When To Use

When data is in the form of dates, such as schedules, timetables, prices calendar, lunar calendar. This component also supports Year/Month switch.

Examples

Basic

```
import React from 'react';
import { Calendar } from 'antd';
import type { CalendarProps } from 'antd';
import type { Dayjs } from 'dayjs';

const App: React.FC = () => {
  const onPanelChange = (value: Dayjs, mode: CalendarProps<Dayjs>['mode'])
=> {
    console.log(value.format('YYYY-MM-DD'), mode);
  };

  return <Calendar onPanelChange={onPanelChange} />;
};

export default App;
```

Notice Calendar

```
import React from 'react';
import type { BadgeProps, CalendarProps } from 'antd';
import { Badge, Calendar } from 'antd';
import type { Dayjs } from 'dayjs';

const getListData = (value: Dayjs) => {
  let listData: { type: string; content: string }[] = []; // Specify the
type of listData
  switch (value.date()) {
    case 8:
      listData = [
        { type: 'warning', content: 'This is warning event.' },
        { type: 'success', content: 'This is usual event.' },
      ];
      break;
    case 10:
      listData = [
        { type: 'warning', content: 'This is warning event.' },
        { type: 'success', content: 'This is usual event.' },
```

```
        { type: 'error', content: 'This is error event.' },
      ];
      break;
    case 15:
      listData = [
        { type: 'warning', content: 'This is warning event' },
        { type: 'success', content: 'This is very long usual event......'
},
        { type: 'error', content: 'This is error event 1.' },
        { type: 'error', content: 'This is error event 2.' },
        { type: 'error', content: 'This is error event 3.' },
        { type: 'error', content: 'This is error event 4.' },
      ];
      break;
    default:
  }
  return listData || [];
};

const getMonthData = (value: Dayjs) => {
  if (value.month() === 8) {
    return 1394;
  }
};

const App: React.FC = () => {
  const monthCellRender = (value: Dayjs) => {
    const num = getMonthData(value);
    return num ? (
      <div className="notes-month">
        <section>{num}</section>
        <span>Backlog number</span>
      </div>
    ) : null;
  };

  const dateCellRender = (value: Dayjs) => {
    const listData = getListData(value);
    return (
      <ul className="events">
        {listData.map((item) => (
          <li key={item.content}>
            <Badge status={item.type as BadgeProps['status']} text=
{item.content} />
          </li>
        ))}
```

```
      </ul>
    );
  };

  const cellRender: CalendarProps<Dayjs>['cellRender'] = (current, info) =>
{
    if (info.type === 'date') return dateCellRender(current);
    if (info.type === 'month') return monthCellRender(current);
    return info.originNode;
  };

  return <Calendar cellRender={cellRender} />;
};

export default App;
```

Card

```
import React from 'react';
import { Calendar, theme } from 'antd';
import type { CalendarProps } from 'antd';
import type { Dayjs } from 'dayjs';

const onPanelChange = (value: Dayjs, mode: CalendarProps<Dayjs>['mode']) =>
{
  console.log(value.format('YYYY-MM-DD'), mode);
};

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

  const wrapperStyle: React.CSSProperties = {
    width: 300,
    border: `1px solid ${token.colorBorderSecondary}`,
    borderRadius: token.borderRadiusLG,
  };

  return (
    <div style={wrapperStyle}>
      <Calendar fullscreen={false} onPanelChange={onPanelChange} />
    </div>
  );
};

export default App;
```

Selectable Calendar

```
import React, { useState } from 'react';
import { Alert, Calendar } from 'antd';
import type { Dayjs } from 'dayjs';
import dayjs from 'dayjs';

const App: React.FC = () => {
  const [value, setValue] = useState(() => dayjs('2017-01-25'));
  const [selectedValue, setSelectedValue] = useState(() => dayjs('2017-01-
25'));

  const onSelect = (newValue: Dayjs) => {
    setValue(newValue);
    setSelectedValue(newValue);
  };

  const onPanelChange = (newValue: Dayjs) => {
    setValue(newValue);
  };

  return (
    <>
      <Alert message={`You selected date: ${selectedValue?.format('YYYY-MM-
DD')}`} />
      <Calendar value={value} onSelect={onSelect} onPanelChange=
{onPanelChange} />
    </>
  );
};

export default App;
```

Lunar Calendar

```
import React from 'react';
import { Calendar, Col, Radio, Row, Select } from 'antd';
import type { CalendarProps } from 'antd';
import { createStyles } from 'antd-style';
import classNames from 'classnames';
import dayjs from 'dayjs';
import type { Dayjs } from 'dayjs';
import { HolidayUtil, Lunar } from 'lunar-typescript';

const useStyle = createStyles(({ token, css, cx }) => {
```

```
const lunar = css`
  color: ${token.colorTextTertiary};
  font-size: ${token.fontSizeSM}px;
`;
const weekend = css`
  color: ${token.colorError};
  &.gray {
    opacity: 0.4;
  }
`;
return {
  wrapper: css`
    width: 450px;
    border: 1px solid ${token.colorBorderSecondary};
    border-radius: ${token.borderRadiusOuter};
    padding: 5px;
  `,
  dateCell: css`
    position: relative;
    &:before {
      content: '';
      position: absolute;
      inset-inline-start: 0;
      inset-inline-end: 0;
      top: 0;
      bottom: 0;
      margin: auto;
      max-width: 40px;
      max-height: 40px;
      background: transparent;
      transition: background-color 300ms;
      border-radius: ${token.borderRadiusOuter}px;
      border: 1px solid transparent;
      box-sizing: border-box;
    }
    &:hover:before {
      background: rgba(0, 0, 0, 0.04);
    }
  `,
  today: css`
    &:before {
      border: 1px solid ${token.colorPrimary};
    }
  `,
  text: css`
    position: relative;
```

```
      z-index: 1;
    `,
    lunar,
    current: css`
      color: ${token.colorTextLightSolid};
      &:before {
        background: ${token.colorPrimary};
      }
      &:hover:before {
        background: ${token.colorPrimary};
        opacity: 0.8;
      }
      .${cx(lunar)} {
        color: ${token.colorTextLightSolid};
        opacity: 0.9;
      }
      .${cx(weekend)} {
        color: ${token.colorTextLightSolid};
      }
    `,
    monthCell: css`
      width: 120px;
      color: ${token.colorTextBase};
      border-radius: ${token.borderRadiusOuter}px;
      padding: 5px 0;
      &:hover {
        background: rgba(0, 0, 0, 0.04);
      }
    `,
    monthCellCurrent: css`
      color: ${token.colorTextLightSolid};
      background: ${token.colorPrimary};
      &:hover {
        background: ${token.colorPrimary};
        opacity: 0.8;
      }
    `,
    weekend,
  };
});

const App: React.FC = () => {
  const { styles } = useStyle({ test: true });

  const [selectDate, setSelectDate] = React.useState<Dayjs>(dayjs());
  const [panelDateDate, setPanelDate] = React.useState<Dayjs>(dayjs());
```

```
  const onPanelChange = (value: Dayjs, mode: CalendarProps<Dayjs>['mode'])
=> {
    console.log(value.format('YYYY-MM-DD'), mode);
    setPanelDate(value);
  };

  const onDateChange: CalendarProps<Dayjs>['onSelect'] = (value,
selectInfo) => {
    if (selectInfo.source === 'date') {
      setSelectDate(value);
    }
  };

  const cellRender: CalendarProps<Dayjs>['fullCellRender'] = (date, info)
=> {
    const d = Lunar.fromDate(date.toDate());
    const lunar = d.getDayInChinese();
    const solarTerm = d.getJieQi();
    const isWeekend = date.day() === 6 || date.day() === 0;
    const h = HolidayUtil.getHoliday(date.get('year'), date.get('month') +
1, date.get('date'));
    const displayHoliday = h?.getTarget() === h?.getDay() ? h?.getName() :
undefined;
    if (info.type === 'date') {
      return React.cloneElement(info.originNode, {
        ...(info.originNode as React.ReactElement<any>).props,
        className: classNames(styles.dateCell, {
          [styles.current]: selectDate.isSame(date, 'date'),
          [styles.today]: date.isSame(dayjs(), 'date'),
        }),
        children: (
          <div className={styles.text}>
            <span
              className={classNames({
                [styles.weekend]: isWeekend,
                gray: !panelDateDate.isSame(date, 'month'),
              })}
            >
              {date.get('date')}
            </span>
            {info.type === 'date' && (
              <div className={styles.lunar}>{displayHoliday || solarTerm ||
lunar}</div>
            )}
          </div>
```

```
        ),
      });
    }

    if (info.type === 'month') {
      // Due to the fact that a solar month is part of the lunar month X
and part of the lunar month X+1,
      // when rendering a month, always take X as the lunar month of the
month
      const d2 = Lunar.fromDate(new Date(date.get('year'),
date.get('month')));
      const month = d2.getMonthInChinese();
      return (
        <div
          className={classNames(styles.monthCell, {
            [styles.monthCellCurrent]: selectDate.isSame(date, 'month'),
          })}
        >
          {date.get('month') + 1}月 ({month}月)
        </div>
      );
    }
  };

  const getYearLabel = (year: number) => {
    const d = Lunar.fromDate(new Date(year + 1, 0));
    return `${d.getYearInChinese()}年
($${d.getYearInGanZhi()}${d.getYearShengXiao()}年) `;
  };

  const getMonthLabel = (month: number, value: Dayjs) => {
    const d = Lunar.fromDate(new Date(value.year(), month));
    const lunar = d.getMonthInChinese();
    return `${month + 1}月 (${lunar}月) `;
  };

  return (
    <div className={styles.wrapper}>
      <Calendar
        fullCellRender={cellRender}
        fullscreen={false}
        onPanelChange={onPanelChange}
        onSelect={onDateChange}
        headerRender={({ value, type, onChange, onTypeChange }) => {
          const start = 0;
          const end = 12;
```

```
const monthOptions = [];

let current = value.clone();
const localeData = value.localeData();
const months = [];
for (let i = 0; i < 12; i++) {
  current = current.month(i);
  months.push(localeData.monthsShort(current));
}

for (let i = start; i < end; i++) {
  monthOptions.push({
    label: getMonthLabel(i, value),
    value: i,
  });
}

const year = value.year();
const month = value.month();
const options = [];
for (let i = year - 10; i < year + 10; i += 1) {
  options.push({
    label: getYearLabel(i),
    value: i,
  });
}
return (
  <Row justify="end" gutter={8} style={{ padding: 8 }}>
    <Col>
      <Select
        size="small"
        popupMatchSelectWidth={false}
        className="my-year-select"
        value={year}
        options={options}
        onChange={(newYear) => {
          const now = value.clone().year(newYear);
          onChange(now);
        }}
      />
    </Col>
    <Col>
      <Select
        size="small"
        popupMatchSelectWidth={false}
        value={month}
```

```
                options={monthOptions}
                onChange={(newMonth) => {
                  const now = value.clone().month(newMonth);
                  onChange(now);
                }}
              />
            </Col>
            <Col>
              <Radio.Group
                size="small"
                onChange={(e) => onTypeChange(e.target.value)}
                value={type}
              >
                <Radio.Button value="month">月</Radio.Button>
                <Radio.Button value="year">年</Radio.Button>
              </Radio.Group>
            </Col>
          </Row>
        );
      }}
    />
  </div>
);
};

export default App;
```

Show Week

v5.23.0

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

const App: React.FC = () => (
  <>
    <Calendar fullscreen showWeek />
    <br />
    <Calendar fullscreen={false} showWeek />
  </>
);

export default App;
```

Customize Header

```tsx
import React from 'react';
import dayjs from 'dayjs';

import 'dayjs/locale/zh-cn';

import { Calendar, Col, Radio, Row, Select, theme, Typography } from
'antd';
import type { CalendarProps } from 'antd';
import type { Dayjs } from 'dayjs';
import dayLocaleData from 'dayjs/plugin/localeData';

dayjs.extend(dayLocaleData);

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

  const onPanelChange = (value: Dayjs, mode: CalendarProps<Dayjs>['mode'])
=> {
    console.log(value.format('YYYY-MM-DD'), mode);
  };

  const wrapperStyle: React.CSSProperties = {
    width: 300,
    border: `1px solid ${token.colorBorderSecondary}`,
    borderRadius: token.borderRadiusLG,
  };

  return (
    <div style={wrapperStyle}>
      <Calendar
        fullscreen={false}
        headerRender={({ value, type, onChange, onTypeChange }) => {
          const start = 0;
          const end = 12;
          const monthOptions = [];

          let current = value.clone();
          const localeData = value.localeData();
          const months = [];
          for (let i = 0; i < 12; i++) {
            current = current.month(i);
            months.push(localeData.monthsShort(current));
          }

          for (let i = start; i < end; i++) {
            monthOptions.push(
```

```
    <Select.Option key={i} value={i} className="month-item">
      {months[i]}
    </Select.Option>,
  );
}

const year = value.year();
const month = value.month();
const options = [];
for (let i = year - 10; i < year + 10; i += 1) {
  options.push(
    <Select.Option key={i} value={i} className="year-item">
      {i}
    </Select.Option>,
  );
}
return (
  <div style={{ padding: 8 }}>
    <Typography.Title level={4}>Custom header</Typography.Title>
    <Row gutter={8}>
      <Col>
        <Radio.Group
          size="small"
          onChange={(e) => onTypeChange(e.target.value)}
          value={type}
        >
          <Radio.Button value="month">Month</Radio.Button>
          <Radio.Button value="year">Year</Radio.Button>
        </Radio.Group>
      </Col>
      <Col>
        <Select
          size="small"
          popupMatchSelectWidth={false}
          className="my-year-select"
          value={year}
          onChange={(newYear) => {
            const now = value.clone().year(newYear);
            onChange(now);
          }}
        >
          {options}
        </Select>
      </Col>
      <Col>
        <Select
```

```
                    size="small"
                    popupMatchSelectWidth={false}
                    value={month}
                    onChange={(newMonth) => {
                      const now = value.clone().month(newMonth);
                      onChange(now);
                    }}
                  >
                    {monthOptions}
                  </Select>
                </Col>
              </Row>
            </div>
          );
        }}
        onPanelChange={onPanelChange}
      />
    </div>
  );
};

export default App;
```

Component Token

Debug

```
import React from 'react';
import { Calendar, ConfigProvider } from 'antd';
import type { CalendarProps } from 'antd';
import type { Dayjs } from 'dayjs';

/** Test usage. Do not use in your production. */
export default () => {
  const onPanelChange = (value: Dayjs, mode: CalendarProps<Dayjs>['mode'])
=> {
    console.log(value.format('YYYY-MM-DD'), mode);
  };

  return (
    <ConfigProvider
      theme={{
        components: {
          Calendar: {
            fullBg: 'red',
            fullPanelBg: 'green',
```

```
            itemActiveBg: 'black',
          },
        },
      }}
    >
      <Calendar onPanelChange={onPanelChange} />
      <br />
      <Calendar onPanelChange={onPanelChange} fullscreen={false} />
    </ConfigProvider>
  );
};
```

API

Common props ref：Common props

Note: Part of the Calendar's locale is read from `value`. So, please set the locale of `dayjs` correctly.

```
// The default locale is en-US, if you want to use other locale, just set locale in entry file globally.
// import dayjs from 'dayjs';
// import 'dayjs/locale/zh-cn';
// dayjs.locale('zh-cn');

<Calendar cellRender={cellRender} onPanelChange={onPanelChange} onSelect={onSelect} />
```

Property	Description	Type	Default	Version
cellRender	Customize cell content	function(current: dayjs, info: { prefixCls: string, originNode: React.ReactElement, today: dayjs, range?: 'start' | 'end', type: PanelMode, locale?: Locale, subType?: 'hour' | 'minute' | 'second' | 'meridiem' }) => React.ReactNode	-	5.4.0
dateFullCellRender	Customize the display of the date cell, the	function(date: Dayjs): ReactNode	-	

	returned content will override the cell			
fullCellRender	Customize cell content	function(current: dayjs, info: { prefixCls: string, originNode: React.ReactElement, today: dayjs, range?: 'start' | 'end', type: PanelMode, locale?: Locale, subType?: 'hour' | 'minute' | 'second' | 'meridiem' }) => React.ReactNode	-	5.4.0
defaultValue	The date selected by default	dayjs	-	
disabledDate	Function that specifies the dates that cannot be selected, `currentDate` is same dayjs object as `value` prop which you shouldn't mutate it] (https://github.com/ant-design/ant-design/issues/30987)	(currentDate: Dayjs) => boolean	-	
fullscreen	Whether to display in full-screen	boolean	true	
showWeek	Whether to display week number	boolean	false	5.23.0
headerRender	Render custom header in panel	function(object: {value: Dayjs, type: 'year' | 'month', onChange: f(), onTypeChange: f()})	-	
locale	The calendar's locale	object	(default)	
mode	The display mode of the calendar	`month` | `year`	`month`	
validRange	To set valid range	[dayjs, dayjs]	-	

value	The current selected date	dayjs	-	
onChange	Callback for when date changes	function(date: Dayjs)	-	
onPanelChange	Callback for when panel changes	function(date: Dayjs, mode: string)	-	
onSelect	Callback for when a date is selected, include source info	function(date: Dayjs, info: { source: 'year' | 'month' | 'date' | 'customize' })	-	`info`: 5.6.0

Design Token

FAQ

How to use Calendar with customize date library?

See Use custom date library

How to set locale for date-related components?

See How to set locale for date-related components

Date-related components locale is not working?

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

How to get date from panel click?

`onSelect` provide `info.source` to help on this:

```
<Calendar
  onSelect={(date, { source }) => {
    if (source === 'date') {
      console.log('Panel Select:', source);
    }
  }}
/>
```