



# Chronoverse

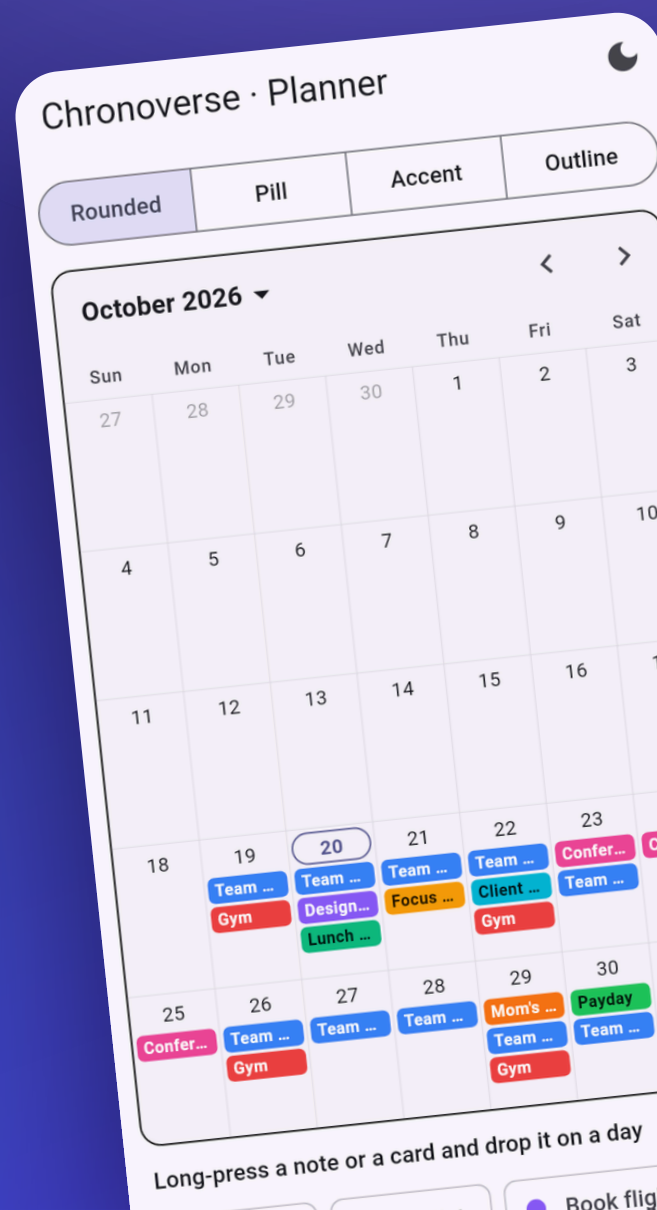
## Integration Guide

How to use every calendar, picker, view and function in Chronoverse, with complete code examples, theming, languages, events, time zones and best practices.

Version 1.0.0 · October 2026

Developed by Sanket Santosh Vidya Manohar

BSD 3-Clause License · Zero dependencies



# Contents

1. Concepts and imports
2. The month calendar
3. Controlling the calendar from code
4. Selection, availability, freeze and disable
5. Events on the calendar
6. Planner cards and drag & drop
7. More views: vertical, year, strip, schedule, agenda, heatmap
8. Pickers and typed input
9. Theming
10. Languages and right-to-left
11. The Chrono toolbox
12. Events, recurrence and iCalendar
13. Time zones
14. App architecture and state
15. Performance, accessibility and testing
16. Quick reference
17. Disclaimer

Before you start, install the package as described in the **Chronoverse Installation Guide**.

# 1 Concepts and imports

Chronoverse has two layers:

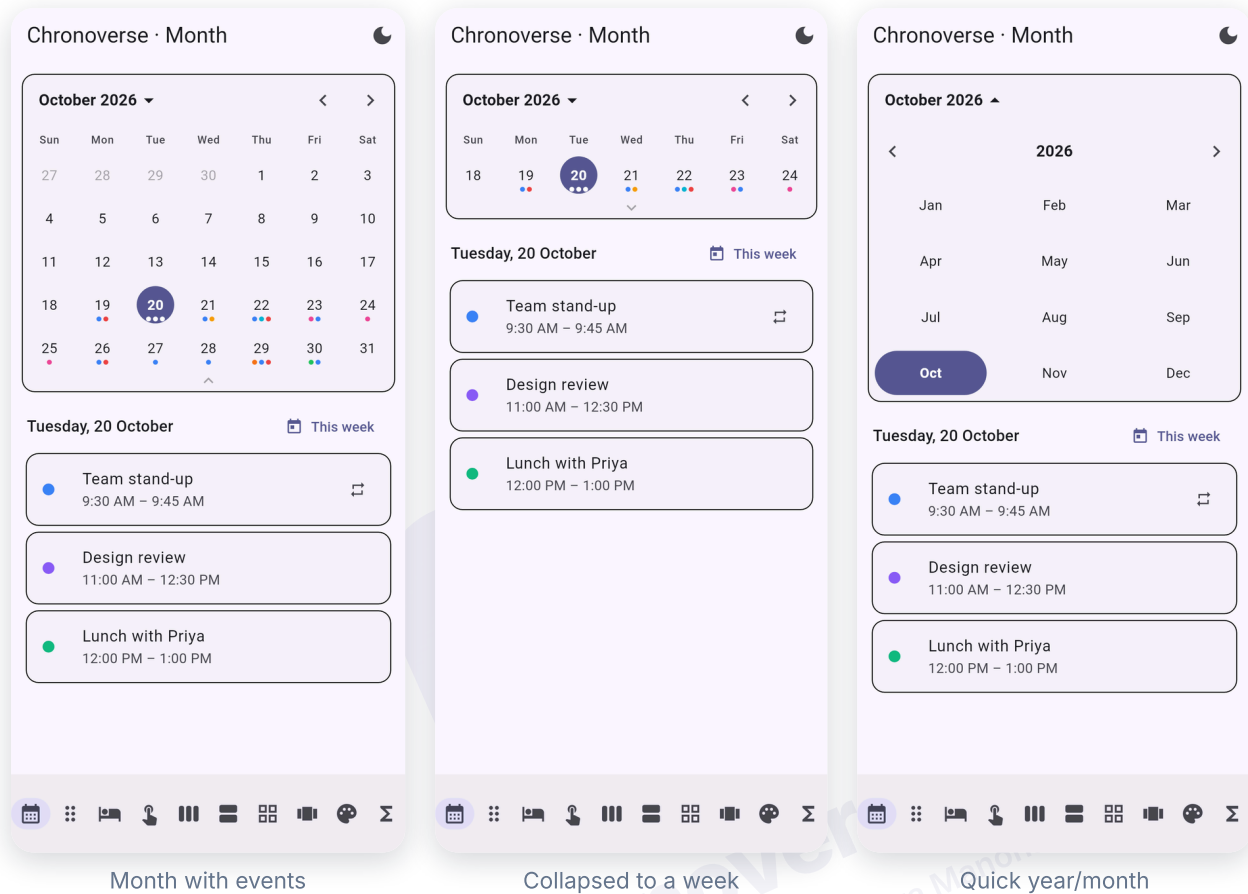
| Import                                            | Contains                                                                                                                  | Use it for                                                    |
|---------------------------------------------------|---------------------------------------------------------------------------------------------------------------------------|---------------------------------------------------------------|
| <code>package:chronoverse/chronoverse.dart</code> | Everything: widgets, dialogs, theme, plus the whole core.                                                                 | Flutter apps.                                                 |
| <code>package:chronoverse/core.dart</code>        | Pure Dart: <code>Chrono</code> , extensions, formatting, parsing, conversions, time zones, events, recurrence, iCalendar. | Business logic, servers, command-line tools, isolates, tests. |

Naming is consistent: widgets start with `Chrono` ( `ChronoCalendar` ), dialogs with `showChrono`, and data classes describe what they hold ( `DateRange`, `CalendarEvent`, `RecurrenceRule` ). All dates are plain `DateTime` values; day-based APIs ignore the time part.

Every option has a sensible default. Start with the smallest code and add options only when you need them.

**Chronoverse**  
Developed by Sanket Santosh Vidya Manohar

## 2 The month calendar



```
ChronoCalendar(  
  initialDate: DateTime(2026, 10, 1),  
  firstDay: DateTime(2020), // navigation limits  
  lastDay: DateTime(2030, 12, 31),  
  mode: ChronoSelectionMode.single, // none, single, multiple, range  
  initialFormat: ChronoCalendarFormat.month, // month, twoWeeks, week  
  expandable: true, // handle to expand/collapse  
  formatToggle: true, // month / 2 weeks / week button  
  showWeekNumbers: true,  
  sixWeekMonths: true, // constant height  
  holidayPredicate: (d) => d.month == 12 && d.day == 25,  
  onDaySelected: (day) => print(day),  
  onPageChanged: (focused) => print('Now showing $focused'),  
)
```

### What users can do

- **Swipe** left or right (touch, mouse drag or trackpad), or use the arrows, to change month or week.
- **Tap the title** to open a year grid, then a month grid, to jump anywhere quickly.
- **Drag the handle** down to expand to a month, up to collapse to a week.

- **Use the keyboard:** arrows move between days, Page Up / Page Down change the page.
- Tap **"today"** in the header to return to the current date.

## Header and visibility options

| Option                       | Default     | Effect                                                          |
|------------------------------|-------------|-----------------------------------------------------------------|
| <code>showHeader</code>      | true        | Title, arrows and today button.                                 |
| <code>showWeekdays</code>    | true        | Weekday names row.                                              |
| <code>showOutsideDays</code> | true        | Days of neighbouring months.                                    |
| <code>showTodayButton</code> | true        | Shortcut back to today.                                         |
| <code>quickPickers</code>    | true        | Tap the title for year and month grids.                         |
| <code>swipeEnabled</code>    | true        | Allow swiping between pages.                                    |
| <code>firstDayOfWeek</code>  | from locale | <code>DateTime.monday</code> ... <code>DateTime.sunday</code> . |
| <code>weekendDays</code>     | from locale | Which days are coloured as weekend.                             |
| <code>headerBuilder</code>   | null        | Replace the header completely.                                  |

**Chronoverse**  
Developed by Sanket Santosh Vidya Manohar

### 3 Controlling the calendar from code

Create a `ChronoCalendarController` to read and change the calendar from anywhere. It is a `ChangeNotifier`, so you can listen to it.

```
final controller = ChronoCalendarController(
  focusedDay: DateTime.now(),
  format: ChronoCalendarFormat.week,
  mode: ChronoSelectionMode.single,
);

ChronoCalendar(controller: controller, expandable: true);

controller.showWeekOf(DateTime(2026, 12, 21)); // collapse to that exact week
controller.expand();                          // full month
controller.collapse();                        // back to the focused week
controller.toggleExpanded();
controller.setFormat(ChronoCalendarFormat.twoWeeks);
controller.jumpTo(DateTime(2027, 3, 1));
controller.nextPage();
controller.previousPage();
controller.today();

controller.select(DateTime(2026, 10, 20));
controller.selectRange(DateRange(DateTime(2026, 10, 1), DateTime(2026, 10, 7)));
controller.selectDays([DateTime(2026, 10, 2), DateTime(2026, 10, 9)]);
controller.clearSelection();

controller.selectedDay;    // DateTime?
controller.selectedDays;  // Set<DateTime>
controller.selectedRange; // DateRange?
controller.isExpanded;

controller.addListener(() => print(controller.focusedDay));
// Dispose it in your State.dispose().
```

If you do not pass a controller, the calendar creates and disposes its own.

## 4 Selection, availability, freeze and disable

### Selection modes

| Mode                                      | Callback                                 | Result                                             |
|-------------------------------------------|------------------------------------------|----------------------------------------------------|
| <code>ChronoSelectionMode.none</code>     | <code>onDaySelected</code><br>(optional) | Taps are reported, nothing stays selected.         |
| <code>ChronoSelectionMode.single</code>   | <code>onDaySelected</code>               | <code>DateTime</code>                              |
| <code>ChronoSelectionMode.multiple</code> | <code>onDaysSelected</code>              | <code>Set&lt;DateTime&gt;</code>                   |
| <code>ChronoSelectionMode.range</code>    | <code>onRangeSelected</code>             | <code>DateRange?</code> (null after the first tap) |

```
ChronoCalendar(  
  mode: ChronoSelectionMode.range,  
  maxRangeDays: 14,           // days beyond the limit are greyed while picking  
  onRangeSelected: (range) {  
    if (range != null) print('${range.lengthInDays} days');  
  },  
)
```

### Which days can be picked

`DateAvailability` describes blocked and allowed days in one object. It is shared by the calendar, the dialogs, the date field, the strip and the vertical calendar.

```
final availability = DateAvailability(  
  disablePast: true,  
  disabledWeekdays: {DateTime.sunday},  
  disabledDates: [DateTime(2026, 12, 25)],  
  disabledRanges: [DateRange(DateTime(2026, 11, 3), DateTime(2026, 11, 6))],  
  predicate: (day) => day.day != 13,    // any rule  
);  
  
availability.isAvailable(DateTime(2026, 12, 25)); // false  
availability.isRangeAvailable(someRange);         // false if any day inside is blocked  
availability.nextAvailable(DateTime.now());        // first free day  
  
ChronoCalendar(availability: availability);
```

To allow *only* certain days, use `enabledDates`. A range that would cross a blocked day is rejected automatically.

## Freeze, read-only and disabled

| State     | How                              | Navigation | Selection | Look       |
|-----------|----------------------------------|------------|-----------|------------|
| Normal    | default                          | yes        | yes       | normal     |
| Frozen    | <code>controller.freeze()</code> | no         | no        | normal     |
| Read-only | <code>readOnly: true</code>      | yes        | no        | normal     |
| Disabled  | <code>enabled: false</code>      | no         | no        | greyed out |

Freezing is ideal while a request is in progress: call `freeze()` before it and `unfreeze()` after. Check the state with `controller.isFrozen`.



## 5 Events on the calendar

Pass a list of `CalendarEvent` objects, or an `eventLoader` that returns items for a day. Repeating events are expanded automatically and indexed once per visible page, so even thousands of events stay fast.

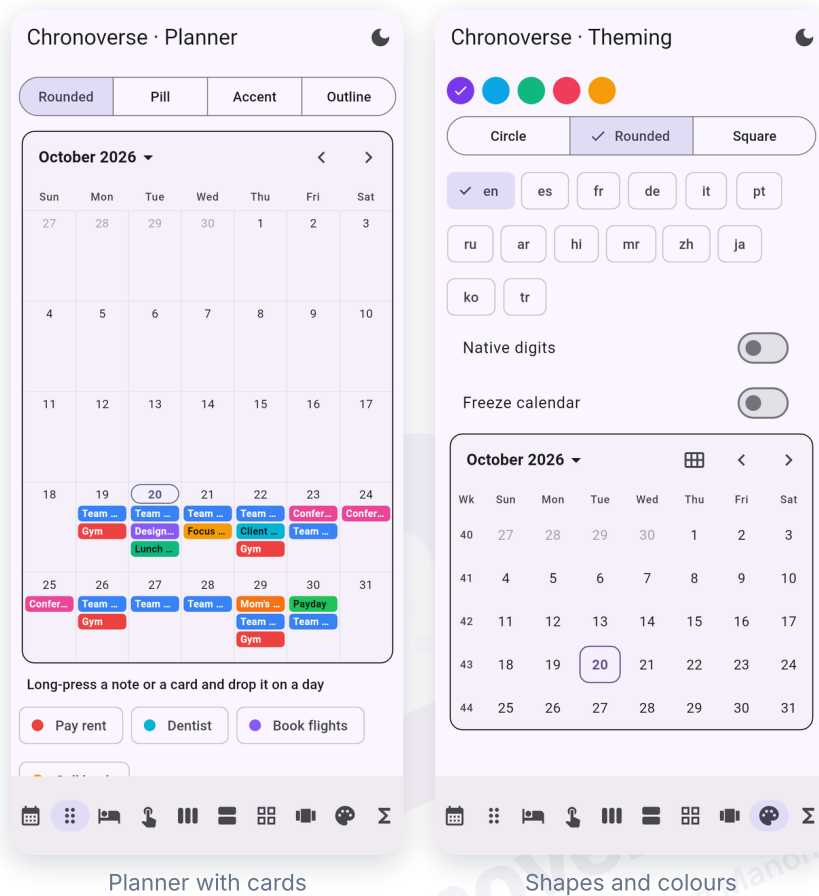
```
final events = [
  CalendarEvent.allDay(id: '1', title: 'Holiday', start: DateTime(2026, 10, 2), color:
    0xFF10B981),
  CalendarEvent(
    id: '2',
    title: 'Design review',
    start: DateTime(2026, 10, 14, 10),
    end: DateTime(2026, 10, 14, 11, 30),
    color: 0xFF6366F1,
    recurrence: RecurrenceRule.weekly,
  ),
];

ChronoCalendar(
  events: events,
  markerStyle: ChronoMarkerStyle.dots, // dots, bars or chips
  onDaySelected: (day) => print(events.on(day).map((o) => o.event.title)),
);

// Or load anything you like:
ChronoCalendar(eventLoader: (day) => myDatabase.itemsFor(day));
```

Each event's `color` is used for its marker; otherwise `markerColor` or the theme's `eventColor` is used.

## 6 Planner cards and drag & drop



Set `markerStyle: ChronoMarkerStyle.chips` to show titled, coloured cards inside each day. Give the cells more height so the cards fit.

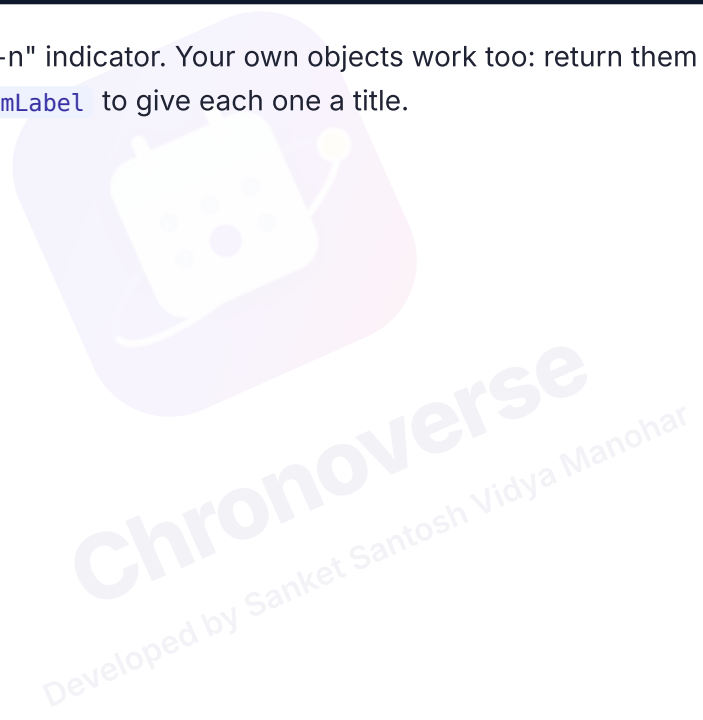
```
ChronoCalendar(  
  events: events,  
  markerStyle: ChronoMarkerStyle.chips,  
  chipStyle: const ChronoChipStyle(  
    shape: ChronoChipShape.accentBar, // rounded, pill, square, accentBar, outlined  
    height: 16,  
    fontSize: 10,  
  ),  
  theme: const ChronoThemeData(cellHeight: 92, dayShape: ChronoDayShape.square, cellPadding:  
0),  
  onTap: (item) => openDetails(item),  
  onDrop: (day, data) => handleDrop(day, data),  
  canDrop: (day, data) => !day.isWeekend(), // optional rule  
)
```

### Dragging notes in from your UI

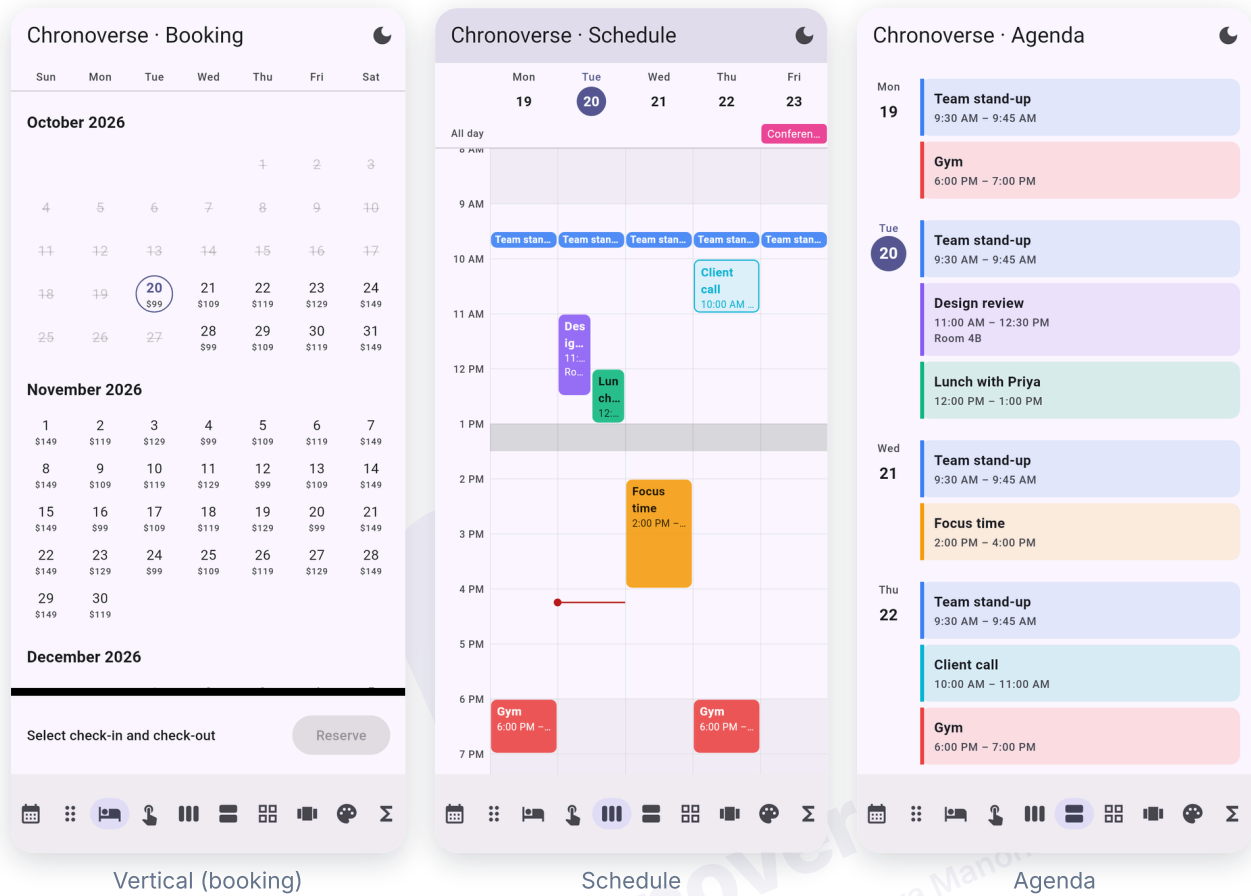
Wrap any widget in `ChronoDragItem`. Long-press (or drag directly with `longPress: false`) and drop it on a day. The target day highlights while hovering.

```
ChronoDragItem<String>(  
  data: 'Pay rent',  
  child: Chip(label: Text('Pay rent')),  
);  
  
void handleDrop(DateTime day, Object data) {  
  if (data is String) {  
    setState(() => events.add(CalendarEvent.allDay(id: uuid(), title: data, start: day,  
color: 0xFFF59E0B)));  
  } else if (data is EventOccurrence) {  
    // A card was moved from another day: shift the event.  
    final shift = data.start.daysUntil(day);  
    // update data.event.start / end by shift days and save  
  }  
}
```

Cards that do not fit show a "+n" indicator. Your own objects work too: return them from `eventLoader` and provide `itemLabel` to give each one a title.



## 7 More views



### Vertical calendar (bookings)

```
ChronoVerticalCalendar(
  mode: ChronoSelectionMode.range,
  availability: DateAvailability(disablePast: true),
  lastDay: DateTime.now().addMonths(6),
  dayLabelBuilder: (context, day) => Text('\${priceFor(day)}'),
  onRangeSelected: (range) => setState(() => stay = range),
)
```

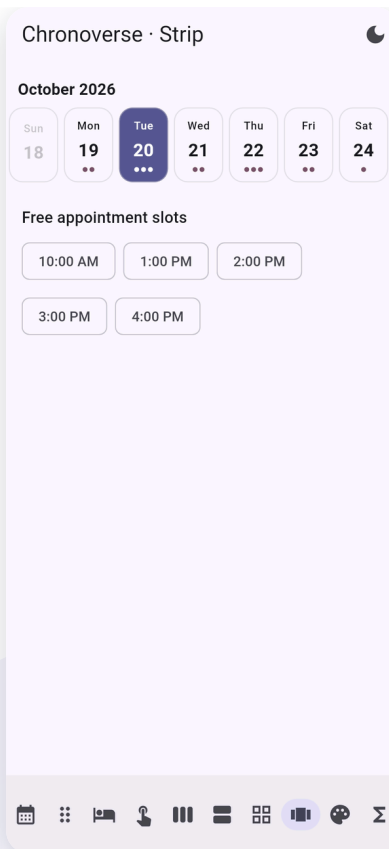
## Schedule (day, 3-day, week)

```
ChronoScheduleView(  
  date: DateTime.now().startOfWeek(),  
  numberOfDays: 5,  
  events: events,  
  startHour: 7, endHour: 21, hourHeight: 56,  
  initialScrollHour: 8,  
  workHours: (9, 18),  
  blockedSlots: [DateRangeTime(lunchStart, lunchEnd)],  
  snap: const Duration(minutes: 15),  
  enableDrag: true, enableResize: true,  
  onEventMoved: (occurrence, start, end) => save(occurrence.event.copyWith(start: start,  
end: end)),  
  onSlotTap: (time) => createEventAt(time),  
)
```

Overlapping events are laid out side by side, all-day events get their own row, and a line marks the current time. Long-press an event to move it; drag its bottom edge to resize.

## Year view, week strip, agenda and heatmap

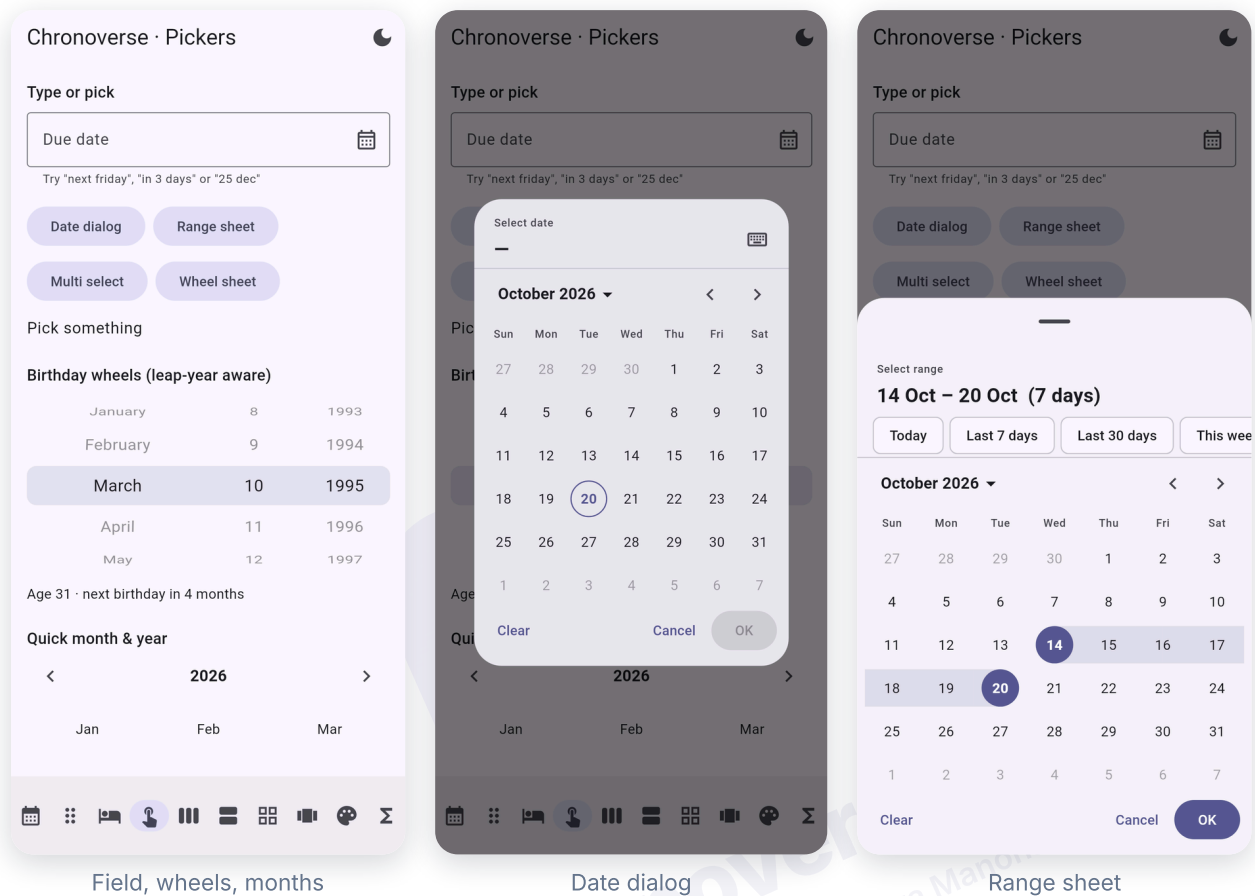
```
ChronoYearView(year: 2026, onMonthTap: openMonth, eventLoader: (d) => events.on(d));  
  
ChronoWeekStrip(selectedDate: day, onDateSelected: (d) => setState(() => day = d));  
  
ChronoAgendaView(events: events, range: DateRange(today, today.addDays(14)), onEventTap:  
open);  
  
ChronoHeatmap(values: {DateTime(2026, 10, 1): 4, DateTime(2026, 10, 2): 9}, onDayTap: (day,  
value) => showDay(day));
```



Chronoverse

Developed by Sanket Santosh Vidya Manohar

## 8 Pickers and typed input



### Dialogs and sheets

```
final date = await showChronoDatePicker(
  context: context,
  initialDate: DateTime.now(),
  availability: DateAvailability(disablePast: true),
  presentation: ChronoPresentation.dialog, // or bottomSheet
  allowTyping: true, // switch to a text field
);

final range = await showChronoRangePicker(
  context: context,
  presets: ChronoRangePreset.common, // Today, Last 7 days, This month...
  maxDays: 30,
  presentation: ChronoPresentation.bottomSheet,
);

final days = await showChronoMultiDatePicker(context: context);
final birthday = await showChronoWheelPicker(context: context, lastDate: DateTime.now());
```

Custom presets: `ChronoRangePreset('Next 3 days', (today) => DateRange(today, today.addDays(2)))`.

## Typed date field

```
ChronoDateField(  
  pattern: 'dd/MM/yyyy', // mask and validation  
  firstDate: DateTime(1900),  
  lastDate: DateTime.now(),  
  allowNaturalLanguage: true, // "next friday", "in 3 days"  
  showPickerButton: true,  
  required: true,  
  decoration: const InputDecoration(labelText: 'Date of birth'),  
  onChanged: (date) => form.dob = date, // null while invalid  
)
```

It works inside a `Form` and shows localized messages for invalid or out-of-range dates.

## Wheels and quick grids

```
ChronoDateWheelPicker(  
  initialDate: DateTime(1995, 3, 10),  
  order: ChronoWheelOrder.dmy, // dmy, mdy, ymd  
  lastDate: DateTime.now(),  
  onChanged: (d) => print(d), // Feb 29 → Feb 28 in non-leap years  
);  
ChronoMonthPicker(year: 2026, onMonthSelected: (m) => print(m));  
ChronoYearPicker(firstYear: 1950, lastYear: 2050, selectedYear: 2026, onYearSelected:  
  print);
```

Chrono  
Developed by Sanket Santosh Vidya Manohar

## 9 Theming

`ChronoThemeData` controls every visual detail. Unset colours are taken from your app's `ColorScheme`, so light and dark mode work automatically.

```
// App-wide
MaterialApp(
  theme: ThemeData(
    colorSchemeSeed: Colors.teal,
    extensions: const [ChronoThemeData(dayShape: ChronoDayShape.rounded, dayRadius: 12,
    cellHeight: 48)],
  ),
);

// One widget
ChronoCalendar(theme: ChronoThemeData.fromSeed(Colors.deepOrange));

// A subtree
ChronoTheme(data: const ChronoThemeData(dayShape: ChronoDayShape.square), child: myScreen);
```

| Group   | Properties                                                                                                                                                                                                                                                                                                                          |
|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Colours | <code>primaryColor</code> <code>onPrimaryColor</code> <code>backgroundColor</code> <code>textColor</code> <code>mutedTextColor</code><br><code>todayColor</code> <code>rangeColor</code> <code>disabledColor</code> <code>weekendTextColor</code> <code>holidayTextColor</code><br><code>eventColor</code> <code>borderColor</code> |
| Shape   | <code>dayShape</code> (circle, rounded, square, stadium) <code>dayRadius</code> <code>todayBorderWidth</code> <code>decoration</code>                                                                                                                                                                                               |
| Size    | <code>cellHeight</code> <code>cellPadding</code> <code>headerHeight</code> <code>weekdayRowHeight</code> <code>markerSize</code> <code>maxMarkers</code>                                                                                                                                                                            |
| Text    | <code>dayTextStyle</code> <code>headerTextStyle</code> <code>weekdayTextStyle</code> <code>weekdayStyle</code> (narrow, short, full)                                                                                                                                                                                                |
| Motion  | <code>animationDuration</code> <code>animationCurve</code>                                                                                                                                                                                                                                                                          |

### Custom cells

Builders receive a `ChronoDayState` ( `date`, `isToday`, `isSelected`, `isRangeStart`, `isRangeEnd`, `isInRange`, `isDisabled`, `isOutside`, `isWeekend`, `isHoliday`, `markerColors`, `items` ). Return `null` to keep the default cell.

```
ChronoCalendar(
  dayBuilder: (context, s) => s.isHoliday ? HolidayCell(s.date) : null,
  dayLabelBuilder: (context, day) => Text('•'),
  weekdayBuilder: (context, weekday) => Text('$weekday'),
)
```

## 10 Languages and right-to-left

Fourteen languages are built in: `en es fr de it pt ru ar hi mr zh ja ko tr`. Each one provides month and weekday names, the first day of the week, weekend days, date order, plural rules, relative-time phrases, button labels and native digits.

```
// Follows the app's locale automatically, or set it:
ChronoCalendar(locale: ChronoLocale.of('ar'), nativeDigits: true);

Chrono.format(DateTime.now(), 'EEEE d MMMM yyyy', locale: 'de'); // Dienstag 20 Oktober
2026
Chrono.timeAgo(DateTime.now().subtract(const Duration(hours: 5)), locale: 'fr');
```

Right-to-left languages mirror automatically under a right-to-left `Directionality`, including swipe direction and arrow keys.

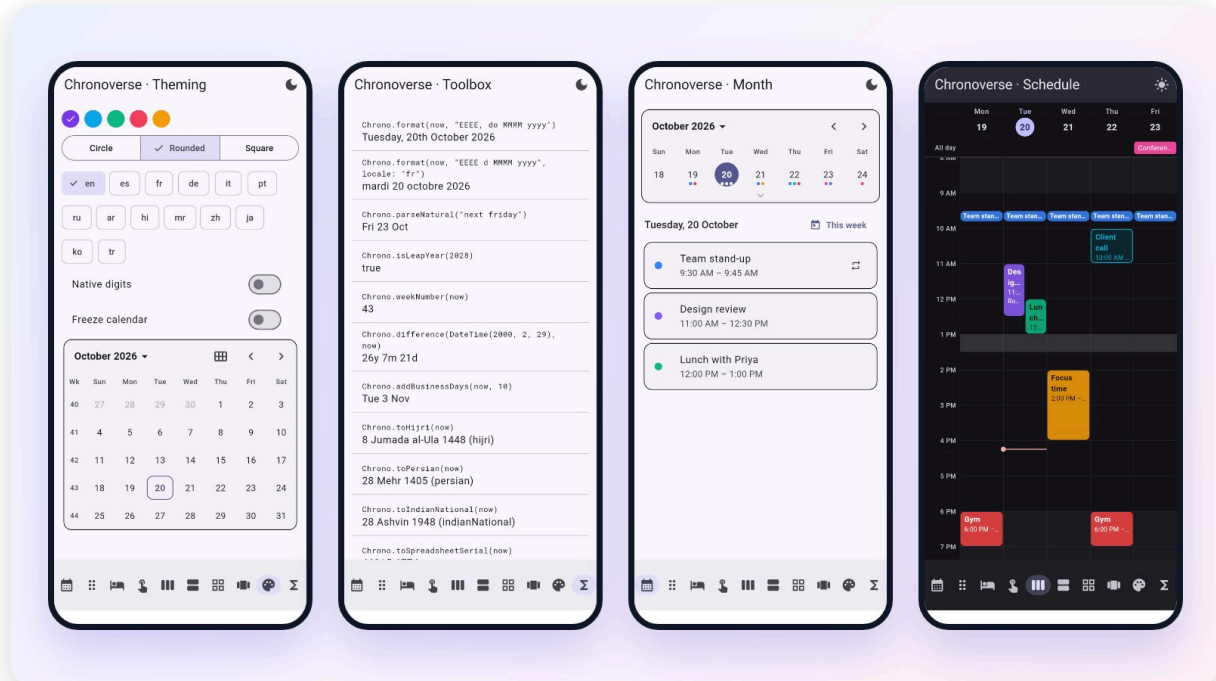
### Adding a language

Create a `ChronoLocale` with your translations (copy a built-in one as a starting point) and register it once at start-up:

```
ChronoLocale.register(myLocale);
ChronoLocale.of(myLocale.code); // now available everywhere
```

# 11 The Chrono toolbox

**Chrono** collects every date and time function in one place. Most are also available as **DateTime** extensions.



| Area           | Functions                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Basics         | <code>now</code> <code>today</code> <code>tomorrow</code> <code>yesterday</code> <code>isLeapYear</code> <code>daysInMonth</code> <code>daysInYear</code> <code>weekNumber</code> <code>weeksInYear</code> <code>quarter</code> <code>dayOfYear</code> <code>isWeekend</code> <code>isPast</code> <code>isFuture</code> <code>isSameDay</code> <code>isValidDate</code> <code>isValidYmd</code>                                                 |
| Arithmetic     | <code>addDays</code> <code>addWeeks</code> <code>addMonths</code> <code>addYears</code> <code>addBusinessDays</code> <code>businessDaysBetween</code> <code>daysBetween</code> <code>weeksBetween</code> <code>monthsBetween</code> <code>yearsBetween</code> <code>difference</code> <code>age</code> <code>nextAnniversary</code> <code>datesBetween</code> <code>monthMatrix</code> <code>easter</code> <code>nthWeekday</code>              |
| Format & parse | <code>format</code> <code>parse</code> <code>parseNatural</code> <code>timeAgo</code> <code>formatDuration</code> <code>formatHms</code> <code>countdown</code> <code>monthNames</code> <code>weekdayNames</code>                                                                                                                                                                                                                               |
| Conversions    | <code>toHijri</code> <code>toPersian</code> <code>toIndianNational</code> <code>toBuddhist</code> <code>toJulianCalendar</code> <code>fromCalendar</code> <code>toJulianDay</code> <code>toUnix</code> <code>fromUnix</code> <code>toUnixMillis</code> <code>fromUnixMillis</code> <code>toSpreadsheetSerial</code> <code>fromSpreadsheetSerial</code> <code>atOffset</code> <code>convertOffset</code> <code>toIso</code> <code>fromIso</code> |
| Time of day    | <code>minutesOfDay</code> <code>isTimeBetween</code> <code>roundTime</code> <code>partOfDay</code>                                                                                                                                                                                                                                                                                                                                              |
| Extras         | <code>zodiacSign</code> <code>chineseZodiac</code> <code>moonPhase</code> <code>moonPhaseName</code>                                                                                                                                                                                                                                                                                                                                            |

## Examples

```
Chrono.addMonths(DateTime(2025, 1, 31), 1); // 2025-02-28, never spills into March
Chrono.addYears(DateTime(2024, 2, 29), 1); // 2025-02-28
Chrono.age(DateTime(2000, 2, 29)); // whole years, leap-day safe
Chrono.difference(start, end); // years, months, days, hours...
Chrono.businessDaysBetween(a, b, holidays: holidays);
Chrono.isTimeBetween(now, '22:00', '06:00'); // night shift, crosses midnight
Chrono.format(now, "EEE, d MMM yyyy 'at' h:mm a");
Chrono.parse('12/25/26', 'MM/dd/yy');
Chrono.parseNatural('end of month');
Chrono.toPersian(now); // CalendarDate(year, month, day)
Chrono.fromCalendar(CalendarDate(CalendarSystem.hijri, 1448, 9, 1));
```

## Format pattern letters

| Token         | Meaning             | Token    | Meaning              |
|---------------|---------------------|----------|----------------------|
| yyyy yy       | year                | HH H     | hour 0–23            |
| MMMM MMM MM M | month name / number | hh h     | hour 1–12            |
| dd d do       | day / ordinal day   | mm m     | minute               |
| EEEE EEE E    | weekday name        | ss s SSS | second / millisecond |
| Q             | quarter             | a        | AM / PM              |
| w             | ISO week            | Z        | UTC offset           |
| D             | day of year         | 'text'   | literal text         |

## 12 Events, recurrence and iCalendar

`CalendarEvent` carries everything a full calendar system needs.

| Field                                                                                                   | Purpose                                                                                                                                                |
|---------------------------------------------------------------------------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------|
| <code>id</code> <code>title</code> <code>description</code><br><code>location</code> <code>color</code> | Basics.                                                                                                                                                |
| <code>start</code> <code>end</code> <code>isAllDay</code><br><code>timeZone</code>                      | When, in which zone.                                                                                                                                   |
| <code>recurrence</code> <code>exceptionDates</code><br><code>additionalDates</code>                     | Repeating rules, skipped and extra dates.                                                                                                              |
| <code>organizer</code> <code>attendees</code>                                                           | People, with responses, optional flag and resources (rooms).                                                                                           |
| <code>reminders</code>                                                                                  | Notification, email or popup, any time before.                                                                                                         |
| <code>status</code> <code>visibility</code> <code>showAs</code><br><code>type</code>                    | Confirmed / tentative / cancelled, public / private, busy / free, normal / out of office / focus time / working location / birthday / task / reminder. |
| <code>url</code> <code>meetingUrl</code> <code>calendarId</code><br><code>data</code>                   | Links, owning calendar, and any custom payload of your own type.                                                                                       |

```
final rule = RecurrenceRule.parse('FREQ=MONTHLY;BYDAY=-1FR;COUNT=12');
rule.describe(); // "Every month on the last Fri, 12 times"
rule.occurrences(DateTime(2026, 1, 30)).take(3);

RecurrenceRule.weekly.copyWith(byWeekdays: [ByWeekday(DateTime.monday),
ByWeekday(DateTime.thursday)]);
RecurrenceRule.weekdays; // also daily, weekly, monthly, yearly

events.occurrencesIn(DateRange.month(DateTime.now())); // expanded, sorted
events.groupByDay(DateRange.month(DateTime.now())); // Map<DateTime,
List<EventOccurrence>>
events.search('review');
events.involving('sam@example.com');

Schedule.freeSlots(events.on(today), from: today.atTime(9), to: today.atTime(17));
Schedule.slots(from: today.atTime(9), to: today.atTime(17),
length: const Duration(minutes: 30), busy: events.on(today));

final text = ICalendar.encode(events, calendarName: 'Team'); // .ics export
final imported = ICalendar.decode(text); // .ics import
```

## 13 Time zones

`ChronoTimeZone` includes about 60 common zones with daylight-saving rules for North America, Europe, Australia and New Zealand. Gaps (clocks jump forward) and overlaps (clocks fall back) are handled.

```
final kolkata = ChronoTimeZone.get('Asia/Kolkata');
final newYork = ChronoTimeZone.get('America/New_York');

final meeting = kolkata.fromWall(2026, 10, 20, 19, 30); // ZonedDateTime
kolkata.convert(DateTime(2026, 10, 20, 19, 30), newYork); // same instant in New York
newYork.isDst(DateTime.now().toUtc());
newYork.offsetLabel(DateTime.now().toUtc());           // UTC-04:00
ChronoTimeZone.ids;                                    // every known id

// Add or override a zone
ChronoTimeZone.register(const ChronoTimeZone('Europe/Example', Duration(hours: 1), dst:
DstRule.europe));

// Fixed offsets
ZonedDateTime.tryParse('2026-10-20T09:00:00+05:30');
```

Time-zone rules change by government decision. Keep the package up to date and verify any time-critical conversion. Register a corrected zone with `ChronoTimeZone.register` if a rule changes before an update is available.

Chrono  
Developed by Sanket Santosh

## 14 App architecture and state

---

- **Keep dates in your state, not in widgets.** Store the selected date or range in your own state object and pass it down, or listen to a `ChronoCalendarController`.
- **The controller is a `ChangeNotifier`**, so it fits any state-management approach that works with listenable objects, including plain `setState`, `ValueListenableBuilder` / `ListenableBuilder`, and inherited widgets.
- **Put business rules in the core.** Availability, recurrence and date math come from `core.dart`, so they can be unit-tested without widgets and reused on a server.
- **Store dates in UTC or ISO 8601** (`Chrono.toIso`) and convert to local or zoned time only for display.
- **Use `data`** on `CalendarEvent<T>` to carry your own model, so callbacks give you your object back without lookups.

```
ListenableBuilder(  
  listenable: controller,  
  builder: (context, _) => Text(Chrono.format(controller.focusedDay, 'MMMM yyyy')),  
)
```

**Chronoverse**  
Developed by Sanket Santosh Vidya Manohar

## 15 Performance, accessibility and testing

---

### Performance

- Pass `events` rather than a slow `eventLoader` ; Chronoverse indexes them per page.
- If you use `eventLoader` , make it a fast map lookup (for example from `groupByDay` ).
- Keep the same list instance when nothing changed, and create a new list when something did.
- Use `const ChronoThemeData(...)` where possible.

### Accessibility

- Every day has a screen-reader label with the full date and its state (selected, today, unavailable).
- Text scales with the system font size without overflowing.
- All interaction is available from the keyboard, and colours come from your accessible color scheme.

### Testing your integration

```
testWidgets('picks a day', (tester) async {
  DateTime? picked;
  await tester.pumpWidget(MaterialApp(home: Scaffold(body: ChronoCalendar(
    initialDate: DateTime(2026, 10, 1),
    onDaySelected: (d) => picked = d,
  ))));
  await tester.tap(find.text('15'));
  expect(picked, DateTime(2026, 10, 15));
});
```

Pass a fixed `initialDate` in tests so results do not depend on the current day.

## 16 Quick reference

| I want to...                 | Use                                                                                       |
|------------------------------|-------------------------------------------------------------------------------------------|
| Show a month and pick a day  | <code>ChronoCalendar(onDaySelected: ...)</code>                                           |
| Pick a range                 | <code>mode: ChronoSelectionMode.range</code> or <code>showChronoRangePicker</code>        |
| Pick several days            | <code>mode: ChronoSelectionMode.multiple</code> or <code>showChronoMultiDatePicker</code> |
| Show one week, expand on tap | <code>expandable: true</code> + <code>controller.showWeekOf(day)</code>                   |
| Jump quickly to a year       | Tap the title, or <code>ChronoYearPicker</code>                                           |
| Let users type a date        | <code>ChronoDateField</code>                                                              |
| Birthday input               | <code>ChronoDateWheelPicker</code> or <code>showChronoWheelPicker</code>                  |
| Hotel or travel booking      | <code>ChronoVerticalCalendar</code> + <code>DateAvailability</code>                       |
| Appointment slots            | <code>ChronoWeekStrip</code> + <code>Schedule.slots</code>                                |
| Day or week planner          | <code>ChronoScheduleView</code>                                                           |
| Notes and cards on days      | <code>markerStyle: ChronoMarkerStyle.chips</code> + <code>ChronoDragItem</code>           |
| Upcoming list                | <code>ChronoAgendaView</code>                                                             |
| Habit or activity tracker    | <code>ChronoHeatmap</code>                                                                |
| Block days                   | <code>DateAvailability</code>                                                             |
| Lock the calendar            | <code>controller.freeze()</code> , <code>readOnly</code> , <code>enabled: false</code>    |
| Any date calculation         | <code>Chrono.*</code>                                                                     |
| Import / export calendars    | <code>ICalendar.decode</code> / <code>ICalendar.encode</code>                             |

# 17 Disclaimer

---

*Last updated: 20 October 2026*

This disclaimer applies to the Chronoverse software package, its source code, example application, documentation, guides, images and any other material distributed with it (together, the "**Software**"). By downloading, installing, copying or using the Software in any way you confirm that you have read, understood and accepted this disclaimer. If you do not accept it, do not use the Software.

## 1. No guarantee of correct results

The Software performs calculations and conversions involving dates, times, durations, time zones, daylight-saving time, leap years, week numbers, business days, ages, anniversaries, recurrence rules, calendar systems (including Hijri, Persian, Indian National, Buddhist and Julian), moon phases, zodiac signs, holidays, parsing and formatting of text, and similar operations (together, the "**Calculations**").

**The Calculations may be incomplete, inaccurate, out of date or wrong.** In particular, but without limitation:

- Time-zone offsets and daylight-saving rules are simplified, are not updated automatically and change by government decision without notice.
- Religious, lunar and regional calendars are computed with arithmetic approximations. Actual dates may differ by one or more days from dates declared by religious or civil authorities, or by local observation.
- Natural-language and typed-date parsing may interpret ambiguous input differently from what the user intended.
- Moon phases, zodiac signs and similar values are approximations for informational and entertainment purposes only.
- Translations, month names, weekday names, plural forms and digit systems may contain errors.
- Results may differ between platforms, devices, operating-system versions and system settings.

## 2. You must verify every result

**You, and every developer, publisher and end user of any application that includes the Software, are solely and fully responsible for independently verifying every result produced by the Software before relying on it.** This applies to any use, and especially to use for legal deadlines, contracts, financial or billing periods, payroll, interest, taxation, medical or medication schedules, safety-critical systems, transport and travel bookings, scheduling of appointments or events, religious observance, age verification, compliance, record keeping, or any other purpose where an error could cause loss, harm or inconvenience.

The Software must not be the sole basis for any decision.

## 3. Provided "as is"

The Software is provided "**AS IS**" and "**AS AVAILABLE**", **WITHOUT WARRANTY OF ANY KIND**, express, implied or statutory, including without limitation any warranty of accuracy, reliability, completeness, timeliness, merchantability, fitness for a particular purpose, title, non-infringement, availability, security, compatibility or freedom from errors, defects, viruses or other harmful components. No advice or information, oral or written, obtained from the authors or through the Software creates any warranty.

## 4. No liability

To the maximum extent permitted by applicable law, **the authors, copyright holders, contributors and maintainers of the Software shall not be liable** for any claim, loss, damage or other liability of any kind, whether direct, indirect,

incidental, special, consequential, exemplary or punitive, including without limitation loss of profits, revenue, data, goodwill, business opportunity, missed deadlines, missed appointments, scheduling conflicts, financial penalties, personal injury or property damage, arising out of or in connection with the Software, its Calculations, its documentation, or the use of or inability to use any of them, whether in contract, tort (including negligence), strict liability or otherwise, even if advised of the possibility of such damage.

Where a jurisdiction does not allow the exclusion or limitation of certain warranties or liabilities, the exclusions and limitations above apply to the fullest extent permitted, and any liability that cannot be excluded is limited to the lowest amount permitted by law.

## **5. Your applications**

If you include the Software in an application, website, library or service, you are solely responsible for that product, for how it uses the Software, for testing it, for the results it shows to your users, and for any notices, terms and disclaimers you give your users. You agree to defend, indemnify and hold harmless the authors and contributors from any claim arising from your product or your use of the Software.

## **6. Documentation and examples**

Documentation, guides, screenshots, code samples and the example application are illustrative only. They may not reflect the latest version and are not a promise of any behaviour, feature or result.

## **7. Third-party names and data**

Any time-zone identifier, calendar name, language name or standard referred to by the Software is used only to describe compatibility. No endorsement by, or affiliation with, any organisation is stated or implied.

## **8. Changes**

This disclaimer may be changed at any time by publishing a new version with the Software. The version included with the copy of the Software you use applies to that copy.

## **9. Relation to the licence**

This disclaimer supplements, and does not replace, the LICENSE. If they conflict, the terms that give the authors and contributors the greater protection apply.