mdn web docs_

# Intl.DateTimeFormat() constructor

The `Intl.DateTimeFormat()` constructor creates `Intl.DateTimeFormat` objects that enable language-sensitive date and time formatting.

## Try it

## Syntax

```
new Intl.DateTimeFormat()
new Intl.DateTimeFormat(locales)
new Intl.DateTimeFormat(locales, options)
```

## Parameters

`locales` Optional

A string with a BCP 47 language tag, or an array of such strings. To use the browser's default locale, pass an empty array. Unicode extension are supported (for example "`en-US-u-ca-buddhist`"). For the general form and interpretation of the `locales` argument, see the Intl page. The following Unicode extension keys are allowed:

`nu`

Numbering system. Possible values include: "`arab`", "`arabext`", "`bali`", "`beng`", "`deva`", "`fullwide`", "`gujr`", "`guru`", "`hanidec`", "`khmr`", "`knda`", "`laoo`", "`latn`", "`limb`", "`mlym`", "`mong`", "`mymr`", "`orya`", "`tamldec`", "`telu`", "`thai`", "`tibt`".

`ca`

Calendar. Possible values include: "`buddhist`", "`chinese`", "`coptic`", "`dangi`", "`ethioaa`", "`ethiopic`", "`gregory`", "`hebrew`", "`indian`", "`islamic`", "`islamic-umalqura`", "`islamic-tbla`", "`islamic-civil`", "`islamic-rgsa`", "`iso8601`", "`japanese`", "`persian`", "`roc`", "`islamicc`".

> **Warning:** The `islamicc` calendar key has been deprecated. Please use `islamic-civil`.

`hc`

Hour cycle. Possible values include: "`h11`", "`h12`", "`h23`", "`h24`".

`options` Optional

An object with some or all of the following properties:

`dateStyle`

The date formatting style to use when calling `format()`. Possible values include:

- "`full`"
- "`long`"
- "`medium`"
- "`short`"

> **Note:** `dateStyle` can be used with `timeStyle`, but not with other options (e.g. `weekday`, `hour`, `month`, etc.).

`timeStyle`

The time formatting style to use when calling `format()`. Possible values include:

- "`full`"
- "`long`"
- "`medium`"
- "`short`"

> **Note:** `timeStyle` can be used with `dateStyle`, but not with other options (e.g. `weekday`, `hour`, `month`, etc.).

`calendar`

Calendar. Possible values include: "`buddhist`", "`chinese`", "`coptic`", "`dangi`", "`ethioaa`", "`ethiopic`", "`gregory`", "`hebrew`", "`indian`", "`islamic`", "`islamic-umalqura`", "`islamic-tbla`", "`islamic-civil`", "`islamic-rgsa`", "`iso8601`", "`japanese`", "`persian`", "`roc`", "`islamicc`".

> **Warning:** The `islamicc` calendar key has been deprecated. Please use `islamic-civil`.

`dayPeriod`

The formatting style used for day periods like "in the morning", "am", "noon", "n" etc. Possible values include: "`narrow`", "`short`", "`long`".

> **Note:**
>
> - This option only has an effect if a 12-hour clock is used.
> - Many locales use the same string irrespective of the width specified.

`numberingSystem`

Numbering System. Possible values include: "`arab`", "`arabext`", "`bali`", "`beng`", "`deva`", "`fullwide`", "`gujr`", "`guru`", "`hanidec`", "`khmr`", "`knda`", "`laoo`", "`latn`", "`limb`", "`mlym`", "`mong`", "`mymr`", "`orya`", "`tamldec`", "`telu`", "`thai`", "`tibt`".

`localeMatcher`

The locale matching algorithm to use. Possible values are "`lookup`" and "`best fit`"; the default is "`best fit`". For information about this option, see the Intl page.

`timeZone`

The time zone to use. The only value implementations must recognize is "`UTC`"; the default is the runtime's default time zone. Implementations may also recognize the time zone names of the IANA time zone database, such as "`Asia/Shanghai`", "`Asia/Kolkata`", "`America/New_York`".

`hour12`

Whether to use 12-hour time (as opposed to 24-hour time). Possible values are `true` and `false`; the default is locale dependent. This option overrides the `hc` language tag and/or the `hourCycle` option in case both are present.

`hourCycle`

The hour cycle to use. Possible values are "`h11`", "`h12`", "`h23`", or "`h24`". This option overrides the `hc` language tag, if both are present, and the `hour12` option takes precedence in case both options have been specified.

`formatMatcher`

The format matching algorithm to use. Possible values are "`basic`" and "`best fit`"; the default is "`best fit`". See the following paragraphs for information about the use of this property.

The following properties describe the date-time components to use in formatted output, and their desired representations. Implementations are required to support at least the following subsets:

- `weekday`, `year`, `month`, `day`, `hour`, `minute`, `second`
- `weekday`, `year`, `month`, `day`
- `year`, `month`, `day`
- `year`, `month`

- `month`, `day`
- `hour`, `minute`, `second`
- `hour`, `minute`

Implementations may support other subsets, and requests will be negotiated against all available subset-representation combinations to find the best match. Two algorithms are available for this negotiation and selected by the `formatMatcher` property: A fully specified "`basic`" algorithm and an implementation-dependent "`best fit`" algorithm.

`weekday`

The representation of the weekday. Possible values are:

- "`long`" (e.g., `Thursday`)
- "`short`" (e.g., `Thu`)
- "`narrow`" (e.g., `T`). Two weekdays may have the same narrow style for some locales (e.g. `Tuesday`'s narrow style is also `T`).

`era`

The representation of the era. Possible values are:

- "`long`" (e.g., `Anno Domini`)
- "`short`" (e.g., `AD`)
- "`narrow`" (e.g., `A`)

`year`

The representation of the year. Possible values are:

- "`numeric`" (e.g., `2012`)
- "`2-digit`" (e.g., `12`)

`month`

The representation of the month. Possible values are:

- "`numeric`" (e.g., `3`)
- "`2-digit`" (e.g., `03`)
- "`long`" (e.g., `March`)
- "`short`" (e.g., `Mar`)
- "`narrow`" (e.g., `M`). Two months may have the same narrow style for some locales (e.g. `May`'s narrow style is also `M`).

`day`

The representation of the day. Possible values are:

- "`numeric`" (e.g., `1`)
- "`2-digit`" (e.g., `01`)

`hour`

The representation of the hour. Possible values are "`numeric`", "`2-digit`".

`minute`

The representation of the minute. Possible values are "`numeric`", "`2-digit`".

`second`

The representation of the second. Possible values are "`numeric`", "`2-digit`".

`fractionalSecondDigits`

The number of digits used to represent fractions of a second (any additional digits are truncated). Possible values are:

- `0` (Fractional part dropped.)
- `1` (Fractional part represented as 1 digit. For example, 736 is formatted as `7`.)
- `2` (Fractional part represented as 2 digits. For example, 736 is formatted as `73`.)
- `3` (Fractional part represented as 3 digits. For example, 736 is formatted as `736`.)

`timeZoneName`

The localized representation of the time zone name. Possible values are:

- "`long`" Long localized form (e.g., `Pacific Standard Time`, `Nordamerikanische Westküsten-Normalzeit`)
- "`short`" Short localized form (e.g.: `PST`, `GMT-8`)
- "`shortOffset`" Short localized GMT format (e.g., `GMT-8`)
- "`longOffset`" Long localized GMT format (e.g., `GMT-0800`)
- "`shortGeneric`" Short generic non-location format (e.g.: `PT`, `Los Angeles Zeit`).
- "`longGeneric`" Long generic non-location format (e.g.: `Pacific Time`, `Nordamerikanische Westküstenzeit`)

> **Note:** Timezone display may fall back to another format if a required string is unavailable. For example, the non-location formats should display the timezone without a specific country/city location like "Pacific Time", but may fall back to a timezone like "Los Angeles Time".

The default value for each date-time component property is `undefined`, but if all component properties are `undefined`, then `year`, `month`, and `day` are assumed to be "`numeric`".

# Examples

## Using DateTimeFormat

In basic use without specifying a locale, `DateTimeFormat` uses the default locale and default options.

```
let date = new Date(Date.UTC(2012, 11, 20, 3, 0, 0));

// toLocaleString without arguments depends on the implementation,
// the default locale, and the default time zone
console.log(new Intl.DateTimeFormat().format(date));
// → "12/19/2012" if run with en-US locale (language) and time zone America/Los_Angeles (UTC-0800)
```

## Using timeStyle and dateStyle

```
let o = new Intl.DateTimeFormat("en" , {
  timeStyle: "short"
});
console.log(o.format(Date.now())); // "13:31 AM"

let o = new Intl.DateTimeFormat("en" , {
  dateStyle: "short"
});
console.log(o.format(Date.now())); // "07/07/20"

let o = new Intl.DateTimeFormat("en" , {
  timeStyle: "medium",
  dateStyle: "short"
});
console.log(o.format(Date.now())); // "07/07/20, 13:31:55 AM"
```

### Using dayPeriod

Use the `dayPeriod` option to output a string for the times of day ("in the morning", "at night", "noon", etc.). Note, that this only works when formatting for a 12 hour clock ( `hourCycle: 'h12'` ) and that for many locales the strings are the same irrespective of the value passed for the `dayPeriod` .

```
let date = Date.UTC(2012, 11, 17, 4, 0, 42);

console.log(new Intl.DateTimeFormat('en-GB', { hour: 'numeric', hourCycle: 'h12',
dayPeriod: 'short', timeZone: 'UTC' }).format(date));
// > 4 at night"  (same formatting in en-GB for all dayPeriod values)

console.log(new Intl.DateTimeFormat('fr', { hour: 'numeric', hourCycle: 'h12',
    dayPeriod: 'narrow', timeZone: 'UTC' }).format(date));
// > "4 mat."  (same output in French for both narrow/short dayPeriod)

console.log(new Intl.DateTimeFormat('fr', { hour: 'numeric', hourCycle: 'h12',
    dayPeriod: 'long', timeZone: 'UTC' }).format(date));
// > "4 du matin"
```

### Using timeZoneName

Use the `timeZoneName` option to output a string for the timezone ("GMT", "Pacific Time", etc.).

```
var date = Date.UTC(2021, 11, 17, 3, 0, 42);
const timezoneNames = ['short', 'long', 'shortOffset', 'longOffset', 'shortGeneric', 'longGeneric']

for (const zoneName of timezoneNames) {
  // Do something with currentValue
  var formatter = new Intl.DateTimeFormat('en-US', {
    timeZone: 'America/Los_Angeles',
    timeZoneName: zoneName,
  });
  console.log(zoneName + ": " + formatter.format(date) );
}

// expected output:
// > "short: 12/16/2021, PST"
// > "long: 12/16/2021, Pacific Standard Time"
// > "shortOffset: 12/16/2021, GMT-8"
// > "longOffset: 12/16/2021, GMT-08:00"
// > "shortGeneric: 12/16/2021, PT"
// > "longGeneric: 12/16/2021, Pacific Time"
```

## Specifications

| Specification |
| --- |

| Specification |
| --- |
| ECMAScript Internationalization API Specification<br># sec-intl-datetimeformat-constructor |

## Browser compatibility

Report problems with this compatibility data on GitHub

| | Chrome | Edge | Firefox | Internet Explorer | Opera | Safari | WebView Android | Chrome Android | Firefox for Android | Opera Android |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| `DateTimeFormat()` constructor | Chrome24 | Edge12 | Firefox29 | Internet Explorer11 | Opera15 | Safari10 | WebView Android4.4 | Chrome Android25 | Firefox for Android56 | Opera Android14 |
| `locales` parameter | Chrome24 | Edge12 | Firefox29 | Internet Explorer11 | Opera15 | Safari10 | WebView Android4.4 | Chrome Android25 | Firefox for Android56 | Opera Android14 |
| `options` parameter | Chrome24 | Edge12 | Firefox29 | Internet Explorer11 | Opera15 | Safari10 | WebView Android4.4 | Chrome Android25 | Firefox for Android56 | Opera Android14 |
| `options.dateStyle` parameter | Chrome76 | Edge79 | Firefox79 | Internet ExplorerNo | Opera63 | Safari14.1 | WebView Android76 | Chrome Android76 | Firefox for Android79 | Opera Android54 |
| `options.dayPeriod` parameter | Chrome92 | Edge92 | Firefox90 | Internet ExplorerNo | Opera78 | Safari14.1 | WebView Android92 | Chrome Android92 | Firefox for Android90 | Opera AndroidNo |
| `options.fractionalSecondDigits` parameter | Chrome84 | Edge84 | Firefox84 | Internet ExplorerNo | Opera70 | Safari14.1 | WebView Android84 | Chrome Android84 | Firefox for Android84 | Opera Android60 |
| `options.hourCycle` parameter | Chrome73 | Edge18 | Firefox58 | Internet ExplorerNo | Opera60 | Safari13 | WebView Android73 | Chrome Android73 | Firefox for Android58 | Opera Android52 |
| `options.timeStyle` parameter | Chrome76 | Edge79 | Firefox79 | Internet ExplorerNo | Opera63 | Safari14.1 | WebView Android76 | Chrome Android76 | Firefox for Android79 | Opera Android54 |
| `options.timeZone` parameter | Chrome24 | Edge12 | Firefox29 | Internet ExplorerNo | Opera15 | Safari10 | WebView Android4.4 | Chrome Android25 | Firefox for Android56 | Opera Android14 |
| IANA time zone names in `options.timeZone` option | Chrome24 | Edge14 | Firefox52 | Internet ExplorerNo | Opera15 | Safari10 | WebView Android37 | Chrome Android25 | Firefox for Android56 | Opera Android14 |
| `options.timeZoneName` parameter | Chrome24 | Edge12 | Firefox29 | Internet ExplorerNo | Opera15 | Safari10 | WebView Android4.4 | Chrome Android25 | Firefox for Android56 | Opera Android14 |
| IANA time zone names in `options.timeZoneName` option | ChromeNo | EdgeNo | Firefox91 | Internet ExplorerNo | OperaNo | Safari15.4 | WebView AndroidNo | Chrome AndroidNo | Firefox for Android91 | Opera AndroidNo |

Full support Partial support No support

## See also

- `Intl.DateTimeFormat`
- `Intl.supportedValuesOf()`
- `Intl`

**Last modified:** Feb 7, 2022, by MDN contributors