

Dates and Times

DGLux5 script stores a point in time in one of two ways:

- as a **number**: milliseconds since the Unix epoch (1970-01-01 00:00:00 UTC). `dateNow()` and `dateParse()` return this.
- as a **DateTime object**, created with `new DateTime(...)`. It gives you the year, month, day, and so on.

To show a date to a user, use `dateFormat()` with a [pattern](#) or an [ICU skeleton](#).

On this page:

- [The DateTime class](#)
- [Date functions](#): `dateNow`, `dateParse`, `dateFormat`, `dateTimeZoneOffset`
- [Date strings that can be parsed](#)
- [Format patterns](#) and [ICU skeletons](#)
- [Examples](#)

See also the [Date Time Operations](#) blocks.

Time zones

`DateTime` objects and date strings without an offset use the time zone of the browser that runs the page. Two people in different time zones can see different results from the same script. To get the same result everywhere, work with UTC: pass `true` as the third argument of `dateFormat()`, or use `toUtc()`.

The DateTime class

Use `DateTime`, not the JavaScript `Date`. `Date` does not exist in DGLux5 script.

In `DateTime`, months start at **1** (January = 1). In JavaScript `Date`, months start at 0.

Creating a DateTime

Constructor	Result
<code>new DateTime()</code>	The current date and time, in local time.
<code>new DateTime(ms)</code>	The date and time <code>ms</code> milliseconds after the epoch, in local time.

Constructor	Result
<code>new DateTime(dateString)</code>	The parsed date string, in local time. Returns null if the string cannot be parsed. See Date strings .
<code>new DateTime(year, month, [day, hour, minute, second, millisecond])</code>	A local date and time built from parts. Missing parts are 0.

When you build a date from parts, values outside the normal range roll over. For example, day 0 is the last day of the previous month, and month 13 is January of the next year. Pass numbers only.

- Missing parts are 0, **including the day**. `new DateTime(2017, 3)` is February 28, 2017, not March 1. Always pass the day.
- With only one argument, the argument is read as milliseconds or as a date string, never as a year.

```
d = new DateTime(2017, 2, 27, 13, 27, 0);
return [d.year, d.month, d.day, d.hour, d.minute, d.second];
```

Output: [2017, 2, 27, 13, 27, 0]

```
// Day 0 of March is the last day of February.
return new DateTime(2024, 3, 0).day;
```

Output: 29

DateTime members

Member	Returns
<code>year</code>	The year, such as 2017.
<code>month</code>	The month, 1-12.
<code>day</code>	The day of the month, 1-31.
<code>hour</code>	The hour, 0-23.
<code>minute</code>	The minute, 0-59.
<code>second</code>	The second, 0-59.
<code>millisecond</code>	The millisecond, 0-999.
<code>weekday</code>	The day of the week. Monday is 1 and Sunday is 7.
<code>millisecondsSinceEpoch</code>	The time as milliseconds since the epoch. This does not depend on the time zone.
<code>isUtc</code>	true if the object is in UTC, false if it is in local time.
<code>timeZoneOffset</code>	The offset from UTC in milliseconds. For example, UTC-5 gives -18000000.
<code>toUtc()</code>	A new <code>DateTime</code> for the same moment, in UTC.
<code>toLocal()</code>	A new <code>DateTime</code> for the same moment, in local time.

Members are read-only. To change a date, build a new one.

```
return new DateTime(2017, 8, 24).weekday;
```

Output: 4 (Thursday)

```
return new DateTime('2017-01-01T13:27:00.798').millisecond;
```

Output: 798

```
u = new DateTime(0).toUtc();  
return [u.isUtc, u.year, u.month, u.day, u.hour];
```

Output: [true, 1970, 1, 1, 0]

Notes

- Arithmetic operators do not work on `DateTime` objects. To find the time between two dates, subtract their `millisecondsSinceEpoch` values.
- If you send a `DateTime` object to the output of a Script block, it is shown as text such as `2017-02-27 13:27:00.000`. Use `dateFormat()` to control the text.

Date functions

dateNow

`dateNow()`

Returns the current time in milliseconds since the epoch. This is the same as `new DateTime().millisecondsSinceEpoch`.

```
start = dateNow();  
// ... do some work ...  
return dateNow() - start; // elapsed milliseconds
```

dateParse

`dateParse(str)`

Parses a [date string](#) and returns milliseconds since the epoch. It accepts the same strings as `new DateTime(str)`. If the string cannot be parsed, the script fails with an error.

```
return dateParse("2002-02-27T19:00:00Z");
```

Output: 1014836400000

```
// The same moment, written with a different offset.  
return dateParse("2002-02-27T14:00:00-0500") ==  
dateParse("2002-02-27T19:00:00Z");
```

Output: true

dateFormat

dateFormat(date, format, [zone])

Returns date as a string, formatted with format.

- date — milliseconds since the epoch, a `DateTime`, or a [date string](#).
- format — a [pattern](#) such as "yyyy-MM-dd HH:mm", or an [ICU skeleton](#) such as "yMMMd".
- zone (optional) — controls the time zone. See the table below.

zone	Result
(omitted)	The date is shown in local time.
true	If date is a number, it is shown in UTC . A <code>DateTime</code> or string is shown as with no zone.
"+hh:mm" or "-hh:mm"	The date is shown in that fixed offset, for example "+05:30". xxx in the format is replaced by the offset.
"auto"	For a string that ends in an offset such as 2020-01-01T10:00:00+05:00, the time is shown as written (10:00), and xxx in the format is replaced by the offset (+05:00).

Returns an empty string if date is not a number, `DateTime`, or string.

```
return dateFormat(0, "yyyy-MM-dd HH:mm:ss", true);
```

Output: 1970-01-01 00:00:00

```
return dateFormat(1500000000000, "yyyy-MM-dd HH:mm xxx", "+05:00");
```

Output: 2017-07-14 07:40 +05:00

```
return dateFormat("2020-01-01T10:00:00+05:00", "HH:mm xxx", "auto");
```

Output: 10:00 +05:00

```
return dateFormat(new DateTime(2017, 8, 24, 15, 38, 0), "EEEE, MMMM d, y 'at'  
h:mm a");
```

Output: Thursday, August 24, 2017 at 3:38 PM

Notes

- Month and day names depend on the locale of the page.
- The first time a script uses `dateFormat`, the block may wait for locale data to load before it runs.

dateTimeZoneOffset

`dateTimeZoneOffset([zone])`

Returns a time-zone offset in milliseconds.

- With no argument: the browser's current offset from UTC.
- With a number: that many hours, for example 5.5.
- With a string "hh:mm" or "-hh:mm": that offset. Other strings return NaN.

```
return [dateTimeZoneOffset(8), dateTimeZoneOffset("-05:30"),  
dateTimeZoneOffset("+05:45")];
```

Output: [28800000, -19800000, 20700000]

Date strings that can be parsed

`new DateTime(str)`, `dateParse(str)`, and `dateFormat(str, ...)` accept these strings. Strings longer than 40 characters are not parsed.

ISO 8601 strings. If the string has Z or an offset, it is converted to local time. Otherwise it is read as local time.

- 2012-02-27
- 2012-02-27 13:27:00
- 2012-02-27 13:27:00.123456z
- 2012-02-27T13:27:00.798
- 20120227 13:27:00
- 20120227T132700
- 20120227
- +20120227
- 2012-02-27T14Z
- 2012-02-27T14+00:00
- 2002-02-27T14:00:00-0500 (the same moment as 2002-02-27T19:00:00Z)

Other formats, always read as local time:

- yyyy-M-d, yyyy/M/d, or M/d/yyyy (US order), for example 2017/8/24 or 8/24/2017
- optionally followed by a time HH:mm:ss or HH:mm:ss.SSS (hours need two digits)
- optionally followed by am or pm, for example 8/24/2017 03:38:00 pm

A number string, such as "15000000000000", is read as milliseconds since the epoch.

Format patterns

A pattern is made of the letters below. Repeat a letter to change the output: for numbers, more letters add leading zeros (m → 6, mm → 06); for text, 3 letters give the short form and 4 give the full form. Any other character, such as :, -, /, or a space, is copied as is. Put other text in single quotes.

Examples are for Thursday, August 24, 2017, 15:38:05.978, in the en_US locale, in time zone UTC-8.

Symbol	Meaning	Example
G	Era	AD
y, yyyy	Year	2017
yy	Year, 2 digits	17
M	Month number	8
MM	Month number, 2 digits	08
MMM	Month, short name	Aug
MMMM	Month, full name	August
L, LL, LLL, LLLL	Standalone month (same as M, but in some languages a different word form)	8, 08, Aug, August
d	Day of month	24
dd	Day of month, 2 digits	24
c	Day of month (standalone)	24
ccc, cccc	Day of week, standalone short/full name	Thu, Thursday
E, EEE	Day of week, short name	Thu
EEEE	Day of week, full name	Thursday
D	Day of year	236
h	Hour, 12-hour clock (1-12)	3
hh	Hour, 12-hour clock, 2 digits	03
H	Hour, 24-hour clock (0-23)	15
HH	Hour, 24-hour clock, 2 digits	15
k	Hour, 24-hour clock (0-23), same as H	15
K	Hour, 12-hour clock (0-11)	3
a	AM/PM marker	PM
m, mm	Minute	38
s, ss	Second	5, 05
S, SSS	Milliseconds (always at least 3 digits)	978

Symbol	Meaning	Example
z	Offset in hours	-8
zz	Offset in hours, 2 digits	-08
zzz	Offset with minutes	-08:00
Z	Offset, RFC 822	-800
ZZ	Offset, RFC 822, 2-digit hour	-0800
v	Time zone name from the browser	PST
Q	Quarter, short	Q3
QQQQ	Quarter, full	3rd quarter
xxx	Replaced by the offset (+05:00) when dateFormat gets a zone argument	+05:00
'text'	Literal text	'Date='
"	A single quote	'o'clock' → o'clock

```
d = new DateTime(2017, 8, 24, 15, 38, 5);
return dateFormat(d, "yyyy-MM-dd HH:mm:ss") + " | " + dateFormat(d, "D") + " | " + dateFormat(d, "QQQQ");
```

Output: 2017-08-24 15:38:05 | 236 | 3rd quarter

ICU skeletons

A skeleton names a set of fields and lets the locale choose the order and punctuation. If the whole format string matches one of the skeletons below, it is used as a skeleton; otherwise it is read as a [pattern](#).

To combine skeletons, separate them with |. Each part is formatted on its own and the results are joined. For example, yMd | |Hms gives 8/24/2017 15:38:00 (the middle part is a space).

Examples are for Thursday, August 24, 2017, 15:38:00, in the en_US locale.

Skeleton	Meaning	Example
d	Day	24
E	Day of week, short	Thu
EEEE	Day of week	Thursday
LLL	Standalone month, short	Aug
LLLL	Standalone month	August
M	Month number	8
Md	Month and day	8/24
MEd	Day of week, month, day	Thu, 8/24
MMM	Month, short	Aug
MMMd	Month and day	Aug 24
MMMEd	Day of week, month, day	Thu, Aug 24

Skeleton	Meaning	Example
MMMM	Month	August
MMMMd	Month and day	August 24
MMMEEEEEd	Day of week, month, day	Thursday, August 24
QQQ	Quarter	Q3
QQQQ	Quarter	3rd quarter
y	Year	2017
yM	Year and month	8/2017
yMd	Year, month, day	8/24/2017
yMEd	Year, month, day, day of week	Thu, 8/24/2017
yMMM	Year and month	Aug 2017
yMMMd	Year, month, day	Aug 24, 2017
yMMMEEd	Year, month, day, day of week	Thu, Aug 24, 2017
yMMMM	Year and month	August 2017
yMMMMd	Year, month, day	August 24, 2017
yMMMEEEEEd	Year, month, day, day of week	Thursday, August 24, 2017
yQQQ	Year and quarter	Q3 2017
yQQQQ	Year and quarter	3rd quarter 2017
H	Hour (24-hour)	15
Hm	Hour and minute (24-hour)	15:38
Hms	Hour, minute, second (24-hour)	15:38:00
j	Hour (locale's clock)	3 PM
jm	Hour and minute (locale's clock)	3:38 PM
jms	Hour, minute, second (locale's clock)	3:38:00 PM
m	Minute	38
ms	Minute and second	38:00
s	Second	0

```
d = new DateTime(2017, 8, 24, 15, 38, 0);
return dateFormat(d, "yMMMd") + " / " + dateFormat(d, "yMd | Hms") + " / " +
dateFormat(d, "jm");
```

Output: Aug 24, 2017 / 8/24/2017 15:38:00 / 3:38 PM

Examples

Number of days in the previous month

```
dt = new DateTime();
lastMonth = new DateTime(dt.year, dt.month, 0);
```

```
return lastMonth.day;
```

The most recent Sunday

weekday is 7 on Sunday, so this returns the previous Sunday when today is a Sunday.

```
dt = new DateTime();  
lastSunday = new DateTime(dt.year, dt.month, dt.day - dt.weekday);  
return dateFormat(lastSunday, "yyyy-MM-dd");
```

Days between two dates

```
ms = dateParse("2024-03-01T00:00:00Z") - dateParse("2024-02-01T00:00:00Z");  
return ms / 86400000;
```

Output: 29

Start of today

```
now = new DateTime();  
start = new DateTime(now.year, now.month, now.day);  
return start.millisecondsSinceEpoch;
```

Duration as hh:mm:ss

Format a number of milliseconds as UTC so that the time zone does not shift it.

```
duration = 3725000; // 1 h 2 min 5 s  
return dateFormat(duration, "HH:mm:ss", true);
```

Output: 01:02:05

From:

<http://wiki.dglogik.com/> - DGLLogik

Permanent link:

http://wiki.dglogik.com/dglux5_wiki:dgscript:datetime

Last update: 2026/10/22 16:32

