DATE

View online: https://www.construct.net/en/make-games/manuals/construct-3/plugin-reference/date

The **Date** object provides a set of conditions and expressions for managing dates and times.

For an example of the Date object, open the Date & Time example.

Scripting

This object has no script interface, because when using JavaScript or TypeScript coding you can use the browser built-in Date object.

Timestamps

The Date object mostly works with *timestamps*. These are the number of milliseconds since January 1st 1970 (also known as the Unix epoch), and is not affected by timezones or leap seconds. This is a standard way to reliably refer to a specific point in time in software development, and is sometimes referred to as the *Unix time*.

A timestamp is just a number, which makes it easy to store them in existing variables or compare them as you would any other number. However it's difficult for humans to make sense of a timestamp since it's just a single large number. The Date object provides expressions to convert between timestamps and human-readable dates and times, and other related tools.

Date & time parts

A timestamp consists of the following parts, all of which are also numbers, with the given ranges and start points as noted:

- **Year:** the full four-digit year, e.g. 2020
- **Month:** the **zero-based** calendar month, e.g. 0 for January, 5 for June
- **Date:** the calendar date, i.e. the day of the month, from 1-31
- **Day:** the zero-based week day from 0-6, starting with Sunday, e.g. 2 for Tuesday
- **Hours:** the hours of the time, 0-23
- **Minutes:** the minutes of the time, 0-59
- **Seconds:** the seconds of the time, 0-59
- **Milliseconds:** the milliseconds of the time, 0-999

Date Conditions

Compare Timestamps

Compare two timestamps. Equal timestamps refer to the same time, and a timestamp that is less than another precedes it in time.

Compare Date Strings

Compare two date strings with each other. The date strings are converted to their equivalent numeric timestamp before the comparison.

Compare Timestamp parts

Compare two timestamp parts with each other. The possible parts are Year (4 digit), Month (0 - 11), Date (1 - 31), Day (0 - 6), Hours (0 - 23), Minutes (0 - 59), Seconds (0 - 59), and Milliseconds (0 - 999).

Compare date string parts

Compare two date string parts with each other, they are converted to their equivalent numeric timestamp before the comparison. The possible parts are Year (4 digit), Month (0 - 11), Date (1 - 31), Day (0 - 6), Hours (0 - 23), Minutes (0 - 59), Seconds (0 - 59), Milliseconds (0 - 999).

Date Expressions

ToString(timeStamp)

Convert a timestamp to a string representation including both date and time.

ToDateString(timeStamp)

Convert a timestamp to a string showing the corresponding date.

ToTimeString(timeStamp)

Convert a timestamp to a string showing the corresponding time.

ToLocaleString(timeStamp)

Convert a timestamp to a localized string representation including both date and time.

ToLocaleDateString(timeStamp)

Convert a timestamp to a localized string showing the corresponding date.

ToLocaleTimeString(timeStamp)

Convert a timestamp to a localized string showing the corresponding time.

ToUTCString(timeStamp)

Convert a timestamp to a string representation including both date and time in universal time.

Parse(dateString)

Parse a date string into the corresponding numeric timestamp. For the supported string formats, refer to the MDN documentation for Date.parse(), which is the underlying method used by this expression.

ToTimerHours(milliseconds)

Convert milliseconds to the equivalent amount of hours as they would be shown in a timer.

ToTimerMinutes(milliseconds)

Convert milliseconds to the equivalent amount of minutes as they would be shown in a timer (0-59).

ToTimerSeconds(milliseconds)

Convert milliseconds to the equivalent amount of seconds as they would be shown in a timer (0-59).

ToTimerMilliseconds(milliseconds)

Convert milliseconds to the equivalent amount of milliseconds as they would be shown in a timer (0-999).

ToTotalHours(milliseconds)

Convert milliseconds to the equivalent amount of hours, which may be a fractional value.

ToTotalMinutes(milliseconds)

Convert milliseconds to the equivalent amount of minutes, which may be a fractional value.

ToTotalSeconds(milliseconds)

Convert milliseconds to the equivalent amount of seconds, which may be a fractional value.

Now

Get the current timestamp (the number of milliseconds since January 1st 1970).

ExportTimestamp

Get the timestamp at the time the project was exported from Construct. In preview mode, this is the time the preview was launched.

Get(year, month, day, hours, minutes, seconds, milliseconds)

Return a timestamp by providing the individual parts of the date and time.

TimezoneOffset

Get the timezone offset of the local system.

GetYear(timeStamp)

Extract the 4 digit year from the provided timestamp in local time.

GetUTCYear(timeStamp)

Extract the 4 digit year from the provided timestamp in universal time.

GetMonth(timeStamp)

Extract the month (0 - 11) from the provided timestamp in local time.

GetUTCMonth(timeStamp)

Extract the month (0 - 11) from the provided timestamp in universal time.

GetDate(timeStamp)

Extract the date (1 - 31) from the provided timestamp in local time.

GetUTCDate(timeStamp)

Extract the date (1 - 31) from the provided timestamp in universal time.

GetDay(timeStamp)

Extract the day (0 - 6) from the provided timestamp in local time.

GetUTCDay(timeStamp)

Extract the day (0 - 6) from the provided timestamp in universal time.

GetHours(timeStamp)

Extract the hours (0 - 23) from the provided timestamp in local time.

GetUTCHours(timeStamp)

Extract the hours (0 - 23) from the provided timestamp in universal time.

GetMinutes(timeStamp)

Extract the minutes (0 - 59) from the provided timestamp in local time.

GetUTCMinutes(timeStamp)

Extract the minutes (0 - 59) from the provided timestamp in universal time.

GetSeconds(timeStamp)

Extract the seconds (0 - 59) from the provided timestamp in local time.

GetUTCSeconds(timeStamp)

Extract the seconds (0 - 59) from the provided timestamp in universal time.

GetUTCMilliseconds(timeStamp)

Extract the milliseconds (0 - 999) from the provided timestamp in local time.

GetMilliseconds(timeStamp)

Extract the milliseconds (0 - 999) from the provided timestamp in universal time.

Difference(first, second)

Calculate the difference between two timestamps.

ChangeYear(timeStamp, year)

Change the year (4 digit) of the provided timestamp in local time, and return as a new timestamp.

ChangeUTCYear(timeStamp, year)

Change the year (4 digit) of the provided timestamp in universal, time and return as a new timestamp.

ChangeMonth(timeStamp, month)

Change the month (0 - 11) of the provided timestamp in local time, and return as a new timestamp.

ChangeUTCMonth(timeStamp, month)

Change the month (0 - 11) of the provided timestamp in universal, time and return as a new timestamp.

ChangeDate(timeStamp, date)

Change the date (1 - 31) of the provided timestamp in local time, and return as a new timestamp.

ChangeUTCDate(timeStamp, date)

Change the date (1 - 31) of the provided timestamp in universal time, and return as a new timestamp.

ChangeDay(timeStamp, day)

Change the day (0 - 6) of the provided timestamp in local time, and return as a new timestamp.

ChangeUTCDay(timeStamp, day)

Change the day (0 - 6) of the provided timestamp in universal time, and return as a new timestamp.

ChangeHours(timeStamp, hours)

Change the hours (0 - 23) of the provided timestamp in local time, and return as a new timestamp.

ChangeUTCHours(timeStamp, hours)

Change the hours (0 - 23) of the provided timestamp in universal time, and return as a new timestamp.

ChangeMinutes(timeStamp, minutes)

Change the minutes (0 - 59) of the provided timestamp in local time, and return as a new timestamp.

ChangeUTCMinutes(timeStamp, minutes)

Change the minutes (0 - 59) of the provided timestamp in universal time, and return as a new timestamp.

ChangeSeconds(timeStamp, seconds)

Change the seconds (0 - 59) of the provided timestamp in local time, and return as a new timestamp.

ChangeUTCSeconds(timeStamp, seconds)

Change the seconds (0 - 59) of the provided timestamp in universal time, and return as a new timestamp.

ChangeMilliseconds(timeStamp, milliseconds)

Change the milliseconds (0 - 999) of the provided timestamp in local time, and return as a new timestamp.

ChangeUTCMilliseconds(timeStamp, milliseconds)

Change the milliseconds (0 - 999) of the provided timestamp in universal time, and return as a new timestamp.

FormatDateWithStyles(locale, timeStamp, dateStyle, timeStyle, hourFormat)

Format the provided time stamp using the locale and optional styles. **"dateStyle"** can be any of *"full"*, *"long"*, *"medium"* or *"short"* and affects the date section of the final result, if an unsupported value is used, the date section will be omitted from the final output of the expression. **"timeStyle"** can be any of *"full"*, *"long"*, *"medium"* or *"short"* and affects the time section of the final result, if an unsupported value is used, the time section will be omitted from the final output of the expression. **"hourFormat"** can be either *"12"* or *"24"* and affects the formatting of the time section of the final result, if an unsupported value is used, the end result will be locale dependant.

FormatDateWithComponents(locale, timeStamp, weekday, year, month, day, hour, minute, second, hourFormat)

Format the provided time stamp using the locale and optional components. **"weekday"** can be any of *"long"*, *"short"* or *"narrow"*. **"year"** can be any of *"numeric"* or *"2-digit"*. **"month"** can be any of *"numeric"*, *"2-digit"*, *"long"*, *"short"* or *"narrow"*. **"day"** can be any of *"numeric"* or *"2-digit"*. **"hour"** can be any of *"numeric"* or *"2-digit"*. **"minute"** can be any of *"numeric"* or *"2-digit"*. **"second"** can be any of *"numeric"* or *"2-digit"*. **"hourFormat"** can be either *"12"* or *"24"* and affects the formatting of the time section of the final result, if an unsupported value is used the end result will be locale dependant.
Any of the components can be omitted from the final result by providing an unsupported value.