? | Start Day of the Week

? Learning Objectives

- Understand global variations in week start days
- Handle calendar displays for different cultures
- Implement locale-aware week calculations
- Work with ISO week dates and week numbering
- Handle calendar systems beyond Gregorian

? The Week Start Day Challenge

What day does the week start on? The answer depends on where you are in the world! While it might seem like a simple question, different cultures have different conventions, and getting this wrong in a calendar interface can be confusing for users.

Week Start Day by Region

?? Sunday Start

Countries: United States, Canada, Australia, Philippines, Japan, South Korea, Mexico, Brazil, Israel (partially)

Standard: Traditional in Americas and parts of Asia

?? Monday Start

Countries: Most of Europe, China, Russia, India, South Africa, most of Africa, Latin America (some)

Standard: ISO 8601 international standard

?? Saturday Start

Countries: Saudi Arabia, UAE, Egypt, and other Middle Eastern countries

Standard: Common in Islamic calendar contexts

Locale	Country	Week Starts	Weekend Days

`en-US`	United States	**Sunday**	Saturday, Sunday
`en-GB`	United Kingdom	**Monday**	Saturday, Sunday
`de-DE`	Germany	**Monday**	Saturday, Sunday
`ar-SA`	Saudi Arabia	**Saturday**	Friday, Saturday
`he-IL`	Israel	**Sunday**	Friday, Saturday
`ja-JP`	Japan	**Sunday**	Saturday, Sunday
`zh-CN`	China	**Monday**	Saturday, Sunday
`pt-BR`	Brazil	**Sunday**	Saturday, Sunday

? ISO 8601 Standard

The international standard **ISO 8601** defines Monday as the first day of the week. However, many countries (particularly in the Americas and parts of Asia) traditionally use Sunday. Your application should respect the user's locale preference, not enforce a standard.

? Implementation Guidelines

Detecting Week Start Day

JavaScript Example

```
// Using Intl.Locale (modern browsers, Node.js 12+)
const locale = new Intl.Locale('en-US');
const weekInfo = locale.weekInfo || locale.getWeekInfo?.();

console.log(weekInfo?.firstDay);  // ? 7 (Sunday, 1=Monday, 7=Sunday)

// Different locales
const locales = ['en-US', 'en-GB', 'de-DE', 'ar-SA'];
locales.forEach(loc => {
  const l = new Intl.Locale(loc);
  const info = l.weekInfo || l.getWeekInfo?.();
  console.log(`${loc}: Week starts on day ${info?.firstDay}`);
});
// ? en-US: Week starts on day 7 (Sunday)
// ? en-GB: Week starts on day 1 (Monday)
// ? de-DE: Week starts on day 1 (Monday)
// ? ar-SA: Week starts on day 6 (Saturday)

// Fallback for older browsers (manual mapping)
```

```
const weekStartByLocale = {
  'en-US': 0,  // Sunday
  'en-GB': 1,  // Monday
  'de-DE': 1,
  'fr-FR': 1,
  'ar-SA': 6,  // Saturday
  'he-IL': 0,  // Sunday
  'ja-JP': 0,
  'zh-CN': 1
};

function getWeekStartDay(locale) {
  // Try modern API first
  try {
    const l = new Intl.Locale(locale);
    const info = l.weekInfo || l.getWeekInfo?.();
    if (info?.firstDay) {
      // Convert 1-7 (Mon-Sun) to 0-6 (Sun-Sat) for JavaScript Date
      return info.firstDay === 7 ? 0 : info.firstDay;
    }
  } catch (e) {}

  // Fallback to lookup table
  return weekStartByLocale[locale] ?? 0;  // Default to Sunday
}

console.log(getWeekStartDay('en-US'));  // ? 0 (Sunday)
console.log(getWeekStartDay('en-GB'));  // ? 1 (Monday)
```

Python Example

```
from babel import Locale
import calendar

# Using Babel
locale = Locale.parse('en_US')
week_start = locale.first_week_day
print(f"Week starts on day: {week_start}")  # ? 6 (Sunday in 0-6 where 0=Monday)

# Different locales
locales = ['en_US', 'en_GB', 'de_DE', 'ar_SA']
day_names = ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']

for loc_code in locales:
    loc = Locale.parse(loc_code)
    start_day = loc.first_week_day
    print(f"{loc_code}: Week starts on {day_names[start_day]}")

# ? en_US: Week starts on Sun
# ? en_GB: Week starts on Mon
# ? de_DE: Week starts on Mon
# ? ar_SA: Week starts on Sat

# Using Python's calendar module (global setting)
# Set first weekday (0=Monday, 6=Sunday)
calendar.setfirstweekday(calendar.SUNDAY)
print(calendar.firstweekday())  # ? 6
```

```
# Get month calendar with custom first day
calendar.setfirstweekday(calendar.MONDAY)
cal = calendar.monthcalendar(2025, 11)
print(cal)  # Week starts on Monday
```

Building Locale-Aware Calendar Displays

JavaScript Calendar Grid Example

```
function generateCalendarGrid(year, month, locale) {
  const firstDay = new Date(year, month, 1);
  const lastDay = new Date(year, month + 1, 0);

  // Get week start for locale
  const weekStart = getWeekStartDay(locale);

  // Get day of week for first day of month
  let firstDayOfWeek = firstDay.getDay();

  // Adjust for locale's week start
  firstDayOfWeek = (firstDayOfWeek - weekStart + 7) % 7;

  // Build calendar grid
  const grid = [];
  let week = new Array(firstDayOfWeek).fill(null);

  for (let day = 1; day <= lastDay.getDate(); day++) {
    week.push(day);

    if (week.length === 7) {
      grid.push(week);
      week = [];
    }
  }

  // Fill remaining days
  if (week.length > 0) {
    while (week.length < 7) {
      week.push(null);
    }
    grid.push(week);
  }

  return grid;
}

// Generate calendar for US (Sunday start)
const usCalendar = generateCalendarGrid(2025, 10, 'en-US');
console.log('US Calendar (November 2025, Sunday start):');
console.log(usCalendar);

// Generate calendar for UK (Monday start)
const ukCalendar = generateCalendarGrid(2025, 10, 'en-GB');
console.log('UK Calendar (November 2025, Monday start):');
console.log(ukCalendar);
```

```
// Get localized day names for header
function getWeekdayNames(locale, weekStart) {
  const formatter = new Intl.DateTimeFormat(locale, { weekday: 'short' });
  const names = [];

  // Start from locale's first day
  for (let i = 0; i < 7; i++) {
    const day = new Date(2024, 0, weekStart + i); // Jan 2024 starts on Monday
    names.push(formatter.format(day));
  }

  return names;
}

console.log('US weekdays:', getWeekdayNames('en-US', 0));
// ? ['Sun', 'Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat']

console.log('UK weekdays:', getWeekdayNames('en-GB', 1));
// ? ['Mon', 'Tue', 'Wed', 'Thu', 'Fri', 'Sat', 'Sun']
```

? Week Numbering Systems

Different regions also have different conventions for numbering weeks of the year. This affects business reporting, scheduling, and date calculations.

Week Numbering Systems

System	Description	Used In
ISO 8601	Week starts Monday. Week 1 contains first Thursday of year.	Europe, most of world
US System	Week starts Sunday. Week 1 contains January 1st.	United States, Canada
Middle Eastern	Week starts Saturday. Varies by country.	Saudi Arabia, UAE, Egypt

JavaScript ISO Week Example

```
// Calculate ISO week number (week starts Monday, week 1 has first Thursday)
function getISOWeek(date) {
  const target = new Date(date.valueOf());
  const dayNum = (date.getDay() + 6) % 7;
  target.setDate(target.getDate() - dayNum + 3);
  const firstThursday = target.valueOf();
  target.setMonth(0, 1);
```

```
  if (target.getDay() !== 4) {
    target.setMonth(0, 1 + ((4 - target.getDay()) + 7) % 7);
  }
  return 1 + Math.ceil((firstThursday - target) / 604800000);
}

const date1 = new Date(2025, 0, 1);  // January 1, 2025
console.log(`ISO Week: ${getISOWeek(date1)}`);  // ? ISO Week: 1

const date2 = new Date(2025, 10, 5);  // November 5, 2025
console.log(`ISO Week: ${getISOWeek(date2)}`);  // ? ISO Week: 45

// Using Intl for week-year formatting (where supported)
const formatter = new Intl.DateTimeFormat('en-GB', {
  year: 'numeric',
  month: 'long',
  day: 'numeric',
  weekday: 'long'
});

console.log(formatter.format(date2));
// ? "Wednesday, 5 November 2025"
```

? Alternative Calendar Systems

While the Gregorian calendar is most widely used, many cultures use alternative calendar systems for religious, cultural, or official purposes.

Major Calendar Systems

Calendar	Used By	Key Features
Gregorian	Worldwide standard	Solar, 12 months, year 2025
Islamic (Hijri)	Muslim communities	Lunar, 12 months, year 1447 (2025 CE)
Hebrew	Jewish communities	Lunisolar, year 5786 (2025 CE)
Chinese	Chinese, Vietnamese	Lunisolar, 12-13 months, zodiac years
Japanese	Japan (official)	Era-based, Reiwa 7 (2025 CE)
Persian	Iran, Afghanistan	Solar, year 1404 (2025 CE)
Buddhist	Thailand, Sri Lanka	Solar, year 2569 (2025 CE)

JavaScript Calendar System Example

```
// Using Intl.DateTimeFormat with different calendars
const date = new Date(2025, 10, 5);  // November 5, 2025

// Gregorian (default)
const gregorian = new Intl.DateTimeFormat('en-US', {
  year: 'numeric',
  month: 'long',
  day: 'numeric',
  calendar: 'gregory'
}).format(date);
console.log(`Gregorian: ${gregorian}`);
// ? "November 5, 2025"

// Islamic/Hijri calendar
const islamic = new Intl.DateTimeFormat('ar-SA', {
  year: 'numeric',
  month: 'long',
  day: 'numeric',
  calendar: 'islamic'
}).format(date);
console.log(`Islamic: ${islamic}`);
// ? "????? ?????? ?? ????" (approx)

// Hebrew calendar
const hebrew = new Intl.DateTimeFormat('he-IL', {
  year: 'numeric',
  month: 'long',
  day: 'numeric',
  calendar: 'hebrew'
}).format(date);
console.log(`Hebrew: ${hebrew}`);
// ? "?? ??????????? ?????" (approx)

// Japanese calendar (with era)
const japanese = new Intl.DateTimeFormat('ja-JP', {
  year: 'numeric',
  month: 'long',
  day: 'numeric',
  calendar: 'japanese',
  era: 'long'
}).format(date);
console.log(`Japanese: ${japanese}`);
// ? "???11?5?"

// Chinese calendar
const chinese = new Intl.DateTimeFormat('zh-CN', {
  year: 'numeric',
  month: 'long',
  day: 'numeric',
  calendar: 'chinese'
}).format(date);
console.log(`Chinese: ${chinese}`);
// ? Chinese calendar date
```

?? Important Note on Calendar Systems

Always store dates in Gregorian calendar internally (as UTC timestamps or ISO 8601 strings). Alternative calendars should only be used for **display purposes**. Converting between calendar systems for storage can lead to data corruption and synchronization issues.

? Best Practices Checklist

Practice	Priority
☐ Detect and respect user's locale for week start day	**CRITICAL**
☐ Display calendar grids with correct week start	**HIGH**
☐ Store dates in Gregorian/UTC internally, convert for display only	**CRITICAL**
☐ Allow users to override calendar preferences in settings	**MEDIUM**
☐ Be aware of different weekend days (Fri-Sat vs Sat-Sun)	**HIGH**
☐ Use ISO 8601 for week numbering in international contexts	**MEDIUM**
☐ Test calendar displays with Sunday, Monday, and Saturday starts	**HIGH**
☐ Support alternative calendars for display in relevant locales	**MEDIUM**

? Additional Resources

- **ISO 8601:** Date and time format standard (includes week numbering)
- **Unicode CLDR:** Locale-specific calendar data
- **Intl.Locale.weekInfo:** MDN documentation
- **Temporal API (Proposed):** Modern JavaScript date/time handling
- **Babel (Python):** Calendar and locale support
- **ICU (International Components for Unicode):** Comprehensive calendar support

Next Topic: Conclusion & Resources →

Revision #2
Created 5 November 2025 23:00:39 by itsLittleKevin
Updated 6 November 2025 19:35:12