Date.parse()

The **`Date.parse()`** method parses a string representation of a date, and returns the number of milliseconds since January 1, 1970, 00:00:00 UTC or `NaN` if the string is unrecognised or contains illegal date values (e.g. 2015-02-31).

Syntax

Direct call:

```
Date.parse(dateString)
```

Implicit call:

```
new Date(dateString)
```

Parameters

`dateString`

A string representing an RFC2822 or ISO 8601 date (other formats may be used, but results may be unexpected).

Description

The `parse()` method takes a date string (such as `"Dec 25, 1995"`) and returns the number of milliseconds since January 1, 1970, 00:00:00 UTC. This function is useful for setting date values based on string values, for example in conjunction with the `setTime()` method and the `Date` object.

Given a string representing a time, `parse()` returns the time value. It accepts the RFC2822 / IETF date syntax (RFC2822 Section 3.3), e.g. `"Mon, 25 Dec 1995 13:30:00 GMT"`. It understands the continental US time zone abbreviations, but for general use, use a time zone offset, for example, `"Mon, 25 Dec 1995 13:30:00 +0430"` (4 hours, 30 minutes east of the Greenwich meridian). If a time zone is not specified and the string is in an ISO format recognized by ES5, UTC is assumed. However, in ECMAScript 2015 ISO format dates without a timezone are treated as local.

GMT and UTC are considered equivalent. The local time zone is used to interpret arguments in RFC2822 Section 3.3 format (or any format not recognized as ISO 8601 in ES5 or ISO 8601 formats in ECMAScript 2015) that do not contain time zone information.

Because of the variances in parsing of date strings, it is recommended to always manually parse strings as results are inconsistent, especially across different ECMAScript implementations where strings like `"2015-10-12 12:00:00"` may be parsed to as `NaN`, UTC or local timezone.

ECMAScript 5 ISO-8601 format support

The date time string may be in ISO 8601 format. For example, `"2011-10-10"` (just date) or `"2011-10-10T14:48:00"` (date and time) can be passed and parsed. The UTC time zone is used to interpret arguments in ISO 8601 format that do not contain time zone information (note that ECMAScript 2015 specifies that date time strings without a time zone are to be treated as local, not UTC).

While time zone specifiers are used during date string parsing to interpret the argument, the value returned is always the number of milliseconds between January 1, 1970 00:00:00 UTC and the point in time represented by the argument or `NaN`.

Because `parse()` is a static method of `Date`, it is called as `Date.parse()` rather than as a method of a `Date` instance.

Differences in assumed time zone

Given a date string of `"March 7, 2014"`, `parse()` assumes a local time zone, but given an ISO format such as `"2014-03-07"` it will assume a time zone of UTC for ES5 or local for ECMAScript 2015. Therefore `Date` objects produced using those strings may represent different moments in time depending on the version of ECMAScript supported unless the system is set with a local time zone of UTC. This means that two date strings that appear equivalent may result in two different values depending on the format of the string that is being converted (this behavior is changed in ECMAScript ed 6 so that both will be treated as local).

Fall-back to implementation-specific date formats

The ECMAScript specification states: If the String does not conform to the standard format the function may fall back to any implementation–specific heuristics or implementation–specific parsing algorithm. Unrecognizable strings or dates containing illegal element values in ISO formatted strings shall cause `Date.parse()` to return `NaN`.

However, invalid values in date strings not recognized as ISO format as defined by ECMA-262 may or may not result in `NaN`, depending on the browser and values provided, e.g.:

```
1 // Non-ISO string with invalid date values
2 new Date('23/25/2014');
```

will be treated as a local date of 25 November, 2015 in Firefox 30 and an invalid date in Safari 7. However, if the string is recognized as an ISO format string and it contains invalid values, it will return `NaN` in all browsers compliant with ES5 and later:

```
// ISO string with invalid values
new Date('2014-25-23').toISOString();
// returns "RangeError: invalid date" in all es5 compliant browsers
```

SpiderMonkey's implementation-specific heuristic can be found in `jsdate.cpp`. The string `"10 06 2014"` is an example of a non–conforming ISO format and thus falls back to a custom routine. See also this rough outline on how the parsing works.

```
new Date('10 06 2014');
```

will be treated as a local date of 6 October, 2014 and not 10 June, 2014. Other examples:

```
new Date('foo-bar 2014').toString();
// returns: "Invalid Date"

Date.parse('foo-bar 2014');
// returns: NaN
```

Examples

Using `Date.parse()`

If `IPOdate` is an existing `Date` object, it can be set to August 9, 1995 (local time) as follows:

```
IPOdate.setTime(Date.parse('Aug 9, 1995'));
```

Some other examples of parsing non–standard date strings:

```
Date.parse('Aug 9, 1995');
```

Returns `807937200000` in time zone GMT-0300, and other values in other time zones, since the string does not specify a time zone and is not ISO format, therefore the time zone defaults to local.

```
Date.parse('Wed, 09 Aug 1995 00:00:00 GMT');
```

Returns `807926400000` no matter the local time zone as GMT (UTC) is provided.

```
Date.parse('Wed, 09 Aug 1995 00:00:00');
```

Returns `807937200000` in time zone GMT-0300, and other values in other time zones, since there is no time zone specifier in the argument and it is not ISO format, so is treated as local.

```
Date.parse('Thu, 01 Jan 1970 00:00:00 GMT');
```

Returns `0` no matter the local time zone as a time zone GMT (UTC) is provided.

```
Date.parse('Thu, 01 Jan 1970 00:00:00');
```

Returns `14400000` in time zone GMT-0400, and other values in other time zones, since no time zone is provided and the string is not in ISO format, therefore the local time zone is used.

```
Date.parse('Thu, 01 Jan 1970 00:00:00 GMT-0400');
```

Returns `14400000` no matter the local time zone as a time zone GMT (UTC) is provided.

Specifications

Specification	Status	Comment
ECMAScript 1st Edition (ECMA-262)	ST Standard	Initial definition. Implemented in JavaScript 1.0.
ECMAScript 5.1 (ECMA-262) The definition of 'Date.parse' in that specification.	ST Standard	ISO 8601 format added.
ECMAScript 2015 (6th Edition, ECMA-262) The definition of 'Date.parse' in that specification.	ST Standard	

Browser compatibility

Desktop | Mobile

Feature	Chrome	Firefox (Gecko)	Internet Explorer	Opera	Safari
Basic support	(Yes)	(Yes)	(Yes)	(Yes)	(Yes)
ISO 8601 format	(Yes)	4.0 (2.0)	9	(Yes)	(Yes)

See also

- `Date.UTC()`