何时使用 {#when-to-use}

当数据是日期或按照日期划分时，例如日程、课表、价格日历等，农历等。目前支持年/月切换。

代码演示

基本

```
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;
```

通知事项日历

```
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;
```

卡片模式

```
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;
```

选择功能

```
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;
```

农历日历

```
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;
```

周数

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;
```

自定义头部

```
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;
```

组件 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

通用属性参考：通用属性

注意：Calendar 部分 locale 是从 value 中读取，所以请先正确设置 dayjs 的 locale。

```
// 默认语言为 en-US，所以如果需要使用其他语言，推荐在入口文件全局设置 locale
// import dayjs from 'dayjs';
// import 'dayjs/locale/zh-cn';
// dayjs.locale('zh-cn');

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

参数	说明	类型	默认值	版本
cellRender	自定义单元格的内容	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	自定义渲染日期单元格，返回内容覆盖单元格，>= 5.4.0 请用 `fullCellRender`	function(date: Dayjs): ReactNode	-	< 5.4.0
fullCellRender	自定义单元格的内容	function(current: dayjs, info: { prefixCls: string, originNode:	-	5.4.0

		React.ReactElement, today: dayjs, range?: 'start' | 'end', type: PanelMode, locale?: Locale, subType?: 'hour' | 'minute' | 'second' | 'meridiem' }) => React.ReactNode		
defaultValue	默认展示的日期	[dayjs]()	-	
disabledDate	不可选择的日期，参数为当前 `value`，注意使用时[不要直接修改]()	(currentDate: Dayjs) => boolean	-	
fullscreen	是否全屏显示	boolean	true	
showWeek	是否显示周数列	boolean	false	5.23.0
headerRender	自定义头部内容	function(object:{value: Dayjs, type: 'year' | 'month', onChange: f(), onTypeChange: f()})	-	
locale	国际化配置	object	[(默认配置)]()	
mode	初始模式	`month` | `year`	`month`	
validRange	设置可以显示的日期	[[dayjs](), [dayjs]()]	-	
value	展示日期	[dayjs]()	-	
onChange	日期变化回调	function(date: Dayjs)	-	
onPanelChange	日期面板变化回调	function(date: Dayjs, mode: string)	-	
onSelect	选择日期回调，包含来源信息	function(date: Dayjs, info: { source: 'year' | 'month' | 'date' | 'customize' })	-	`info`: 5.6.0

主题变量（Design Token）

FAQ

如何在 Calendar 中使用自定义日期库

参考 [使用自定义日期库]()。

如何给日期类组件配置国际化?

参考 如何给日期类组件配置国际化。

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? 。

如何仅获取来自面板点击的日期?

`onSelect` 事件提供额外的来源信息，你可以通过 `info.source` 来判断来源：

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