Leap Years and Leap-Day Handling in Date Math

> February 29 exists in most but not all years divisible by 4, and a date calculation that gets the leap-year rule wrong is silently off by exactly one day.

The Gregorian calendar approximates a real solar year using a leap-year rule that is more specific than "every four years." A year divisible by 4 is a leap year, unless it is also divisible by 100, unless it is also divisible by 400. This three-part rule exists specifically so that century years like 1900 (not a leap year) and 2000 (a leap year) resolve correctly, and any date calculation that only implements the simple "divisible by 4" version will silently produce a wrong February day count once a century boundary is involved.

Divisible by 4?	Divisible by 100?	by 400?

1900: yes / yes / no -> not a leap year (Feb has 28 days)

2000: yes / yes / yes -> leap year (Feb has 29 days)

Both are divisible by 4 and by 100; only the third check tells them apart.

Century years need the full three-part rule; the simple "divisible by 4" shortcut gets 1900 and similar years wrong.

Why "divisible by 4" alone is not enough

A pure every-four-years rule would add roughly three extra days per four centuries compared to the actual solar year, so the calendar corrects for that by removing the leap day in three out of every four century years (1700, 1800, 1900) while keeping it in the fourth (1600, 2000, 2400). Any date library or hand-written calculation that skips the divisible-by-100/divisible-by-400 exception will compute February as having 29 days in 1900, which is wrong.

Where the error actually shows up

Because the exception only triggers on century years, code that implements the simplified rule passes every ordinary test for decades before it is ever exercised, since 2024, 2028, and similar nearby leap years are correctly identified either way. The error only surfaces on a century-boundary

date, which is exactly the kind of edge case that a fixture-based test suite has to deliberately include rather than discover by accident.

What this means for a leap-day span or anniversary calculation

Any calculation that counts days across a February, adds or subtracts a year from a February 29 date, or reports a person's age when born on a leap day depends on this rule being applied correctly for every year in range, not just recent ones. WhenMath's temporal layer implements the full three-part Gregorian rule and is verified against a leap-day-span fixture that specifically spans a century boundary.

> **Key takeaway.** The Gregorian leap-year rule has three parts, not one. Divisible by 4, except centuries, except centuries also divisible by 400. Century-boundary dates are the only place a simplified rule actually fails.

Frequently asked

Is the year 2100 a leap year?

No. 2100 is divisible by 4 and by 100 but not by 400, so it follows the same exception as 1900 and is not a leap year under the Gregorian calendar.

Does every calendar system use this exact rule?

No, this is specifically the Gregorian calendar's correction. WhenMath's engines work exclusively in the Gregorian calendar, which is the calendar in near-universal civil use today.

How does WhenMath verify this instead of just trusting a library?

The shared temporal adapter is checked against explicit leap-year and leap-day-span fixtures, including a century-boundary case, rather than assuming the underlying date library gets every case right by default.

WhenMath is a browser-local calculation tool, not a legal, payroll, or scheduling authority. This guide explains a calculation convention; it is educational, not a determination that any specific contract, employer, or jurisdiction follows it.