When To Use

By clicking the input box, you can select a date from a popup calendar.

Examples

Basic

```
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;
```

Range Picker

```
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;
```

Multiple

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;
```

Multiple 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;
```

Need Confirm

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;
```

Switchable picker

```
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;
```

Date Format

```
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;
```

Choose Time

```
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;
```

Mask Format

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;
```

Limit Date Range

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;
```

Disabled

```
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;
```

Disabled Date & Time

```
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;
```

Allow Empty

```
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;
```

Select range dates

```
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;
```

Preset Ranges

```
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;
```

Extra Footer

```
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;
```

Three Sizes

```
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;
```

Customized Cell Rendering

```
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;
```

Customize Panel

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;
```

Buddhist Era

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';
```

```
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;
```

Status

```
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;
```

Variants

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;
```

Placement

```
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;
```

Controlled Panels

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;
```

Customized Range Picker

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;
```

Prefix and Suffix

```
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;
```

Component 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

Common props ref：[Common props]()

There are five kinds of picker:

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

Localization

The default locale is en-US, if you need to use other languages, recommend to use internationalized components provided by us at the entrance. Look at: [ConfigProvider]().

If there are special needs (only modifying single component language), Please use the property: local. Example: [default]().

```
// The default locale is en-US, if you want to use other locale, just set
locale in entry file globally.
// Make sure you import the relevant dayjs file as well, otherwise the
locale won't change for all texts (e.g. range picker months)
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 When use with Next.js App Router, make sure to add `'use client'` before import locale file of dayjs. It's because all components of Ant Design only works in client, importing locale in RSC will not work. :::

Common API

The following APIs are shared by DatePicker, RangePicker.

Property	Description	Type	Default	Vers
allowClear	Customize clear button	boolean | { clearIcon?: ReactNode }	true	5.8.0: Suppor object t

autoFocus	If get focus when component mounted	boolean	false	
className	The picker className	string	-	
dateRender	Custom rendering function for date cells, >= 5.4.0 use `cellRender` instead.	function(currentDate: dayjs, today: dayjs) => React.ReactNode	-	< 5.4.0
cellRender	Custom rendering function for picker cells	(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	Custom panels	Record<Panel | 'input', React.ComponentType>	-	5.14.0
defaultOpen	Initial open state of picker	boolean	-	
disabled	Determine whether the DatePicker is disabled	boolean	false	
disabledDate	Specify the date that cannot be selected	(currentDate: dayjs, info: { from?: dayjs, type: Picker }) => boolean	-	`info`: 5
format	To set the date format, support multi-format matching when it is an	[formatType]()	[rc-picker]()	

	array, display the first one shall prevail. refer to dayjs#format. for example: Custom Format			
order	Auto order date when multiple or range selection	boolean	true	5.14.0
popupClassName	To customize the className of the popup calendar	string	-	4.23.0
preserveInvalidOnBlur	Not clean input on blur even when the typing is invalidate	boolean	false	5.14.0
getPopupContainer	To set the container of the floating layer, while the default is to create a `div` element in body	function(trigger)	-	
inputReadOnly	Set the `readonly` attribute of the input tag (avoids virtual keyboard on touch devices)	boolean	false	
locale	Localization configuration	object	default	

minDate	The minimum date, which also limits the range of panel switching	dayjs	-	5.14.0
maxDate	The maximum date, which also limits the range of panel switching	dayjs	-	5.14.0
mode	The picker panel mode（ Cannot select year or month anymore?)	`time` | `date` | `month` | `year` | `decade`	-	
needConfirm	Need click confirm button to trigger value change. Default `false` when `multiple`	boolean	-	5.14.0
nextIcon	The custom next icon	ReactNode	-	4.17.0
open	The open state of picker	boolean	-	
panelRender	Customize panel render	(panelNode) => ReactNode	-	4.5.0
picker	Set picker type	`date` | `week` | `month` | `quarter` | `year`	`date`	`quarte` 4.1.0
placeholder	The placeholder of date input	string | [string,string]	-	

placement	The position where the selection box pops up	`bottomLeft` `bottomRight` `topLeft` `topRight`	bottomLeft	
popupStyle	To customize the style of the popup calendar	CSSProperties	{}	
prefix	The custom prefix	ReactNode	-	5.22.0
presets	The preset ranges for quick selection, Since `5.8.0`, preset value supports callback function.	{ label: React.ReactNode, value: Dayjs | (() => Dayjs) }[]	-	
prevIcon	The custom prev icon	ReactNode	-	4.17.0
size	To determine the size of the input box, the height of `large` and `small`, are 40px and 24px respectively, while default size is 32px	`large` | `middle` | `small`	-	
status	Set validation status	'error' | 'warning'	-	4.19.0
style	To customize the style of the input box	CSSProperties	{}	
suffixIcon	The custom suffix icon	ReactNode	-	

superNextIcon	The custom super next icon	ReactNode	-	4.17.0
superPrevIcon	The custom super prev icon	ReactNode	-	4.17.0
variant	Variants of picker	`outlined` | `borderless` | `filled` | `underlined`	`outlined`	5.13.0 | `underl` 5.24.0
onOpenChange	Callback function, can be executed whether the popup calendar is popped up or closed	function(open)	-	
onPanelChange	Callback when picker panel mode is changed	function(value, mode)	-	

Common Methods

Name	Description	Version
blur()	Remove focus	
focus()	Get focus	

DatePicker

Property	Description	Type	Default	Version
defaultPickerValue	Default panel date, will be reset when panel open	dayjs	-	5.14.0
defaultValue	To set default date, if start time or end time is null or undefined, the date range will be an open interval	dayjs	-	

disabledTime	To specify the time that cannot be selected	function(date)	-	
format	To set the date format. refer to dayjs#format	formatType	`YYYY-MM-DD`	
multiple	Enable multiple selection. Not support `showTime`	boolean	false	5.14.0
pickerValue	Panel date. Used for controlled switching of panel date. Work with `onPanelChange`	dayjs	-	5.14.0
renderExtraFooter	Render extra footer in panel	(mode) => React.ReactNode	-	
showNow	Show the fast access of current datetime	boolean	-	4.4.0
showTime	To provide an additional time selection	object | boolean	TimePicker Options	
showTime.defaultValue	To set default time of selected date, demo	dayjs	dayjs()	
showWeek	Show week info when in DatePicker	boolean	false	5.14.0
value	To set date	dayjs	-	
onChange	Callback function, can be executed when the selected time is changing	function(date: dayjs, dateString: string)	-	
onOk	Callback when click ok button	function()	-	
onPanelChange	Callback function for panel changing	function(value, mode)	-	

DatePicker[picker=year]

Property	Description	Type	Default	Version
defaultValue	To set default date	dayjs	-	
format	To set the date format. refer to dayjs#format	formatType	YYYY	
multiple	Enable multiple selection	boolean	false	5.14.0
renderExtraFooter	Render extra footer in panel	() => React.ReactNode	-	
value	To set date	dayjs	-	
onChange	Callback function, can be executed when the selected time is changing	function(date: dayjs, dateString: string)	-	

DatePicker[picker=quarter]

Added in `4.1.0` .

Property	Description	Type	Default	Version
defaultValue	To set default date	dayjs	-	
format	To set the date format. refer to dayjs#format	formatType	`YYYY-\QQ`	
multiple	Enable multiple selection	boolean	false	5.14.0
renderExtraFooter	Render extra footer in panel	() => React.ReactNode	-	
value	To set date	dayjs	-	
onChange	Callback function, can be executed when the selected time is changing	function(date: dayjs, dateString: string)	-	

DatePicker[picker=month]

Property	Description	Type	Default	Version
defaultValue	To set default date	dayjs	-	

format	To set the date format. refer to [dayjs#format]()	[formatType]()	YYYY-MM	
multiple	Enable multiple selection	boolean	false	5.14.0
renderExtraFooter	Render extra footer in panel	() => React.ReactNode	-	
value	To set date	[dayjs]()	-	
onChange	Callback function, can be executed when the selected time is changing	function(date: dayjs, dateString: string)	-	

DatePicker[picker=week]

Property	Description	Type	Default	Version
defaultValue	To set default date	[dayjs]()	-	
format	To set the date format. refer to [dayjs#format]()	[formatType]()	YYYY-wo	
multiple	Enable multiple selection	boolean	false	5.14.0
renderExtraFooter	Render extra footer in panel	(mode) => React.ReactNode	-	
value	To set date	[dayjs]()	-	
onChange	Callback function, can be executed when the selected time is changing	function(date: dayjs, dateString: string)	-	
showWeek	Show week info when in DatePicker	boolean	true	5.14.0

RangePicker

Property	Description	Type	Default
allowEmpty	Allow start or end input leave empty	[boolean, boolean]	[false, false]
cellRender	Custom rendering	(current: dayjs, info: { originNode:	-

	function for picker cells	React.ReactElement,today: DateType, range?: 'start' | 'end', type: PanelMode, locale?: Locale, subType?: 'hour' | 'minute' | 'second' | 'meridiem' }) => React.ReactNode	
dateRender	Custom rendering function for date cells, >= 5.4.0 use `cellRender` instead.	function(currentDate: dayjs, today: dayjs) => React.ReactNode	-
defaultPickerValue	Default panel date, will be reset when panel open	dayjs	-
defaultValue	To set default date	[dayjs, dayjs]	-
disabled	If disable start or end	[boolean, boolean]	-
disabledTime	To specify the time that cannot be selected	function(date: dayjs, partial: `start` | end, info: { from?: dayjs })	-
format	To set the date format. refer to dayjs#format	formatType	`YYYY-MM-DD HH:mm:ss`
id	Config input ids	{ start?: string, end?: string }	-
pickerValue	Panel date. Used for controlled switching of panel date. Work with `onPanelChange`	dayjs	-
presets	The preset ranges for quick selection, Since	{ label: React.ReactNode, value: (Dayjs | (() => Dayjs))[] }[]	-

	`5.8.0`, preset value supports callback function.		
renderExtraFooter	Render extra footer in panel	() => React.ReactNode	-
separator	Set separator between inputs	React.ReactNode	`<SwapRightOutli />`
showTime	To provide an additional time selection	object | boolean	[TimePicker Options
showTime.defaultValue	To set default time of selected date, [demo]	[dayjs][]	[dayjs(), dayjs()]
value	To set date	[[dayjs], [dayjs]]	-
onCalendarChange	Callback function, can be executed when the start time or the end time of the range is changing. `info` argument is added in 4.4.0	function(dates: [dayjs, dayjs], dateStrings: [string, string], info: { range:`start`|end })	-
onChange	Callback function, can be executed when the selected time is changing	function(dates: [dayjs, dayjs], dateStrings: [string, string])	-
onFocus	Trigger when get focus	function(event, { range: 'start' | 'end' })	-
onBlur	Trigger when lose focus	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';
    };
```

Note: `type` is added in `5.14.0`.

Design Token

FAQ

When set mode to DatePicker/RangePicker, cannot select year or month anymore?

Please refer [FAQ]

Why does the date picker switch to the date panel after selecting the year instead of the month panel?

After selecting the year, the system directly switches to the date panel instead of month panel. This design is intended to reduce the user's operational burden by allowing them to complete the year modification with just one click, without having to enter the month selection interface again. At the same time, it also avoids additional cognitive burden of remembering the month.

How to use DatePicker with customize date library like dayjs?

Please refer [Use custom date library]

Why config dayjs.locale globally not work?

DatePicker default set `locale` as `en` in v4. You can config DatePicker `locale` prop or [ConfigProvider `locale`] prop instead.

Date-related components locale is not working?

See FAQ [Date-related-components-locale-is-not-working?]

How to modify start day of week?

Please use correct [language] ([#5605]), or update dayjs `locale` config:

- Example: 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,
});
```

Why origin panel don't switch when using `panelRender` ?

When you change the layout of nodes by `panelRender` , React will unmount and re-mount it which reset the component state. You should keep the layout stable. Please ref #27263 for more info.

How to understand disabled time and date?

Please refer to the blog 'Why is it so hard to disable the date?', to learn how to use it.