的位置	`bottomLeft` `bottomRight` `topLeft` `topRight`	bottomLeft	
popupStyle	额外的弹出日历样式	CSSProperties	{}	

prefix	自定义前缀	ReactNode	-	5.22.0
prevIcon	自定义上一个图标	ReactNode	-	4.17.0
presets	预设时间范围快捷选择, 自 `5.8.0` 起 value 支持函数返回值	{ label: React.ReactNode, value: Dayjs | (() => Dayjs) }[]	-	
size	输入框大小，`large` 高度为 40px，`small` 为 24px，默认是 32px	`large` | `middle` | `small`	-	
status	设置校验状态	'error' | 'warning'	-	4.19.0
style	自定义输入框样式	CSSProperties	{}	
suffixIcon	自定义的选择框后缀图标	ReactNode	-	
superNextIcon	自定义 >> 切换图标	ReactNode	-	4.17.0
superPrevIcon	自定义 << 切换图标	ReactNode	-	4.17.0
variant	形态变体	`outlined` | `borderless` | `filled` | `underlined`	`outlined`	5.13.0 `under` 5.24.0
onOpenChange	弹出日历和关闭日历的回调	function(open)	-	
onPanelChange	日历面板切换的回调	function(value, mode)	-	

共同的方法

名称	描述	版本
blur()	移除焦点	
focus()	获取焦点	

DatePicker

参数	说明	类型	默认值	版本

defaultPickerValue	默认面板日期，每次面板打开时会被重置到该日期	dayjs	-	5.14.0
defaultValue	默认日期，如果开始时间或结束时间为 `null` 或者 `undefined`，日期范围将是一个开区间	dayjs	-	
disabledTime	不可选择的时间	function(date)	-	
format	展示的日期格式，配置参考 dayjs#format。	formatType	`YYYY-MM-DD`	
multiple	是否为多选，不支持 `showTime`	boolean	false	5.14.0
pickerValue	面板日期，可以用于受控切换面板所在日期。配合 `onPanelChange` 使用。	dayjs	-	5.14.0
renderExtraFooter	在面板中添加额外的页脚	(mode) => React.ReactNode	-	
showNow	显示当前日期时间的快捷选择	boolean	-	
showTime	增加时间选择功能	Object | boolean	TimePicker Options	
showTime.defaultValue	设置用户选择日期时默认的时分秒，例子	dayjs	dayjs()	
showWeek	DatePicker 下展示当前周	boolean	false	5.14.0
value	日期	dayjs	-	
onChange	时间发生变化的回调	function(date: dayjs, dateString: string)	-	
onOk	点击确定按钮的回调	function()	-	
onPanelChange	日期面板变化时的回调	function(value, mode)	-	

DatePicker[picker=year]

参数	说明	类型	默认值	版本
defaultValue	默认日期	dayjs	-	
format	展示的日期格式，配置参考 dayjs#format。	formatType	YYYY	
multiple	是否为多选	boolean	false	5.14.0
renderExtraFooter	在面板中添加额外的页脚	() => React.ReactNode	-	
value	日期	dayjs	-	
onChange	时间发生变化的回调，发生在用户选择时间时	function(date: dayjs, dateString: string)	-	

DatePicker[picker=quarter]

`4.1.0` 新增。

参数	说明	类型	默认值	版本
defaultValue	默认日期	dayjs	-	
format	展示的日期格式，配置参考 dayjs#format。	formatType	`YYYY-\QQ`	
multiple	是否为多选	boolean	false	5.14.0
renderExtraFooter	在面板中添加额外的页脚	() => React.ReactNode	-	
value	日期	dayjs	-	
onChange	时间发生变化的回调，发生在用户选择时间时	function(date: dayjs, dateString: string)	-	

DatePicker[picker=month]

参数	说明	类型	默认值	版本
defaultValue	默认日期	dayjs	-	
format	展示的日期格式，配置参考 dayjs#format。	formatType	`YYYY-MM`	
multiple	是否为多选	boolean	false	5.14.0
renderExtraFooter	在面板中添加额外的页脚	() => React.ReactNode	-	
value	日期	dayjs	-	

onChange	时间发生变化的回调，发生在用户选择时间时	function(date: dayjs, dateString: string)	-	

DatePicker[picker=week]

参数	说明	类型	默认值	版本
defaultValue	默认日期	[dayjs]()	-	
format	展示的日期格式，配置参考 [dayjs#format]()。	[formatType]()	YYYY-wo	
multiple	是否为多选	boolean	false	5.14.0
renderExtraFooter	在面板中添加额外的页脚	(mode) => React.ReactNode	-	
value	日期	[dayjs]()	-	
onChange	时间发生变化的回调，发生在用户选择时间时	function(date: dayjs, dateString: string)	-	
showWeek	DatePicker 下展示当前周	boolean	true	5.14.0

RangePicker

参数	说明	类型	默认值
allowEmpty	允许起始项部分为空	[boolean, boolean]	[false, false]
cellRender	自定义单元格的内容。	(current: dayjs, info: { originNode: React.ReactElement,today: DateType, range?: 'start' | 'end', type: PanelMode, locale?: Locale, subType?: 'hour' | 'minute' | 'second' | 'meridiem' }) => React.ReactNode	-
dateRender	自定义日期单元格的内容，5.4.0 起用 `cellRender` 代替	function(currentDate: dayjs, today: dayjs) => React.ReactNode	-
defaultPickerValue	默认面板日期，每次面板打开时	[dayjs]()[]	-

	会被重置到该日期		
defaultValue	默认日期	dayjs[]	-
disabled	禁用起始项	[boolean, boolean]	-
disabledTime	不可选择的时间	function(date: dayjs, partial: `start` | end, info: { from?: dayjs })	-
format	展示的日期格式，配置参考 dayjs#format。	formatType	`YYYY-MM-DD HH:mm:ss`
id	设置输入框 id 属性。	{ start?: string, end?: string }	-
pickerValue	面板日期，可以用于受控切换面板所在日期。配合 `onPanelChange` 使用。	dayjs[]	-
presets	预设时间范围快捷选择，自 `5.8.0` 起 value 支持函数返回值	{ label: React.ReactNode, value: (Dayjs | (() => Dayjs))[] }[]	-
renderExtraFooter	在面板中添加额外的页脚	() => React.ReactNode	-
separator	设置分隔符	React.ReactNode	`<SwapRightOutli />`
showTime	增加时间选择功能	Object|boolean	TimePicker Options
showTime.defaultValue	设置用户选择日期时默认的时分秒，例子	dayjs[]	[dayjs(), dayjs()]
value	日期	dayjs[]	-
onCalendarChange	待选日期发生变化的回调。`info` 参数自 4.4.0 添加	function(dates: [dayjs, dayjs], dateStrings: [string, string], info: { range:`start`|end })	-

onChange	日期范围发生变化的回调	function(dates: [dayjs, dayjs], dateStrings: [string, string])	-
onFocus	聚焦时回调	function(event, { range: 'start' | 'end' })	-
onBlur	失焦时回调	function(event, { range: 'start' | 'end' })	-

formatType

```
import type { Dayjs } from 'dayjs';

type Generic = string;
type GenericFn = (value: Dayjs) => string;

export type FormatType =
  | Generic
  | GenericFn
  | Array<Generic | GenericFn>
  | {
      format: string;
      type?: 'mask';
    };
```

注意：`type` 定义为 `5.14.0` 新增。

主题变量（Design Token）

FAQ

当我指定了 DatePicker/RangePicker 的 mode 属性后，点击后无法选择年份/月份?

请参考常见问答

为何日期选择年份后返回的是日期面板而不是月份面板?

当用户选择完年份后，系统会直接切换至日期面板，而非显式提供月份选择。这样做的设计在于用户只需进行一次点击即可完成年份修改，无需再次点击进入月份选择界面，从而减少了用户的操作负担，同时也避免需要额外感知月份的记忆负担。

如何在 DatePicker 中使用自定义日期库（如 Moment.js ）?

请参考《使用自定义日期库》

为什么时间类组件的国际化 locale 设置不生效?

参考 FAQ 为什么时间类组件的国际化 locale 设置不生效？。

如何修改周的起始日?

请使用正确的语言包（#5605），或者修改 dayjs 的 `locale` 配置：https://codesandbox.io/s/dayjs-day-of-week-x9tuj2?file=/demo.tsx

```
import dayjs from 'dayjs';

import 'dayjs/locale/zh-cn';

import updateLocale from 'dayjs/plugin/updateLocale';

dayjs.extend(updateLocale);
dayjs.updateLocale('zh-cn', {
  weekStart: 0,
});
```

为何使用 `panelRender` 时，原来面板无法切换?

当你通过 `panelRender` 动态改变层级结构时，会使得原本的 Panel 被当做新的节点删除并创建。这使得其原本的状态会被重置，保持结构稳定即可。详情请参考 #27263。

如何理解禁用时间日期?

欢迎阅读博客《为什么禁用日期这么难？》了解如何使用。