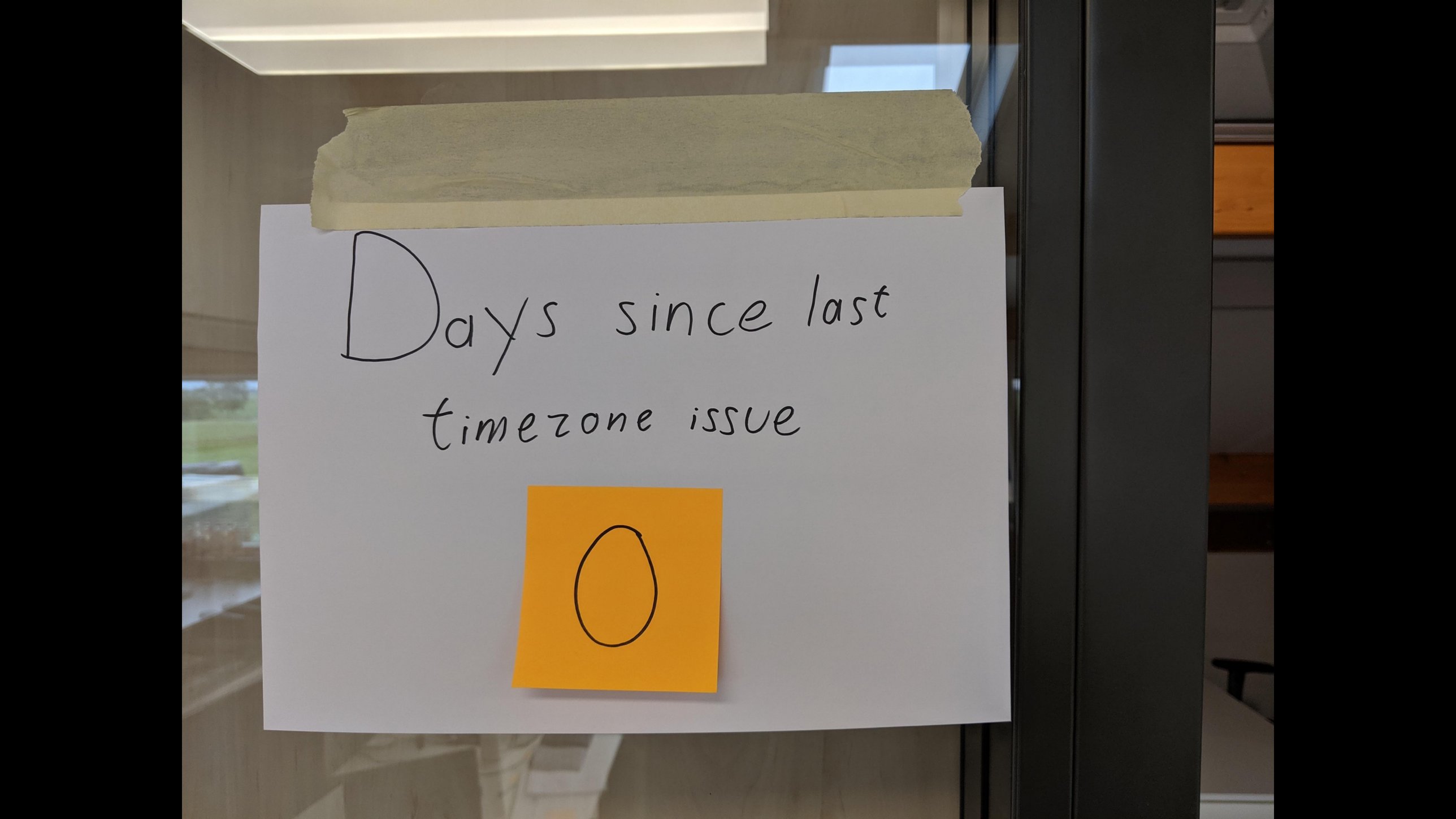
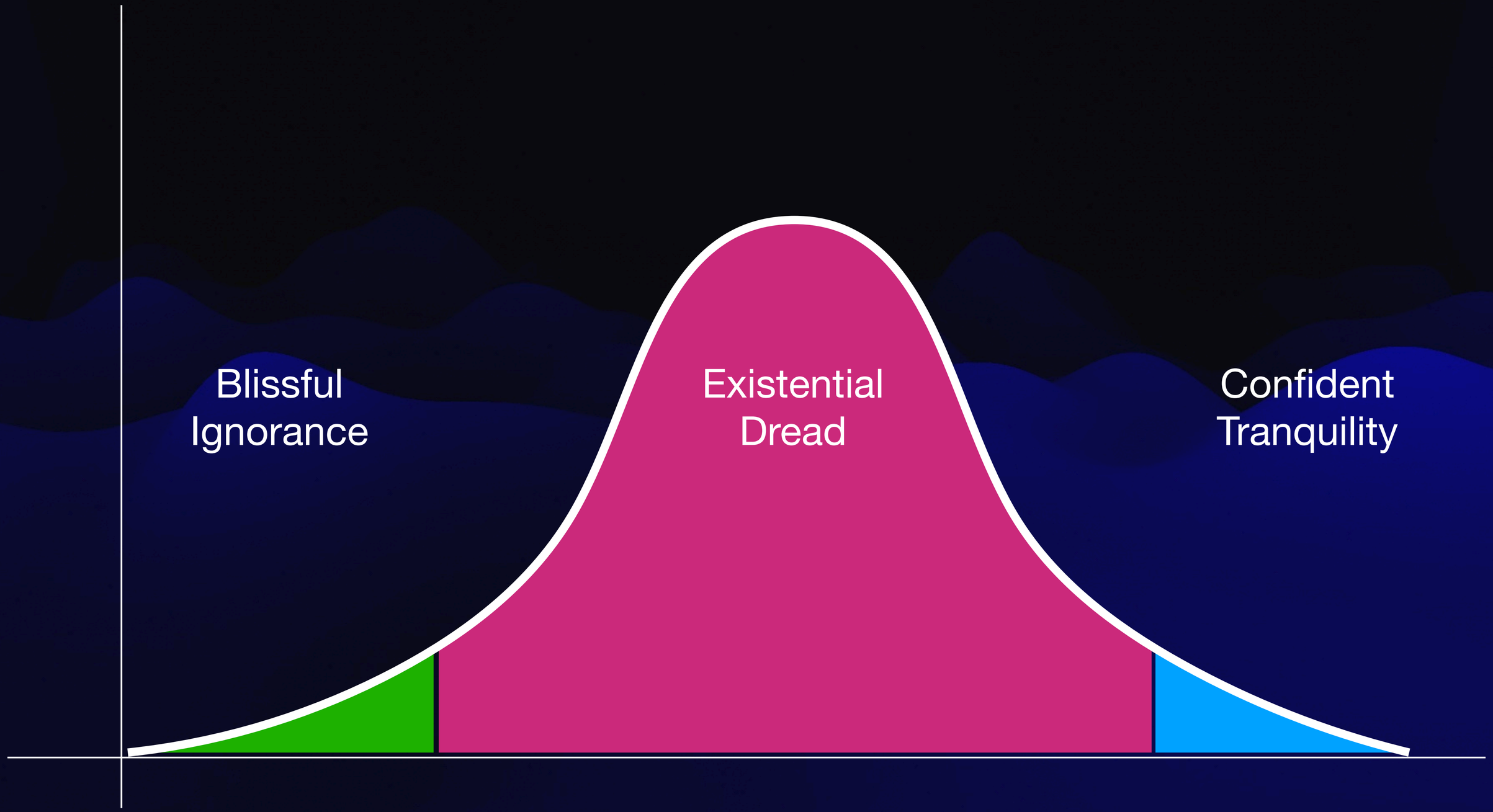
The Temporal Axis of Space-Time

Dave DeLong – Time Lord

Deep Dish Swift 2024


Days since last
timezone issue
0


Blissful
Ignorance
Existential
Dread
Confident
Tranquility

FYI #1
Calendaring is intrinsic to computers

Computus

When is Easter?

FYI #2
Unusual Terminology

FYI #3
Unlocalized

Understand time by understanding *space*

Space and Time

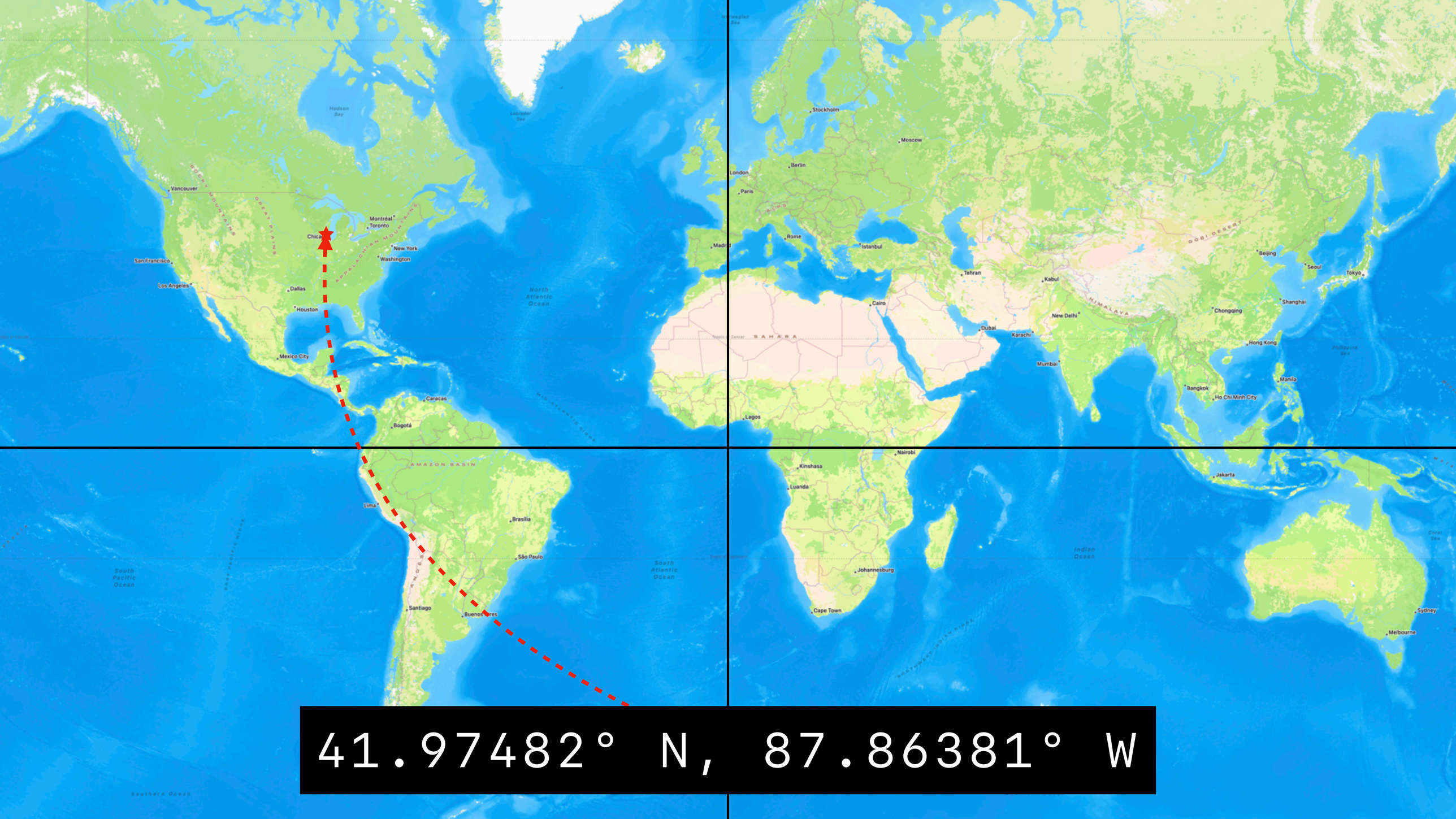
Parallel Concepts

Space	Time
Locator (GPS)	Clock
Coordinates	Timestamp
Meridian + Equator	Epoch
Map	Calendar
Address	Calendar Value
Relative Names	Day Periods + Time Zones

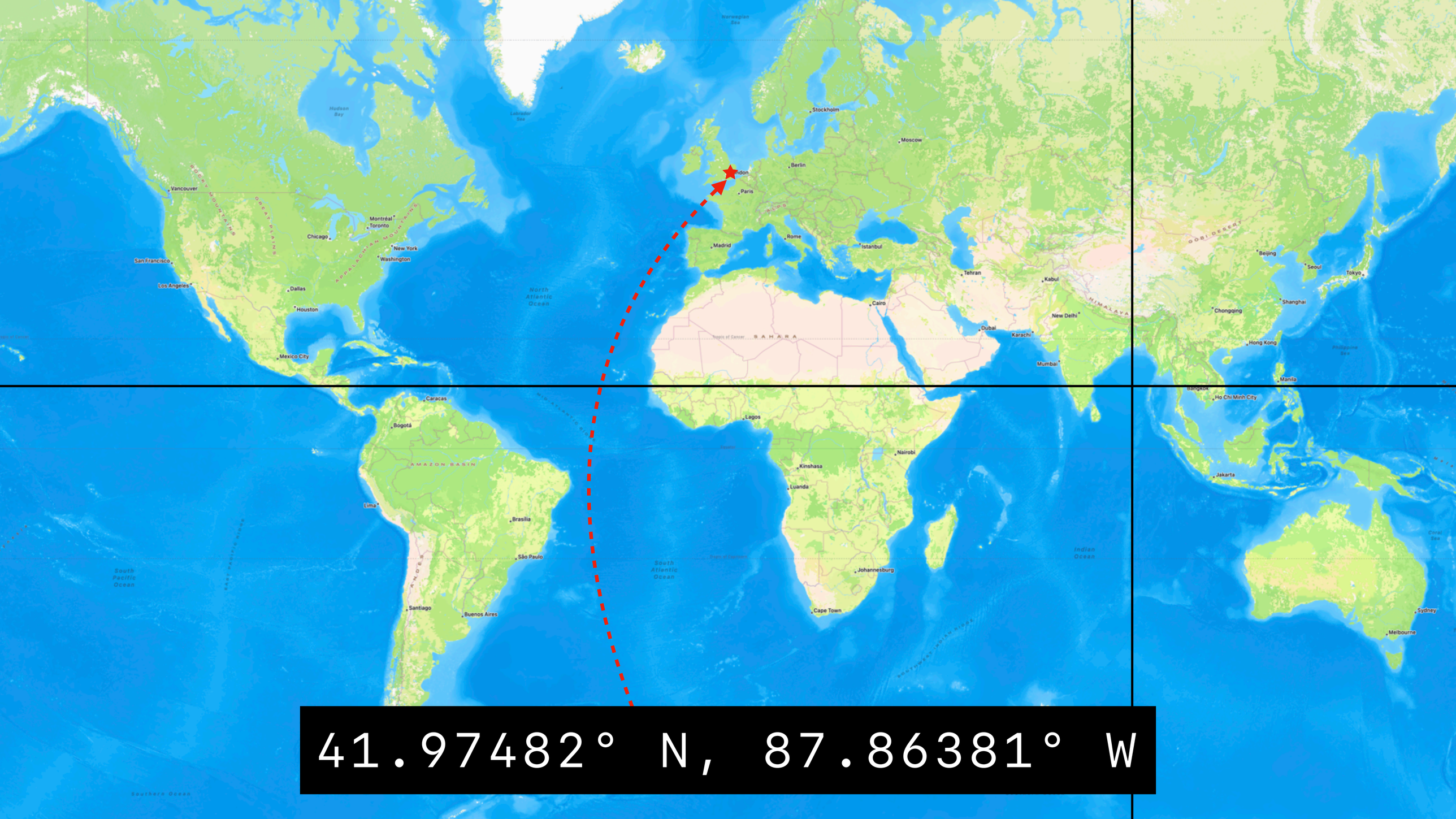
Coordinates and Meridians

`CLLocationCoordinate2D` and `Date`

- Raw numbers
- Require a frame of reference: meridians and epochs
- Different origins produce different results



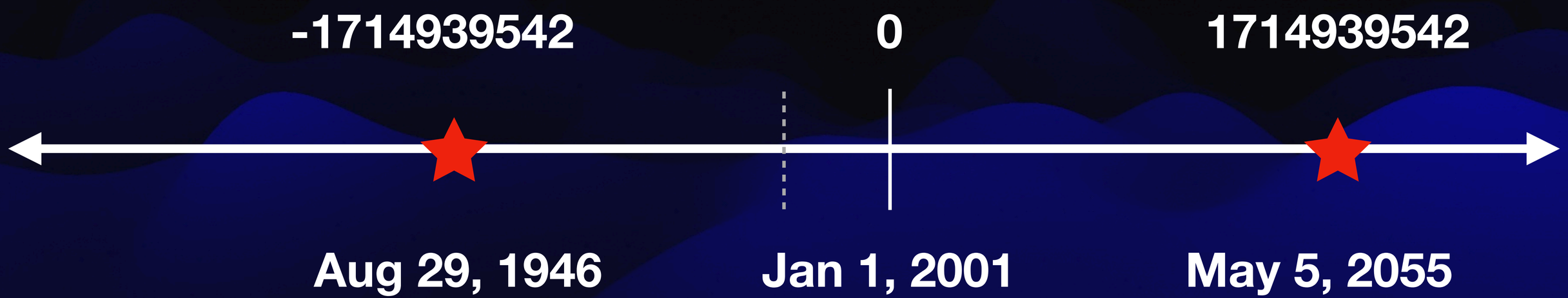
41.97482° N, 87.86381° W


Stockholm
Moscow
Berlin
Paris
Vancouver
Montréal
Toronto
Chicago
New York
Washington
San Francisco
Los Angeles
Dallas
Houston
Mexico City
Madrid
Rome
Istanbul
Tehran
Kabul
Cairo
Dubai
Karachi
New Delhi
Mumbai
Beijing
Seoul
Tokyo
Shanghai
Chongqing
Hong Kong
Manila
Bangkok
Ho Chi Minh City
Jakarta
Caracas
Bogotá
Lima
Brasília
São Paulo
Santiago
Buenos Aires
Lagos
Kinshasa
Luanda
Nairobi
Johannesburg
Cape Town
Sydney
Melbourne
North Atlantic Ocean
South Atlantic Ocean
South Pacific Ocean
Indian Ocean
Southern Ocean
SAHARA
AMAZON BASIN
ANDES
ROCKY MOUNTAINS
GREAT PLAINS
APPALACHIAN MOUNTAINS
HIMALAYA
GOBI DESERT
41.97482° N, 87.86381° W

The Unix Epoch

The Cocoa Epoch

In API: “the reference date”



Coordinates and Meridians

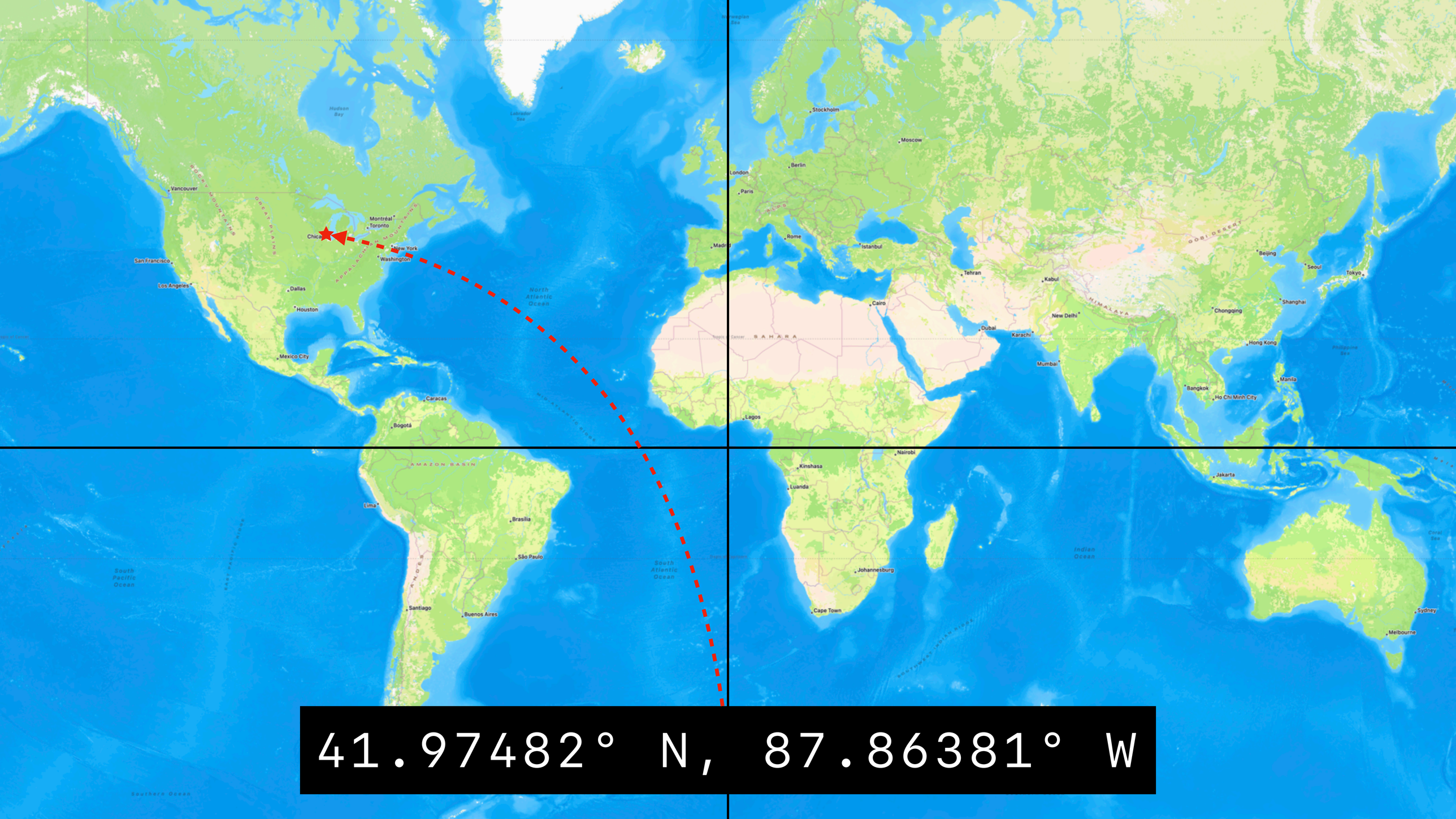
`CLLocationCoordinate2D` and `Date`

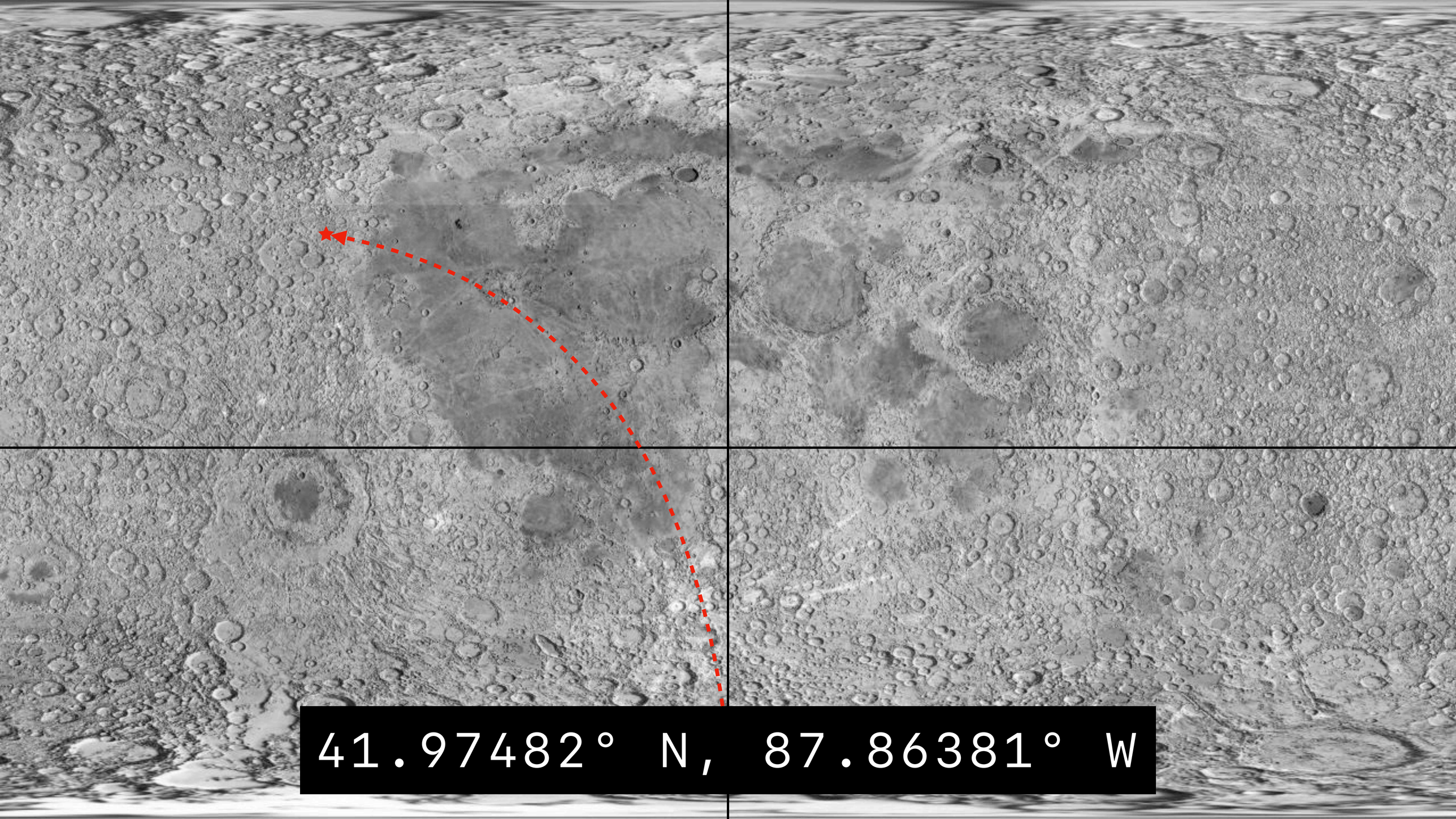
- Raw numbers
- Require a frame of reference: meridians and epochs
- Different origins produce different results
- `CLLocationCoordinate2D` and `CLLocation` are *spatial* coordinates
- `Date` and `TimeInterval` are *temporal* coordinates

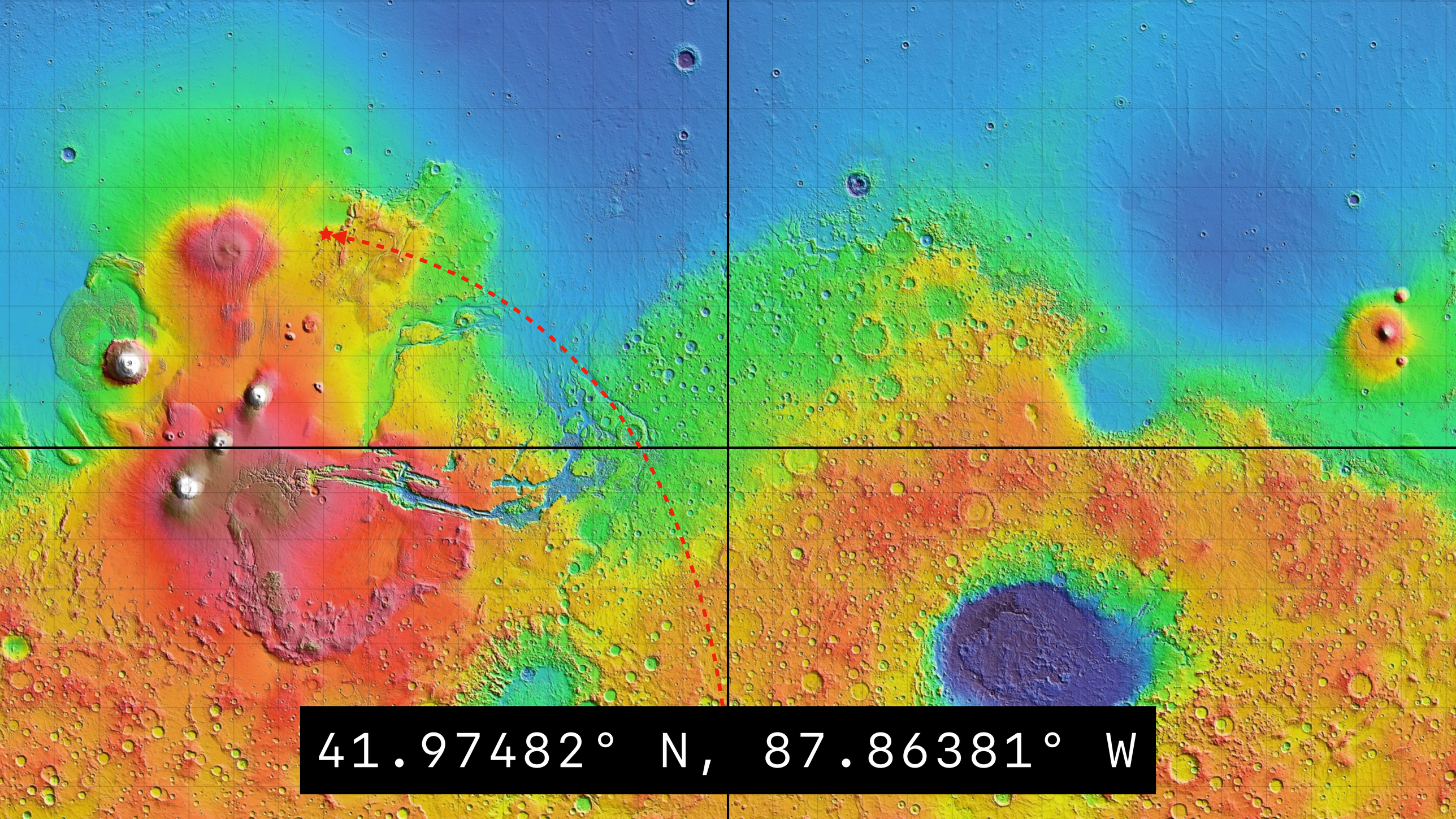
Maps of Space and Time

`CLGeocoder` and `Calendar`

- Interpreting coordinates requires a map
- Different maps produce different results


Hudson Bay
Labrador Sea
Norwegian Sea
North Atlantic Ocean
Vancouver
ROCKY MOUNTAINS
GREAT PLAINS
APPALACHIAN MOUNTAINS
Montréal
Toronto
Chicago
New York
Washington
San Francisco
Los Angeles
Dallas
Houston
Mexico City
Caracas
Bogotá
Stockholm
Moscow
Berlin
London
Paris
Rome
Madrid
Istanbul
Tehran
Kabul
Cairo
SAHARA
Dubai
Karachi
New Delhi
HIMALAYA
GOBI DESERT
Beijing
Seoul
Tokyo
Shanghai
Chongqing
Hong Kong
Mumbai
Manila
Bangkok
Ho Chi Minh City
Jakarta
Lagos
Nairobi
Kinshasa
Luanda
Johannesburg
Cape Town
Indian Ocean
Sydney
Melbourne
AMAZON BASIN
Lima
Brasília
São Paulo
ANDES
Santiago
Buenos Aires
South Pacific Ocean
South Atlantic Ocean
Southern Ocean
41.97482° N, 87.86381° W


41.97482° N, 87.86381° W


41.97482° N, 87.86381° W

Maps of Space and Time

`CLGeocoder` and `Calendar`

- Interpreting coordinates requires a map
- Different maps produce different results
- Calendars are maps of time
- Maps change over time
- Maps are *political*

Yes, “Political”

Humans are messy

- More accurate depictions of borders
- Governments argue about borders
- Shifting borders and disputed areas
- Evolving political affiliations
- Changing timezones
- Last-minute decisions

Addresses

`CNPostalAddress` and `DateComponents`

- Things on maps have an “address”
- Provide human-readable names of significant locations
- Spatial address: `CLPlacemark`, `MKMapItem`, `CNPostalAddress`
- Temporal address: `DateComponents`

Addresses Approximate Hierarchy

AKA “Granularity”

5300 N River Rd
Chicago, IL 60018
United States

5 May 2024
3:05:42 PM

Addresses Approximate Hierarchy

AKA “Granularity”

5300 N River Rd
Chicago, IL 60018
United States

5 May 2024
3:05:42 PM

Relative Names

- “Downtown”, “Main Street”
- “Around the corner from the bakery”
- “This morning”, “Afternoon”
- “Before dinner”
- No meaningful APIs for manipulating these

Truths that Transcend Time

Approaching Temporal Nirvana

Everything is Arbitrary

Arbitrary Names

… are ultimately meaningless

- Names originated in natural world
- Nature is constantly changing
- Definitions no longer based on nature
- ~~1 second = $1/86400$ th of a day~~
- 1 second = 9,192,631,770 hyperfine transitions of the unperturbed ground-state caesium 133 atom

Arbitrary Addresses

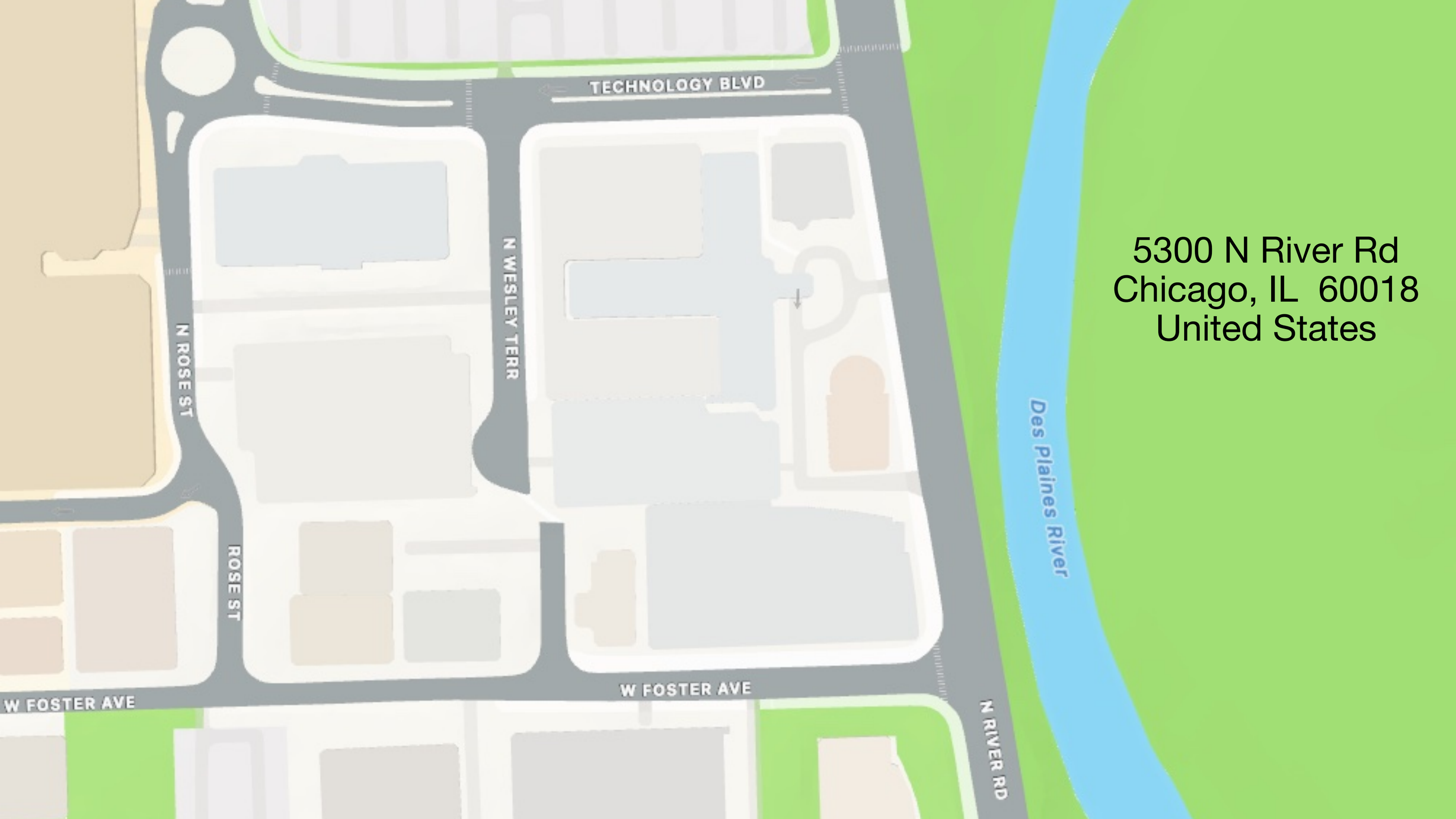
You want consistency? You can’t handle consistency!

2389 546 536 526 518 508 498

Arbitrary Addresses

- There are no true rules around addresses
- 1 AM is not *necessarily* followed by 2 AM
- 28 February is not *necessarily* followed by 1 March
- Maps provide the ultimate truth; use them
- Consider nonsensical unit names

Addresses represent *areas*



5300 N River Rd
Chicago, IL 60018
United States


TECHNOLOGY BLVD
N WESLEY TERR
N ROSE ST
ROSE ST
W FOSTER AVE
W FOSTER AVE
N RIVER RD
Des Plaines River
5300 N River Rd
Chicago, IL 60018
United States


TECHNOLOGY BLVD
N WESLEY TERR
N ROSE ST
ROSE ST
W FOSTER AVE
W FOSTER AVE
N RIVER RD
Des Plaines River
5300 N River Rd
Chicago, IL 60018
United States

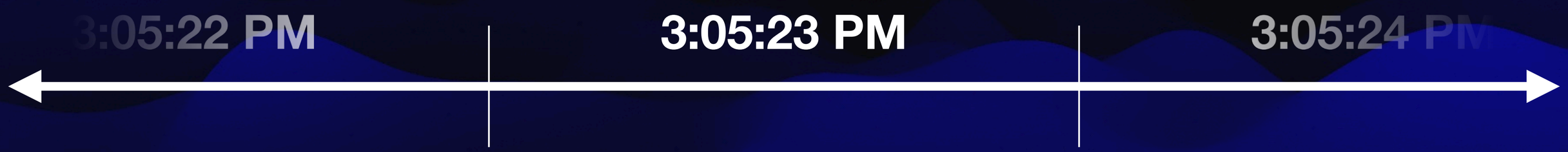

5300 N River Rd
Chicago, IL 60018
United States
TECHNOLOGY BLVD
N WESLEY TERR
N ROSE ST
ROSE ST
W FOSTER AVE
W FOSTER AVE
N RIVER RD
Des Plaines River

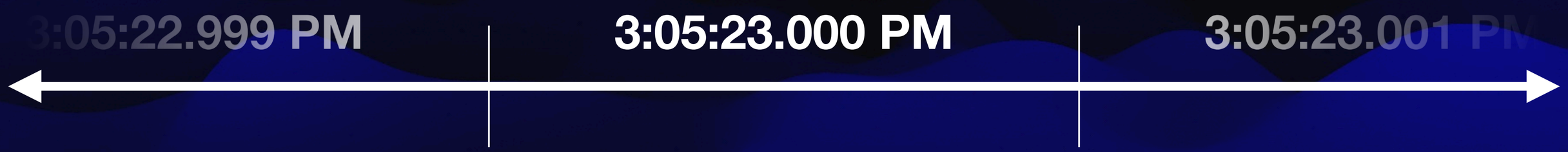
5300 N River Rd
Chicago, IL 60018
United States


May 4th, 2024
May 5th, 2024
May 6th, 2024


May 4th, 2024
May 5th, 2024
May 6th, 2024


May 4th, 2024
May 5th, 2024
May 6th, 2024


3:05:22 PM
3:05:23 PM
3:05:24 PM


3:05:22.999 PM
3:05:23.000 PM
3:05:23.001 PM

Temporal Areas

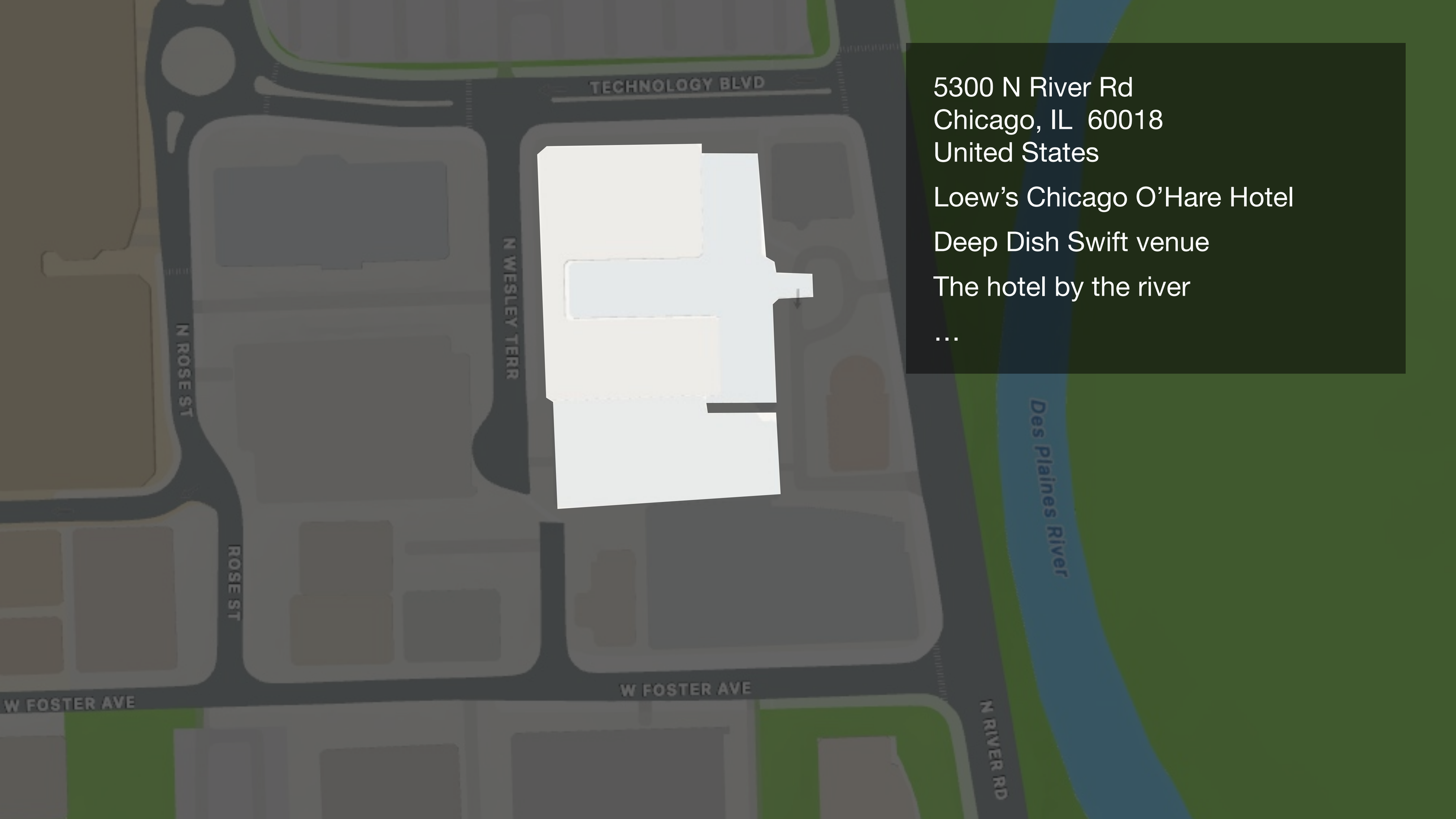
```
let aDate: Date = …
let unit: Calendar.Component = …

let interval = calendar.dateInterval(of: unit, for: aDate)

let startOfPeriod = interval.start
let startOfNextPeriod = interval.end
```

Temporal Areas

```
extension Calendar {
    public func range(of unit: Calendar.Component,
                      containing date: Date) -> Range<Date>?  {

        let interval = self.dateInterval(of: unit, for: date)

        guard let interval else {
            return nil
        }

        return interval.start ..< interval.end
    }
}
```

Space and Time are Infinite


Des Plaines River
N RIVER RD
TECHNOLOGY BLVD
W FOSTER AVE
N WESLEY TERR
ROSE ST
N ROSE ST
W FOSTER AVE



5300 N River Rd
Chicago, IL 60018
United States

Loew’s Chicago O’Hare Hotel

Deep Dish Swift venue

The hotel by the river

…

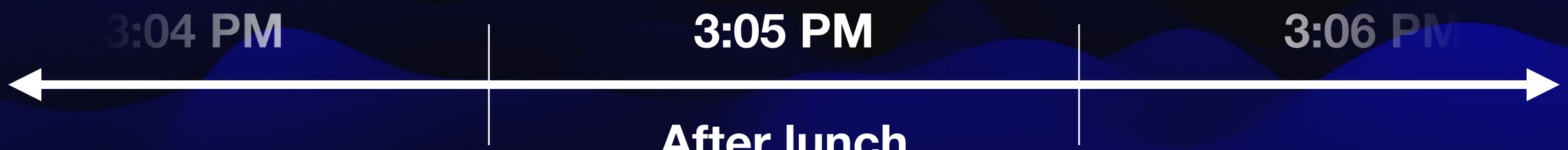

3:04 PM
3:05 PM
3:06 PM

4:05 AM Beijing

4:05 PM Eastern

1:05 PM Pacific

8:05 PM GMT



After lunch

During the conference

...

Infinite Mapping

- Every address contains an *infinite* number of coordinates
- Every coordinate has an *infinite* number of addresses

Practical Time

Asking the right question

Comparing Coordinates

```
let now1 = Date()                                   "May 5, 2024 at 3:05 PM"
let now2 = Date()                                   "May 5, 2024 at 3:05 PM"
print(now1 == now2)                                                   false

print(now1.timeIntervalSince1970)                   1714939500.020886
print(now2.timeIntervalSince1970)                   1714939500.022726
```

Comparing Coordinates

```
let now1 = Date()
let now2 = Date()

let datesAreEquivalent = cal.isDate(now1,
                                    equalTo: now2,
                                    toGranularity: .second)
```

Comparing Coordinates

```
let now1 = Date()
let now2 = Date()

let datesAreEquivalent = cal.isDate(now1,
                                    equalTo: now2,
                                    toGranularity: .month)
```

Comparing Coordinates

```
let now1 = Date()
let now2 = Date()

let datesAreEquivalent = cal.isDate(now1,
                                    equalTo: now2,
                                    toGranularity: .year)
```

Comparing Coordinates

Know your required granularity

- Operators compare raw numeric values
- Provide relative ordering
- Use `Calendar.isDate(_:equalTo:toGranularity:)`
- Comparing `DateComponent` values can usually work…
  - …except when it doesn’t

Arbitrariness

Specific names may not exist

5300 S River Blvd
Los Angeles, MN 12345
Canada

10 March 2024
2:13:42 AM

Arbitrariness

Everything is ~~awesome~~ made up

- Address can *look* right, but still be wrong
- Don’t assume a particular `Calendar`, `Locale`, or `TimeZone`
- Understand setting vs offsetting

Arbitrary Setting

Requiring a particular *named value*

- *Setting* is not safe
  - `Calendar.date(bySetting:value:ofDate:)`
  - `Calendar.date(from:)`
  - `Calendar.enumerateDates(startingAfter:matching:…)`
- Can often return `nil`

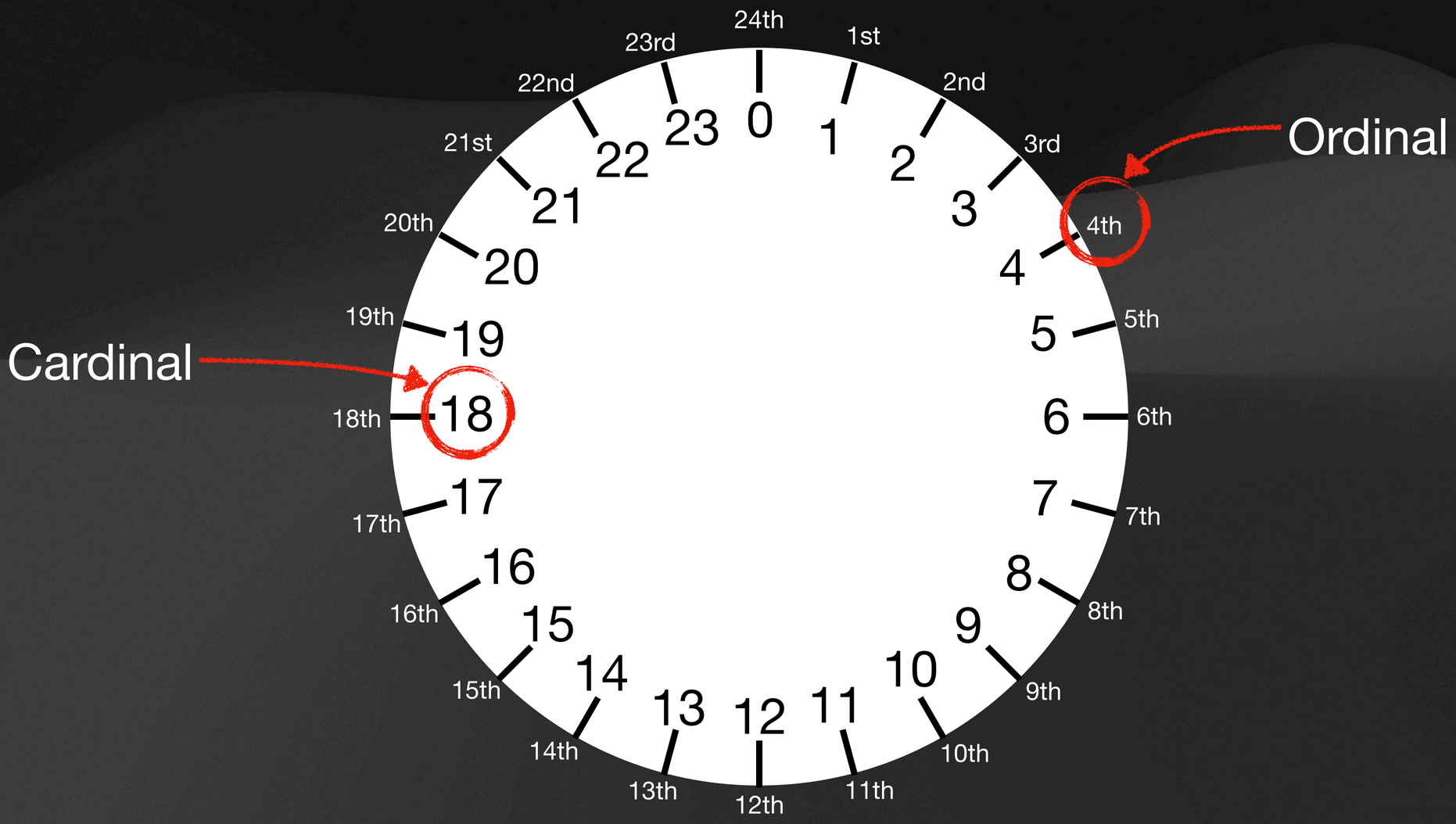
Arbitrary Offsetting

Move by an amount

- *Offsetting* is generally safe
  - `Calendar.date(byAdding:value:toDate:)`
- Rarely returns `nil`
- Can return unmodified values!

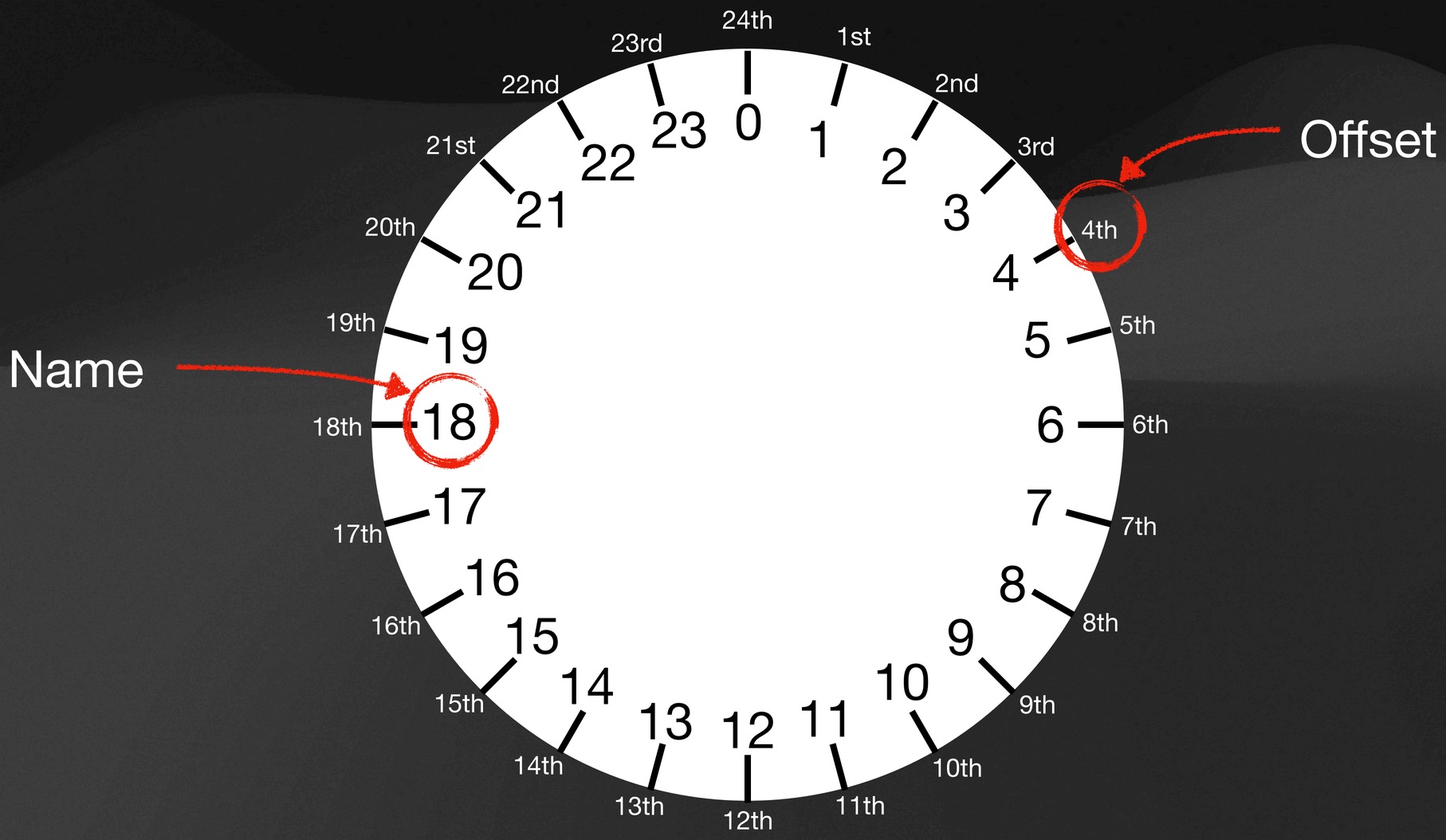
Number vs Position

Cardinals and Ordinals



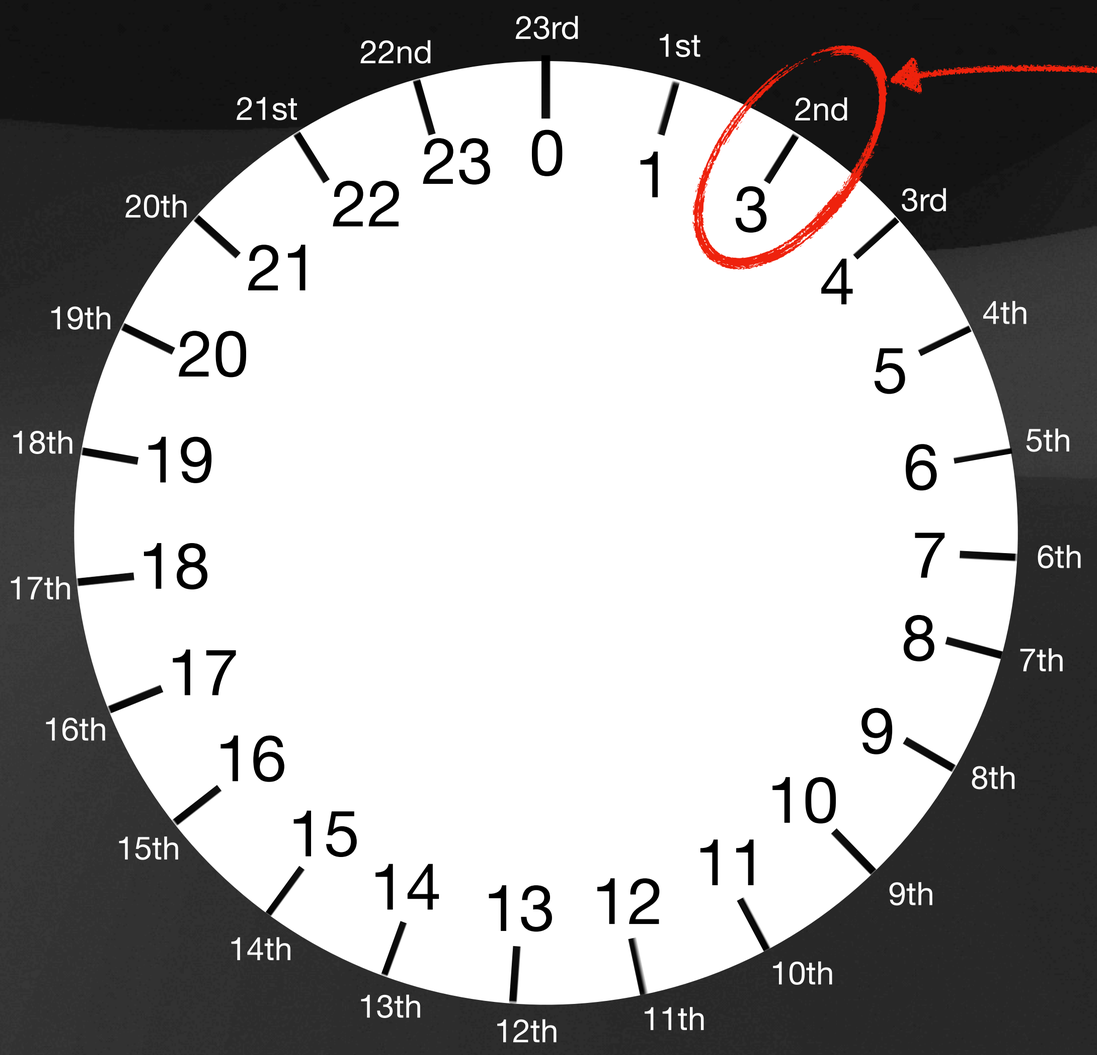
Number vs Position

Cardinals and Ordinals



Number vs Position

Cardinals and Ordinals



Ordinal
≠
Cardinal

Number vs Position

- Number ≠ Position
- Understand the requirement
- The third hour, or the hour named “3”?

Number vs Position

The Third Hour

```
let startOfDay: Date = ...

let startOfThirdHour = aCalendar.date(byAdding: .hour,
                                      value: 2,
                                      to: startOfDay)
```

Number vs Position

The Hour Named “Three”

```
let startOfDay: Date = ...

let startOfThirdHour = aCalendar.date(bySetting: .hour,
                                      value: 3,
                                         of: startOfDay)
```

Number vs Position

The Hour Named “Three”

```
let startOfDay: Date = ...

let match = DateComponents(hour: 3)
let startOfHourThree = aCalendar.nextDate(after: startOfDay,
                                          matching: match,
                                          matchingPolicy: .nextTime)
```

Correctness

86,400 might be *fine*

- How correct do you need to be?
- Will the user care if you’re wrong?
- Expiring a cache → imprecise with leniency
- Showing a timer → must be correct

Be Descriptive

You Deserve Nice Things

- Use very descriptive variable names
- `startOfThirdHour`, `dateRangeOfCurrentMonth`
- `firstInstantWhenCacheInvalidationCanHappen`
- The compiler doesn’t care, but Future You will

Make it Easier

(Shameless self-promotion)

import Time

github.com/davedelong/time

- Typesafe calendrical calculations
- Convenient manipulations and component extractions
- Fully localized
- Common things are easy
- Wrong things are impossible

The End of Time

Time ↔ Space

- Embrace ambiguity and infinity
- Trust your map
- Spatial concepts relate to temporal concepts
- Understanding one translates to the other
- Solutions naturally reveal themselves

Thanks!

davedelong.com

@davedelong@mastodon.social

github.com/davedelong/time