My favorites ▼ | Sign in

datejs

A JavaScript Date Library

Search projects

Project Home Downloads Wiki Issues Source

Search Current pages for Search

APIDocumentation

Date class and instance methods provided by the Datejs library.

Featured

Updated Jul 28, 2010 by geoff%co...@gtempaccount.com

Public Static Methods

---

today

Date.today () : `Date`

Gets a date that is set to the current date. The time is set to the start of the day (00:00 or 12:00 AM).

Parameters

None

Return Value

{`Date`} The current date.

Example

```
Date.today(); // Equivalent to new Date().clearTime()
```

.

compare

Date.compare (`Date` date1, `Date` date2) : `Number`

Compares the first date to the second date and returns an number indication of their relative values. -1 = this is lessthan date. 0 = values are equal. 1 = this is greaterthan date.

Parameters

- {`Date` date1} First Date object to compare. `required`
- {`Date` date2} Second Date object to compare to. `required`

Return Value

{`Number`} -1 = date1 is *lessthan* date2. 0 = values are *equal*. 1 = date1 is *greaterthan* date2.

Example

```
var today = Date.today();
var past = Date.today().add(-6).days();
var future = Date.today().add(6).days();


Date.compare(today, future);                    // -1
Date.compare(today, new Date().clearTime());    // 0
Date.compare(today, past)                       // 1
```

See Also

- between
- compareTo
- equals
- isAfter
- isBefore

.

equals

Date.equals (`Date` date1, `Date` date2) : `Boolean`

Compares the first Date object to the second Date object and returns true if they are equal.

Parameters

- {`Date` date1} First Date object to compare. `required`
- {`Date` date2} Second Date object to compare to `required`

Return Value

{`Boolean`} true if dates are equal. false if they are not equal.

Example

```
Date.equals(Date.today(), new Date().clearTime());   // true

var d1 = Date.today();
var d2 = Date.today().addHours(5);
var d3 = new Date().clearTime();
var d4 = "foo";

Date.equals(d1, d2);  // false
Date.equals(d1, d3);  // true
Date.equals(d1, d4);  // throws exception
```

See Also

- between
- compareTo
- equals
- isAfter
- isBefore

.

getDayNumberFromName

Date.getDayNumberFromName (`String` dayName) : `Number`

Gets the day number (0-6) if given a culture-specific string which is a valid full or abbreviated day name.

Parameters

- {`String` dayName} The name of the day (eg. "Monday, "Mon", "tuesday", "tue", "We", "we").

Return Value

{`Number`} The day number

Example

```
Date.getDayNumberFromName('Tuesday'); // 2
Date.getDayNumberFromName('sat');     // 6
Date.getDayNumberFromName('SUNDAY');     // 0
```

.

getMonthNumberFromName

Date.getMonthNumberFromName (`String` monthName) : `Number`

Gets the month number (0-11) if given a culture-specific string which is a valid full or abbreviated month name.

Parameters

- {`String` monthName} The name of the month (eg. "February, "Feb", "october", "oct").

Return Value

{`Number`} The month number

Example

```
Date.getMonthNumberFromName('January'); // 0
Date.getMonthNumberFromName('feb');  // 1
Date.getMonthNumberFromName('July'); // 6
Date.getMonthNumberFromName('DECEMBER'); // 11
```

.

isLeapYear

Date.isLeapYear (`Number` year) : `Boolean`

Returns true if the given year is a leap year, false otherwise.

Parameters

- {`Number` year} The year.

Return Value

{`Boolean`} true if date is within a LeapYear, otherwise false.

Example

```
Date.isLeapYear(Date.today().getFullYear()); // true|false
var date = new Date(2008,1,29);
Date.isLeapYear(date.getFullYear);           // true
```

.

getDaysInMonth

Date.getDaysInMonth (`Number` year, `Number` month) : `Number`

Gets the number of days in the month, given a year and month value. Automatically corrects for leap year.

Parameters

- {`Number` year} The year.
- {`Number` month} The month (0-11).

Return Value

{`Number`} The number of days in the month.

Example

```
Date.getDaysInMonth(2008, 1);  // 29
Date.getDaysInMonth(2007, 10); // 30
```

.

getTimezoneAbbreviation

Date.getTimezoneAbbreviation (`Number` timezoneOffset, `Boolean` isDayLightSavingsTime) : `String`

Returns a culture-specific timezone abbreviation based on a given offset and a boolean indicating whether daylight savings time is in effect.

Parameters

- {`Number` timezoneOffset} The year.
- {`Boolean` isDayLightSavingsTime} Whether the date instance is observing Daylight Saving Time (Summer Time)

Return Value

{`String`} The timezone abbreviation

Example

```
var today = Date.today();
Date.getTimezoneAbbreviation(today.getTimezoneOffset, today.isDayLightSavingsTime
()); // "UTC", "GMT", "EST", "PDT", etc.
```

.

getTimezoneOffset

Date.getTimezoneOffset (`String` timezoneAbbreviation, `Boolean` isDayLightSavingsTime) : `Number`

Gets the timezone offset if given a culture-specific string which is a valid full or abbreviated timezone name and a boolean indicating whether daylight savings time is in effect.

Parameters

- {`Number` timezoneAbbreviation} The timezone abbreviation
- {`Boolean` isDayLightSavingsTime} Whether the date instance is observing Daylight Saving Time (Summer Time)

Return Value

{`String`} The timezone abbreviation

Example

```
Date.getTimezoneOffset("PST", true);
```

.

parse

Date.parse (`String` dateString) : `Date`

Converts the specified string value into its JavaScript Date equivalent using culture-specific format information.

Parameters

- {`String` dateString} The string value to convert into a Date object. `required`

Return Value

{`Date`} A Date object or null if the string cannot be converted into a Date.

Example

```
Date.parse("today")
Date.parse("t + 5 d") // today + 5 days
Date.parse("next thursday")
Date.parse("February 20th 1973?)
Date.parse("Thu, 1 July 2004 22:30:00?)
```

.

parseExact

Date.parseExact (`String` dateString, `String` formatStringOrArray) : `Date`

Converts the specified string value into its JavaScript Date equivalent using the specified format (string) or formats (array). The format of the string value must match one of the supplied formats exactly.

Parameters

- {String dateString} The string value to convert into a Date object. required.
- {Object formatStringOrArray} The expected format {String} or an array of expected formats {Array} of the date string. required.

Return Value

{Date} A Date object or null if the string cannot be converted into a Date.

Example

```
Date.parseExact("10/15/2004", "M/d/yyyy");  // The Date of 15-Oct-2004
Date.parse("15-Oct-2004", "d-MMM-yyyy");    // The Date of 15-Oct-2004
Date.parse("2004.10.15", "yyyy.MM.dd");     // The Date of 15-Oct-2004
Date.parseExact("10/15/2004", ["M/d/yyyy", "MMMM d, yyyy"]); // The Date of 15-Oct-2004
```

.

validateDay

Date.validateDay (Number day, Number fullYear, Number monthNumber) : Boolean

Validates the number is within an acceptable range for the days in a month [0-MaxDaysInMonth].

Parameters

- {Number day} The number to check if within range.
- {Number fullYear} The number to check if within range.
- {Number monthNumber} The number to check if within range.

Return Value

{Boolean} true if within range, otherwise false.

Example

```
Date.validateDay(15, 2007, 1);  // true, 15-Feb-2007
Date.validateDay(31, 2007, 10); // false, throws RangeError exception
```

.

validateHour

Date.validateHour (`Number` hour) : `Boolean`

Validates the number is within an acceptable range for hours [0-23]. Returns true if within range, otherwise false.

Parameters

- {`Number` hour} The number to check if within range.

Return Value

{`Boolean`} true if within range, otherwise false.

Example

```
Date.validateHour(15); // true
Date.validateHour(24); // false, throws RangeError exception
```

.

validateMillisecond

Date.validateMillisecond (`Number` milliseconds) : `Boolean`

Validates the number is within an acceptable range for milliseconds [0-999]. Returns true if within range, otherwise false.

Parameters

- {`Number` milliseconds} The number to check if within range.

Return Value

{`Boolean`} true if within range, otherwise false.

Example

```
Date.validateMillisecond(500); // true
```

.

validateMinute

Date.validateMinute (`Number` minutes) : `Boolean`

Validates the number is within an acceptable range for minutes [0-59]. Returns true if within range, otherwise false.

Parameters

- {`Number` minutes} The number to check if within range.

Return Value

{`Boolean`} true if within range, otherwise false.

Example

```
Date.validateMinute(45); // true
Date.validateMinute(60); // false, throws RangeError exception
```

.

validateMonth

Date.validateMonth (`Number` month) : `Boolean`

Validates the number is within an acceptable range for months [0-11].

Parameters

- {`Number` month} The number to check if within range.

Return Value

{`Boolean`} true if within range, otherwise false.

Example

```
Date.validateMonth(0);  // true, "January"
Date.validateMonth(12); // false, throws RangeError exception
```

.

validateSecond

Date.validateSecond (`Number` second) : `Boolean`

Validates the number is within an acceptable range for seconds [0-59]. Returns true if within range, otherwise false.

Parameters

- {`Number` second} The number to check if within range.

Return Value

{`Boolean`} true if within range, otherwise false.

Example

```
Date.validateSecond(15); // true
Date.validateSecond(60); // false, throws RangeError exception
```

.

validateYear

Date.validateYear (`Number` year) : `Boolean`

Validates the number is within an acceptable range for years [0-9999].

Parameters

- {`Number` year} The number to check if within range.

Return Value

{`Boolean`} true if within range, otherwise false.

Example

```
Date.validateYear(2007); // true
```

.

Public Instance Methods

add

.add (`Object` config) : `Date`

Adds (or subtracts) to the value of the year, month, day, hour, minute, second, millisecond of the date instance using given configuration object. Positive and Negative values allowed.

Parameters

- {`Object` config} Configuration object containing attributes (months, days, etc.)

Return Value

{`Date`} this

Example

```
Date.today().add({ days: 5, months: 1 });
new Date().add({ years: -1, hours: -6 });
```

The following details all the options for .add().

Example

```
// returns Jul 26 2009 18:45:30 given today as 11-Jan-2008
Date.today().add({
    milliseconds: 500,
    seconds: 30,
    minutes: 45,
    hours: 18,
    days: 15,
    months: 6,
    years: 1
    });

// as one line
Date.today().add({milliseconds: 500, seconds: 30, minutes: 45, hours: 18, days:
15, months: 6, years: 1});

// as separate config object
var config = {milliseconds: 500, seconds: 30, minutes: 45, hours: 18, days: 15, months:
6, years: 1};
Date.today().add(config);
```

See Also

- set

.

addMilliseconds

.addMilliseconds (`Number` milliseconds) : `Date`

Adds the specified number of milliseconds to this instance.

Parameters

- {`Number` milliseconds} The number of milliseconds to add. The number can be positive or negative. `required`

Return Value

{`Date`} this

Example

```
Date.today().addMilliseconds(500);
Date.today().addMilliseconds(-500);
```

.

addSeconds

.addSeconds (`Number` seconds) : `Date`

Adds the specified number of seconds to this instance given the number of seconds to add. The number can be positive or negative.

Parameters

- {`Seconds`} The number of seconds to add. The number can be positive or negative. `required`

Return Value

{`Date`} this

Example

```
Date.today().addSeconds(30);
Date.today().addSeconds(-30);
```

.

addMinutes

.addMinutes (`Number` minutes) : `Date`

Adds the specified number of minutes to this instance given the number of minutes to add. The number can be positive or negative.

Parameters

- {`Number minutes`} The number of seconds to add. The number can be positive or negative. `required`

Return Value

{`Date`} this

Example

```
Date.today().addMinutes(45);
Date.today().addMinutes(-45);
```

.

addHours

.addHours (`Number` hours) : `Date`

Adds the specified number of hours to this instance given the number of hours to add. The number can be positive or negative.

Parameters

- {`Number` hours} The number of hours to add. The number can be positive or negative. `required`

Return Value

{`Date`} this

Example

```
Date.today().addHours(6);
Date.today().addHours(-6);
```

.

addDays

.addDays (`Number` days) : `Date`

Adds the specified number of days to this instance. The number can be positive or negative.

Parameters

- {`Number` days} The number of days to add. The number can be positive or negative. `required`

Return Value

{`Date`} this

Example

```
Date.today().addDays(1);
Date.today().addDays(-1);
```

.

addWeeks

.addWeeks (Number weeks) : Date

Adds the specified number of weeks to this instance given the number of weeks to add. The number can be positive or negative.

Parameters

- {`Number` weeks} The number of weeks to add. The number can be positive or negative. `required`

Return Value

{`Date`} this

Example

```
Date.today().addWeeks(1);
Date.today().addWeeks(-1);
```

.

addMonths

.addMonths (Number months) : Date

Adds the specified number of months to this instance given the number of months to add. The number can be positive or negative.

Parameters

- {`Number` months} The number of months to add. The number can be positive or negative. `required`

Return Value

{Date} this

Example

```
Date.today().addMonths(6);
Date.today().addMonths(-6);
```

.

addYears

.addYears (Number years) : Date

Adds the specified number of years to this instance given the number of years to add. The number can be positive or negative.

Parameters

- {Number years} The number of years to add. The number can be positive or negative. `required`

Return Value

{Date} this

Example

```
Date.today().addYears(10);
Date.today().addYears(-10);
```

.

clearTime

.clearTime ( ) : Date

Resets the time of this Date object to 12:00 AM (00:00), which is the start of the day.

Parameters

None

Return Value

{`Date`} this

Example

```
new Date().clearTime(); // same as Date.today()
```

.

setTimeToNow

.setTimeToNow ( ) : `Date`

Resets the time of this Date object to the current time ('now').

Parameters

None

Return Value

{`Date`} this

Example

```
Date.today().setTimeToNow();
```

.

clone

.clone ( ) : `Date`

Returns a new Date object that is an exact date and time copy of the original instance.

Parameters

None

Return Value

{`Date`} A new Date instance

Example

```
// Wrong way
var d1 = new Date(2007, 0, 1); // 1-Jan-2007
var d2 = d1;
d2.add(6).days();

console.log(d1); // 7-Jan-2007
console.log(d2); // 7-Jan-2007

// Correct way
var d1 = new Date(2007, 0, 1); // 1-Jan-2007
var d2 = d1.clone();
d2.add(6).days();

console.log(d1); // 1-Jan-2007
console.log(d2); // 7-Jan-2007
```

.

between

.between (`Date` startDate, `Date` endDate) : `Boolean`

Determines if this instance is between a range of two dates or equal to either the start or end dates.

Parameters

- {`Date` startDate} Start of range. `required`
- {`Date` endDate} End of range. `required`

Return Value

{`Boolean`} true is this is between or equal to the start and end dates, else false

Example

```
var past = new Date(2000, 4, 5);
var future = new Date(2010, 11, 25)
Date.today().between(past, future); // true|false
```

See Also

- compareTo
- equals
- isAfter
- isBefore

.

compareTo

.compareTo (`Date` date) : `Number`

Compares this instance to a Date object and returns an number indication of their relative values. -1 = this is lessthan date. 0 = values are equal. 1 = this is greaterthan date.

Parameters

- {`Date` date} Date object to compare. `required`

Return Value

{`Number`} -1 = this is lessthan date. 0 = values are equal. 1 = this is greaterthan date.

Example

```
var past = Date.today().add(-6).days();
var future = Date.today().add(6).days();

Date.today().compareTo(future);                 // -1
Date.today().compareTo(new Date().clearTime()); // 0
Date.today().compareTo(past);                   // 1
```

See Also

- between
- equals
- isAfter
- isBefore

.

equals

.equals (`Date` date) : `Boolean`

Compares this instance to another Date object and returns true if they are equal, otherwise false.

Parameters

- {`Date` date} Date object to compare. `required`

Return Value

{`Boolean`} true if dates are equal. false if they are not equal.

Example

```
Date.today().compareTo(new Date().clearTime()); // true

var d1 = Date.today();
var d2 = Date.today().addHours(5);
d1.equals(d2); // false
```

See Also

- between
- compareTo
- isAfter
- isBefore

.

isAfter

.isAfter (`Date` date) : `Boolean`

Determines if this date occurs after the date to compare to.

Parameters

- {`Date` date} Date object to compare. If no date to compare, new Date() ("now") is used.

Return Value

{`Boolean`} true if this date instance is greater than the date to compare to (or "now"), otherwise false.

Example

```
var tomorrow = new Date().add(1).day();
Date.today().isAfter(tomorrow); // false
Date.today().isBefore(tomorrow); // true

var yesterday = new Date().add(-1).day();
Date.today().isAfter(yesterday); // true
Date.today().isBefore(yesterday); // false

// No date to compare to...
Date.today().isAfter(); // false
Date.today().isBefore(); // true
```

See Also

- compareTo
- between
- equals
- isBefore

.

isBefore

.isBefore (`Date` date) : `Boolean`

Determines if this date occurs before the date to compare to.

Parameters

- {`Date` date} Date object to compare. If no date to compare, new Date() ("now") is used.

Return Value

{`Boolean`} true if this date instance is greater than the date to compare to (or "now"), otherwise false.

Example

```
var tomorrow = new Date().add(1).day();
Date.today().isAfter(tomorrow); // false
Date.today().isBefore(tomorrow); // true

var yesterday = new Date().add(-1).day();
Date.today().isAfter(yesterday); // true
Date.today().isBefore(yesterday); // false

// No date to compare to...
Date.today().isAfter(); // false
Date.today().isBefore(); // true
```

See Also

- compareTo
- between
- equals
- isAfter

.

getOrdinalNumber

.getOrdinalNumber ( ) : `Number`

Get the Ordinal day (numeric day number) of the year, adjusted for leap year. Returns 1 through 365 (366 in leap years).

Parameters

None

Return Value

{`Number`} 1 through 365 (366 in leap years)

Example

```
Date.today().getDayOfYear();         // 323
new Date(2000, 0, 1).getDayOfYear(); // 1
```

.

getTimezone

.getTimezone ( ) : `String`

Get the timezone abbreviation of the current date.

Parameters

None

Return Value

{`String`} The abbreviated time zone name (e.g. "EST")

Example

```
Date.today().getTimezone();
```

.

getUTCOffset

.getUTCOffset ( ) : UTCOffset

Get the offset from UTC of the current date. Returns the 4-character offset string prefixed with + or - (e.g. "-0500").

Parameters

None

Return Value

{`String`} The 4-character offset string prefixed with + or - (e.g. "-0500")

Example

```
Date.today().getUTCOffset(); // "-0600"
```

.

getWeek

.getWeek ( ) : `Number`

Get the week number. Week one (1) is the week which contains the first Thursday of the year. Monday is considered the first day of the week.

The .getWeek() function does NOT convert the date to UTC. The local datetime is used. Please use .getISOWeek () to get the week of the UTC converted date.

Parameters

None

Return Value

{`Number`} 1 to 53

Example

```
Date.today().getWeek(); // 7
```

See Also

- getISOWeek
- setWeek

.

getISOWeek

.getISOWeek ( ) : `String`

Get the ISO 8601 week number. Week one ("01") is the week which contains the first Thursday of the year. Monday is considered the first day of the week.

The .getISOWeek() function does convert the date to it's UTC value. Please use .getWeek() to get the week of the local date.

Parameters

None

Return Value

{`String`} "01" to "53"

Example

```
Date.today().getISOWeek(); // "07"
```

See Also

- getWeek
- setWeek
- toISOString

.

setWeek

.setWeek (`Number week`) : `Date`

Moves the date to Monday of the week set. Week one (1) is the week which contains the first Thursday of the year.

Parameters

- {`Number` week} A Number (1 to 53) that represents the week of the year.

Return Value

{`Date`} this

Example

```
Date.today().setWeek(7); // Returns a Date set to the Monday of the week specified
```

See Also

- getWeek
- getISOWeek

.

getOrdinalNumber

.getOrdinalNumber ( ) : `Number`

Get the Ordinal day (numeric day number) of the year, adjusted for leap year. Return a number 1 through 365 (366 in leap years).

Parameters

None

Return Value

{`Number`} 1 through 365 (366 in leap years)

Example

```
Date.today().getOrdinalNumber(); // 46
```

.

hasDaylightSavingTime

.hasDaylightSavingTime ( ) : `Boolean`

Indicates whether Daylight Saving Time is observed in the current time zone.

Parameters

None

Return Value

{`Boolean`} true|false

Example

```
Date.today().hasDaylightSavingTime(); // true|false
```

See Also

- isDaylightSavingTime

.

isDaylightSavingTime

.isDaylightSavingTime ( ) : `Boolean`

Indicates whether this Date instance is within the Daylight Saving Time range for the current time zone.

Parameters

None

Return Value

{`Boolean`} true|false

Example

```
Date.today().isDaylightSavingTime(); // true|false
```

See Also

- hasDaylightSavingTime

.

moveToDayOfWeek

.moveToDayOfWeek (`Number` dayOfWeek, `Number` direction) : `Date`

Move to the next or previous dayOfWeek. Whether to move into the future (+1) or past (-1) is controlled by the optional direction parameter.

Parameters

- {`Number` dayOfWeek} The dayOfWeek to move to
- {`Number` direction} Forward (+1) or Back (-1). Defaults to +1. `optional`

Return Value

{`Date`} this

Example

```
Date.today().moveToDayOfWeek(0);     // move to next Sunday
Date.today().moveToDayOfWeek(0, -1); // move to last Sunday
```

.

moveToFirstDayOfMonth

.moveToFirstDayOfMonth ( ) : `Date`

Moves the date to the first day of the month.

Parameters

None

Return Value

{`Date`} this

Example

```
new Date(2007, 10, 19).moveToFirstDayOfMonth(); // 1-Nov-2007
```

.

moveToLastDayOfMonth

.moveToLastDayOfMonth ( ) : `Date`

Moves the date to the last day of the month.

Parameters

None

Return Value

{`Date`} this

Example

```
new Date(2007, 10, 19).moveToLastDayOfMonth(); // 30-Nov-2007
```

.

moveToMonth

.moveToMonth (`Number` month, `Number` direction) : `Date`

Move to the next or previous month. Whether to move into the future (+1) or past (-1) is controlled by the optional direction parameter.

Parameters

- {`Number` month} The month to move to. 0 = January, 11 = December
- {`Number` direction} Forward (+1) or Back (-1). Defaults to +1. `optional`

Return Value

{`Date`} this

Example

```
Date.today().moveToMonth(0);     // move to next January
Date.today().moveToMonth(0, -1); // move to last January
```

.

moveToNthOccurrence

.moveToNthOccurrence (`Number` dayOfWeek, `Number` occurrence) : `Date`

Moves the date to the next n'th occurrence of the dayOfWeek starting from the beginning of the month. The number (-1) is a magic number and will return the last occurrence of the dayOfWeek in the month.

Parameters

- {`Number` dayOfWeek} The dayOfWeek to move to
- {`Number` occurrence} The n'th occurrence to move to. Use (-1) to return the last occurrence in the month

Return Value

{`Date`} this

Example

```
Date.today().moveToNthOccurrence(0, 1);     // First Sunday of the month
Date.today().moveToNthOccurrence(0, 3);     // Third Sunday of the month
Date.today().moveToNthOccurrence(0, -1);    // Last Sunday of the month
```

.

set

.set (`Object` config) : `Date`

Set the value of year, month, day, hour, minute, second, millisecond of date instance using given configuration object.

Parameters

- {`Object` config} Configuration object containing attributes (month, day, etc.)

Return Value

{`Date`} this

Example

```
Date.today().set({ day: 15, hour: 8 }); // Sets the day to the 15th day of the
current month and the hour to 8 (AM).
```

The following list all property options for .set().

Example

```
// returns Jul 15 2008 18:45:30
Date.today().set({
    millisecond: 500,
    second: 30,
    minute: 45,
    hour: 18,
    day: 15,
    month: 6,
    year: 2008
    });

// as one line
Date.today().set({millisecond: 500, second: 30, minute: 45, hour: 18, day: 15, month:
6, year: 2008});

// as separate config object
var config = {millisecond: 500, second: 30, minute: 45, hour: 18, day: 15, month: 6,
year: 2008};
Date.today().set(config);
```

See Also

- add

.

setTimezone

.setTimezone (`String` timezoneAbbreviation) : `Date`

Set the timezone for the current date using a culture-specific timezone abbreviation ("PST"). Note that in most JavaScript implementations, this will appear to change the time since the timezone is always based on the locale.

Parameters

- {`String` timezoneAbbreviation} The timezone abbreviation.

Return Value

{`Date`} this

Example

```
Date.today().setTimezone("PST");
```

.

setTimezoneOffset

.setTimezoneOffset (`Number` timezoneOffset) : `Date`

Set the timezone for the current date using an offset (-0700). Note that in most JavaScript implementations, this will appear to change the time since the timezone is always based on the locale.

Parameters

- {`Number` timezoneOffset} The timezone offset

Return Value

{`Date`} this

Example

```
Date.today().setTimezoneOffset(-0700);
```

.

toISOString

.toISOString ( ) : `String`

Converts the current date instance into a string with an ISO 8601 format. The date is converted to it's UTC value. As per the ISO 8601 specification, the string will be wrapped with double quotation marks (").

Parameters

None

Return Value

{`String`} ISO 8601 string of date

Example

```
new Date().toISOString();  // ""2008-04-13T10:07:15Z""
```

The local time version of ISO 8601 formatted string can be created by passing a custom format to the .toString () function.

Example

```
new Date().toString("yyyy-MM-ddTHH:mm:ssZ");  // "2008-04-13T04:11:05Z"
```

See Also

- toString

.

toString

.toString (`String` format) : `String`

Converts the value of the current Date object to its equivalent string representation. Use format argument to specify format (optional). See FormatSpecifiers for more info.

Parameters

- {`String` format} A format string consisting of one or more format spcifiers. `optional`

Return Value

{`String`} A string representation of the current Date object.

Example

```
Date.today().toString();           // native .toString() functionality
Date.today().toString("M/d/yyyy"); // 11/19/2007
Date.today().toString("d-MMM-yyyy"); // 19-Nov-2007
new Date().toString("HH:mm");      // 18:45
```

See Also

- toISOString

Comment by basstrad...@gmail.com, Nov 27, 2007

comprehensive, very nice

Comment by silou...@gmail.com, Dec 19, 2007

This lib is very helpfull for me.Thanks for the work.

But I think there's a mistake in the documentation of th method "compareTo"

In the version 1.1 alpha the method compareTo return the value

Date.today().compareTo(future); // -1 Date.today().compareTo(new Date().clearTime()); // 0 Date.today().compareTo(past); // 1

Comment by geoff%co...@gtempaccount.com, Dec 19, 2007

@silouane - Thanks for pointing out the error in the documentation. I have updated the sample code.

Comment by DeepakRa...@gmail.com, Dec 21, 2007

Why are the methods is(), final(), first() etc are not documented? I get a function not defined error when I use first(), final() in my code. They are not in the date.js file either.

Comment by geoff%co...@gtempaccount.com, Dec 21, 2007

@DeepakRaghav? - The .is, .final and .first functions as well a several others were added after the Alpha-1 release. To get the latest files you have use SVN (http://code.google.com/p/datejs/source) or browse directly to the files you require (http://datejs.googlecode.com/svn/trunk/).

The /build/ folder in SVN is always up to date with the latest code.

The documentation will be updated shortly.

Comment by geoff%co...@gtempaccount.com, Dec 26, 2007

Hi all, if you find a bug in the code, please post the bug to the "Issues" tab (http://code.google.com/p/datejs/issues/list). Or, you can discuss in the Datejs forums (http://www.datejs.com/forums/).

Thanks for the help.

Comment by mpria...@gmail.com, Jan 11, 2008

Set the value of year, month, day, hour, minute, second, millisecond of date instance using given configuration object.

Can you give a `full` example of the configuration object that shows all possible properties?

Comment by geoff%co...@gtempaccount.com, Jan 11, 2008

@mpriatel - I added a full code sample to both .set and .add.

Basically they both accept all the date parts as parameters of the config object.

.set({day: 1, month: 1, year: 2008})

.add({days: 5, months: 5, years: 5})

Hope this helps.

Comment by mpria...@gmail.com, Jan 23, 2008

perfect!

Comment by nickneck...@gmail.com, Mar 7, 2008

You list Date.compare(Date ,Date) as a public static method, but this method doesn't appear to be part of the download as of 3.7.K8.

Comment by geoff%co...@gtempaccount.com, Mar 7, 2008

@nickneckthepirate - The Date.compare function was added after the Alpha-1 release, but is included with the latest code in SVN (http://www.datejs.com/svn/).

Comment by chrishar...@gmail.com, Mar 9, 2008

I couldn't get getUTCOffset() to work in IE7 - I had to change the line

from: `return r[0]+r.substr(2);` to: `return r.charAt(0)+r.substr(2);`

Comment by david.hu...@gmail.com, Mar 11, 2008

This is great! FYI: there is a typo in the doc. There are 2 addHours(). One should be addDays.

Comment by geoff%co...@gtempaccount.com, Mar 11, 2008

@david.humpherys - Thanks for pointing out the typo. I have corrected the documentation.

Comment by smsh...@gmail.com, Apr 3, 2008

is there a function to find the difference between 2 dates?

Comment by geoff%co...@gtempaccount.com, Apr 3, 2008

@smshuja - Within the /trunk/src/ folder (http://www.datejs.com/svn/) is a file called time.js. The time.js file contains a class called TimeSpan??. The TimeSpan?? object can be used to determine the difference between two dates (days, hours, minutes, seconds, milliseconds).

Example

```
var future = new Date().add({months: 5, days: 4, hours: 3, minutes: 2, seconds: 1}); var now = new Date();
```

```
var span = new TimeSpan??(future - now);
```

```
console.log("Days:", span.getDays()); console.log("Hours:", span.getHours()); console.log("Minutes:", span.getMinutes()); console.log("Seconds:", span.getSeconds());
```

Hope this helps.

Comment by mayank.r...@gmail.com, Apr 6, 2008

Here is a test file I created...

Valid js lines in the comments eg. // Date.friday() and so on give me a javascript errors.

Are there more js files to include besides date.js?

Thanks.

Mayank

```
<html> <head> <title> Date JS Testing </title> <script type=text/javascript src=date.js></script> </head> <body> <script> // Get today.s date alert(Date.today());
```

```
// Add 5 days to today alert(Date.today().add(5).days());
```

```
// Get Friday of this week -- This does not work? Date.friday is not a function //alert(Date.friday());
```

```
// Get March of this year -- Same for this. Does not work. //alert(Date.march());
```

```
// Is today Friday? alert(Date.today().is().friday()); // true|false
// What day is it? alert(Date.today().getDayName());

// Get the last Friday of the year alert(Date.dec().final().fri());

</script>

</body> </html>
```

Comment by geoff%co...@gtempaccount.com, Apr 6, 2008

@mayank.r.jain - The Date.friday() functionality was added after the Alpha-1 release. The latest bits are only available directly from SVN (http://www.datejs.com/svn/).

The latest date.js file can be download directly via the following link...

http://datejs.googlecode.com/svn/trunk/build/date.js

Hope this helps.

Comment by vincent....@gmail.com, Apr 18, 2008

Documentation Error:

Date.getDayNumberFromName (String dayName) : Number (...)

Parameters

- {String dayName} The name of the month (eg. "February, "Feb", "october", "oct").

(...)

Date.getMonthNumberFromName (String monthName) : Number (...)

Parameters

- {String monthName} The name of the day (eg. "Monday, "Mon", "tuesday", "tue", "We", "we").

Comment by geoff%co...@gtempaccount.com, Apr 18, 2008

@vincent.chute - Thanks for pointing out the documentation error with getDayNumberFromName and getMonthNumberFromName. The error has been corrected.

Comment by mpria...@gmail.com, Apr 21, 2008

I've added these two convenience methods:

```
Date.prototype.isAfter = function(d) {

    return this.compareTo(d) > 0;

} Date.prototype.isBefore = function(d) {

    return this.compareTo(d) < 0;

}
```

Not sure if I've re-invented the wheel here, but I find it easier to work with the syntax above, and it makes the code a little more readable.

Comment by geoff%co...@gtempaccount.com, Apr 21, 2008

@mpriatel - You know, I think I've added those two functions about 3 times to the library. I stare at them for while, trying to decide whether the usefulness is worth the added weight. Then I always delete. ... But, not this time. I'm adding those two into the library and I'm not thinking about it again. I certainly agree that they're very useful and do clean up some otherwise awkward syntax.

Comment by Greg.Yac...@gmail.com, May 8, 2008

Why does add() use fields in the plural (like "hours", "minutes", etc.) while set() uses fields in the singular ("hour", "minute", etc.).

I have a utility function that parses a date or time string into an object that is sometimes passed to set() and sometimes passed to add(). To work around this, I end up setting both singular and plural named fields. E.g. timeObj.hours = timeObj.hour = ...;

This is in the latest alpha (or is it beta?) code. I think that the previous release accepted both singular and plural field names. I can see how it might be problematic in the case where the object has both named fields with different values, but would it be possible to just decide on one naming convention?

Comment by geoff%co...@gtempaccount.com, May 9, 2008

@Grreg.Yachuk - Can you post this question in the forums? That would be a better place to discuss and would encourage others to join in. We can link back to the forum post from here. thanks!

Comment by Greg.Yac...@gmail.com, May 9, 2008

Done.

Comment by wath1...@gmail.com, May 13, 2008

Subj: Missing .getElapsed() method.

It is in the docs above but missing from the code. For those like me that did not realise it you can simple do d1 - d2.

Thanks for the package, Gavin

Comment by geoff%co...@gtempaccount.com, May 14, 2008

@wath1000 - thanks for pointing out that .getElapsed was still in the docs. I removed .getElapsed from the library because all it did was d1-d2. The getElapsed documentation has been removed from this page.

Comment by exedesi...@gmail.com, Jul 5, 2008

I am not a programmer at all, but can someone guide me on how to have a counter on a site to count down to a specific date (yyyy-mm-w-dd-hh-mm-ss) please.

Comment by Alain.Fa...@gmail.com, Jul 23, 2008

I now just miss some `.toShortLocaleString()` that would take only the date part, not the time, to the browser's locale format.

Instead of "quarta-feira, 23 de julho de 2008 13:59:26" (standard `.toLocaleString()` method) I would expect "23/07/2008" (here in Brazil) or "07/23/2008" (for the US).

Comment by cesc.sa...@gmail.com, Aug 20, 2008

Spanish Date.parse("20 agosto 2008") do not give spected result, Date.parse("20-agosto-2008") does.

Comment by dwb0...@yahoo.com, Sep 18, 2008

I had reached my point of frustration, and was about to start building my own date library to allow me to mimic the C# datetime syntax in JS, when I stumbled upon this. GREAT WORK! saved me hours upon hours of time. Thanks.

Comment by FlaviusV...@gmail.com, Sep 18, 2008

Where is the "isBefore" function? I've search every file in the full download and it can't find it.

Comment by geoff%co...@gtempaccount.com, Sep 18, 2008

@FlaviusVictor? - Please ensure you download the latest build from SVN (http://www.datejs.com/svn/). The isBefore function was added recently, after the Alpha-1 release. Hope this helps.

Comment by arup.sof...@gmail.com, Dec 4, 2008

Very Nice... Great.

Comment by tonyhs...@gmail.com, Feb 5, 2009

getWeek() doesn't seem to work ...

Looked in date.js - getWeekOfYear() does seem to work. =)

Comment by nicholas...@gmail.com, Mar 10, 2009

Using en-GB and core.js from SVN

typeof(Date.parse("18/04/1981")) // number typeof(Date.today()) // object

Date.today().isBefore() // true Date.parse("18/04/1981").isBefore() // isBefore is not a function

Date.today().isBefore(Date.parse('18/04/1981')) gives //TypeError?("Tue Mar 10 2009 00:00:00 GMT+0000 (GMT Standard Time) - 391993200000")@:0 Firebug?

Comment by NFinst...@gmail.com, Mar 13, 2009

Great Stuff! feature suggestion: Date.parse(dateString,past|future) So if looking for a date in the past, and it is now Monday, "Tuesday" will mean last Tuesday, not tomorrow.

Comment by xphuon...@gmail.com, Mar 16, 2009

how can to convert time.time() from python to javascript's Date. Eg: t= time.time() #t= 1237203330.015857 I want to convert t to Date in javascript. Pls help me. Thanks

Phuong

Comment by jed...@gmail.com, Mar 31, 2009

@nicholas.associates.software same here, but with `isAfter()`

using `date.js` or `date-pt-BR.js`

Comment by jed...@gmail.com, Mar 31, 2009

Using `date-en-US.js` from `svn/trunk/build/` now.

Seems to work... :)

Before I've tried the ones from http://www.datejs.com/download/.

Well, nice job btw! It's a great time saver! :D

Comment by juanef...@gmail.com, Apr 21, 2009

Very nice lib, thanks. BTW toString and parseExact command examples are incorrect. .

Comment by cyrus.da...@gmail.com, Apr 22, 2009

Gr8 lib! Thanks!!!!

Comment by wilson...@gmail.com, Jun 25, 2009

Bug????????????????? Parse doesn't work

var InitialDate?=Date.parse("01/07/2009", "dd/MM/yyyy");

when i alert(InitialDate?);

it shows me "Wed Jan 07 2009 00:00:00 GMT+1300"

it just uses "MM/dd/yyyy" format for me...

Comment by ecoldw...@gmail.com, Jul 20, 2009

There appears to be a problem displaying and parsing 12 o'clock. "12pm" cannot be parsed. "12am" is parsed as 12pm. Midnight is displayed as "0:00:00 AM" instead of "12:00:00 AM"

Comment by wise...@gmail.com, Oct 25, 2009

Yes, I have reproduced ecoldwell's problem

Comment by fares.fa...@gmail.com, Oct 27, 2009

am I doing a mistake here? I'm trying to set the date with something like :

var dateConfig = {minute:35, hour:13, day:24, month:10, year:2009}; dateConfigDate = Date.today().set(dateConfig);

but I'm keep getting the today's day (the value of the day should be 24 as its explicitly set in the dateConfig JSON)..

anyone got the same problem?

Comment by manasag...@gmail.com, Nov 18, 2009

I want to know whether all the functionality is in date.js or it is spread in core.js,timer.js and other js files. Please help.

Comment by anildbes...@gmail.com, Dec 9, 2009

Hi,

Great work, very nice lib.

I have an issue though, i want to get last week from Sunday to Saturday. '1 week ago' always show last Wednesday if today is Wednesday but i want last week from Sunday to Saturday so if today is 12-09-2009 i want to get date 11-29-2009 or say first day of last week.

thanks.

Comment by designbykamal, Jan 14, 2010

(6).months().ago().toString("yyyy-MM-dd") doesn't properly span the new year.

Use Date.parse('-6months').toString('yyyy-MM-dd')) instead.

Comment by riodepa...@gmail.com, Mar 10, 2010

is the code for isBefore() finally found?

Comment by bunnyh...@gmail.com, Mar 14, 2010

a table of contents or index at the top of this page would be very helpful :) thanks.

Comment by ipremra...@gmail.com, Mar 21, 2010

I would like to parse the date time like: if current date time is 12Mar,2010 08:02 PM and I give the date 12Mar,2010 08:00 PM then it should say - about 2 minutes ago ... same like twitter. Is it possible with DateJs? I seen docs but haven't found.

Comment by irm...@gmail.com, Mar 29, 2010

I couldn't make compare method work. Do I missing something because it says 'Date.compare is not a function' when I try to call that method.

Thanks

Comment by rosskarc...@gmail.com, Apr 10, 2010

It'd be handy if there were a "get" complement to moveToNthOccurrence-- say, given a date, return something that indicates it's the third thursday or whatever

Comment by jnthnl...@gmail.com, May 6, 2010

This is really great! The only shortcoming I've found is that there is no function to create a Date object from an ISO string, even though there is a function to output one.

I found this - http://delete.me.uk/2005/03/iso8601.html - which does the job.

Thanks,

J.

Comment by jnthnl...@gmail.com, May 8, 2010

Hello!

I think it would be good if the built version of datejs.js was updated from the version made in 2007 (http://code.google.com/p/datejs/downloads/list) to the one made in 2008 (updated 2008-05-13). The one from 2008 has some functions not in the earlier version, such as Date.compare.

J.

Comment by grubbs...@gmail.com, Jun 4, 2010

It appears that the "12pm" issue still remains. When you attempt to parse "12pm", "12 PM", or any variant of that it pretty much just dies. Any fix for this in place or in the works? // JC

Comment by fluoro@gmail.com, Jun 6, 2010

The timezone setting doesn't seem to work. Right now Date.today().toString() returns "Mon Jun 07 2010 00:00:00 GMT-0500 (CDT)" for me. If I try to do Date.today().setTimezone ("PST").toString() it gives me the same output. Also trying to do setTimezoneOffset(-0500) causes strange results, the minutes appear to change in unpredictable ways rather than hours changing in predictable ways.

Comment by fluoro@gmail.com, Jun 6, 2010

Perhaps related to my previous comment, but Date.parse() seems to be giving the wrong results.

```
var s = "2010/06/10 06:15:00 UTC";
Date.parse(s); // "Thu Jun 10 2010 06:15:00 GMT-0500 (CDT)" (incorrect)
new Date(s);   // "Thu Jun 10 2010 01:15:00 GMT-0500 (CDT)" (correct)
```

Or am I doing something wrong?

Comment by zhaoheng...@gmail.com, Jul 22, 2010

Date.compare ••••!!!

Comment by matshof...@gmail.com, Jul 27, 2010

There is a little error at the parseExact documentation:

```
Date.parse("15-Oct-2004", "M-ddd-yyyy"); // The Date of 15-Oct-2004
```

Should be

```
Date.parse("15-Oct-2004", "d-MMM-yyyy"); // The Date of 15-Oct-2004
```

Comment by justinvi...@gmail.com, Sep 7, 2010

If you want to get/set timestamp you can do something like this..

```
var myDate = Date.parse('10:12pm 01/01/2010');

var timestamp = myDate.getTime();
```

Comment by fluoro@gmail.com, Sep 18, 2010

There is a problem parsing dates without any kind of separators:

```
Date.parse('20100913', 'yyyyMMdd')

    => Fri Oct 20 913 00:00:00 GMT-0500 (CDT)
```

Comment by geoff%co...@gtempaccount.com, Sep 18, 2010

@fluoro - please use .parseExact

```
Date.parseExact('20100913', 'yyyyMMdd');
```

Comment by dilis...@gmail.com, Oct 15, 2010

@ge...@coolite.com -- Thanks !! that worked !

Comment by paulrajj, Oct 30, 2010

Hi,

How to calculate an age from given date.

Comment by j...@prosyn.org, Oct 31, 2010

The equals method seems to throw an exception no matter what arguments you pass.

Comment by darian.c...@gmail.com, Nov 9, 2010

I've been trying to use .setTimeToNow() without success. I've looked through the source code and I think this function is missing? Can anyone confirm this please?

Is there an easy workaround to this issue? Many thanks :-)

Comment by darian.c...@gmail.com, Nov 9, 2010

Related to my comment above:

The regular Datejs from the download page does not have the setTimeToNow() function. This is the header:

Version: 1.0 Alpha-1 Build Date: 13-Nov-2007

However by checking out the Source tab I found there's a newer version of the core.js that includes it (http://code.google.com/p/datejs/source/browse/trunk/build/core.js). But this is missing parseExact()?! Here's the header from that:

version: 1.0 Alpha-1 date: 2008-05-13

This is very confusing. The API doesn't match the download :-(

Comment by Galkin.I...@gmail.com, Nov 11, 2010

Hi, Once the parse is complete, is it possible to know what format was it in, so that I can convert another date object to string in the same style? Thanks!

Comment by shane.co...@gmail.com, Dec 22, 2010

var myDate = new Date("2010-07-12T10:30:00Z"); works in firefox NaN in IE8 Am I doing something wrong?

Comment by mark.far...@gmail.com, Mar 3, 2011

Where is setTimeToNow()? Was it removed from the core??

Comment by njl_0...@163.com, Apr 4, 2011

where is the 'datediff()' function ???

Comment by steadm...@yahoo.com, Apr 5, 2011

I am trying to determine whether a particular date (not today()) is between two other dates. The documentation example uses: Date.today ().between(past, future); How can I modify the "today()" reference to be a date I want to compare past and future to?

Comment by GoodNat...@gmail.com, Apr 26, 2011

`getOrdinalNumber` should be renamed to `getDayOfYear` in the documentation to reflect the code.

also it appears `getWeek` has been renamed `getWeekOfYear`

Comment by claire.c...@gmail.com, Jun 13, 2011

Is there any way to get a number ordinal? ie 1st, 2nd, 3rd etc

Comment by rayno...@gmail.com, Jun 15, 2011

The method setTimezone simply add or subtract hours to date, or also calculate the daylight saving time respectly to the timezone?

Comment by jlivermo...@gmail.com, Jul 20, 2011

This is a great library. Thank you! Thank you! Thank you!

Comment by jimbrow...@gmail.com, Jul 26, 2011

There's an error in the documentation for getTimezoneAbbreviation: Instead of: Date.getTimezoneAbbreviation(today. getTimezoneOffset, today.isDayLightSavingsTime()); Should be: Date.getTimezoneAbbreviation(today.getUTCOffset(), today.isDayLightSavingsTime());

Comment by cam.mars...@gmail.com, Aug 30, 2011

I feel like I'm missing something. .isDaylightSavingTime() doesn't exist in date.js. I downloaded the entire package and the only reference to it is .isDST() in core-debug.js and it looks incomplete. Is there actually a .isDaylightSavingTime() somewhere? Am I missing something?

Comment by snagb...@gmail.com, Nov 3, 2011

Awesome library! Has helped me out immensely! Quick question though... Is it possible to get the dates that fall between 2 already established dates and place them into an array?

Thanks!

Comment by ned...@gmail.com, Nov 15, 2011

This is an awesome library, I use it whenever I need to do stuff with dates.

It would be great if something like John Resig's Pretty Date (http://ejohn.org/blog/javascript-pretty-date) or the slightly improved humaneDate.js (https://github.com/zachleat/Humane-Dates) were integrated into this, so you could things like

```
Date.today().next().thursday().toString('humane')
```

or maybe just

```
Date.today().humane()
```

Comment by ned...@gmail.com, Nov 15, 2011

... to continue on that thought, Moment.js (http://momentjs.com/) has a fromNow() method that does pretty dates:

```
moment().add('days', 7).fromNow()
```

though in other ways it is less elegant then date.js

Comment by JornAndr...@gmail.com, Nov 29, 2011

quote: "Awesome library! Has helped me out immensely! Quick question though... Is it possible to get the dates that fall between 2 already established dates and place them into an array?

Thanks!"

@snagb...@gmail.com Have you found a library that can do that? I need it too :)

Comment by j...@johnmills.co.nz, Dec 13, 2011

Great library! Definitely fills in the gaps of the default Date object.

Comment by eschop...@gmail.com, Jan 4, 2012

How can I globally change the Date object by 5 hours so every Date object is offset? Thanks!

Comment by darko.ro...@gmail.com, Jan 22, 2012

Hi, there's no method .compare(date1, date2) in date.js maybe the documentation is old?

darko

Comment by larosa.k...@gmail.com, Mar 4, 2012

I love you o_o

Comment by dominiq...@dommedia.net, Apr 10, 2012

Hi, I found "addDays" but where is "substractDays"?

Comment by kieran.s...@gmail.com, Apr 30, 2012

If you want to subtract days, use addDays but with a negative number.

Comment by lovevis...@gmail.com, May 30, 2012

Really nice.

I have one doubt is there is any function is available to compare the array of date. Example :- am pass one array in that array some of the date is present.

Am need the output is sort the date and return the array.

Like that any function is available ?

Please help me on this.

Comment by shailes...@gmail.com, Jun 14, 2012

Hi thanks so much for this library. I think there is a bug somewhere using the parse or parseExact method, try this:

```
alert("Jun 11 2012 12:52:48 PM using parseExact is: "
+ Date.parseExact("Jun 11 2012 12:52:48 PM", ["MMM dd yyyy hh:mm:ss tt"]).toString('yyyy-MM-dd hh:mm:ss tt')
+ "\n"
+ "Jun 11 2012 12:52:48 PM using new Date() is: "
+ (new Date("Jun 11 2012 12:52:48 PM PM")).toString('yyyy-MM-dd hh:mm:ss tt'));
```

The first is incorrect, the second is correct.

Comment by Tobias.S...@gmail.com, Jun 26, 2012

hi,

i got the following Code:

function TimeSpan?() {

var ArrivalStart? = $("#StartDatepicker?").val(); var ArrivalEnd? = $("#EndDatepicker?").val();

var ArrivalStarttest? = Date.parseExact(ArrivalStart?, "dd-MM-yyyy hh:mm"); var ArrivalEndtest? = Date.parseExact(ArrivalEnd?, "dd-MM-yyyy hh:mm"); var span = new TimePeriod?(ArrivalEndtest? - ArrivalStarttest?); alert(span); // Here i got object Object alert (span.getMinutes()); // And here every time i got 0

}

how can i get this to work

Comment by FARAZFRA...@gmail.com, Jun 28, 2012

How to compare 2 time (start time and end time) Neew to compare: starttime must be always less then endtime

var sttm = '07:00 PM'; var edtm = '09:00 AM';

how to compare using library 'date.js'

Thanks, someone reply fast...

-Frank

Comment by shounakg...@gmail.com, Jul 5, 2012

were these functions removed from the latest version? getDayName getMonthName isWeekday

Comment by louiseze...@gmail.com, Jul 26, 2012

Comment by cam.mars...@gmail.com, Aug 30, 2011

I feel like I'm missing something. .isDaylightSavingTime() doesn't exist in date.js. I downloaded the entire package and the only reference to it is .isDST() in core-debug.js and it looks incomplete. Is there actually a .isDaylightSavingTime() somewhere? Am I missing something?

I'm having the same problem - isDaylightSavingTime() doesn't seem to exist..

Comment by razvan.m...@gmail.com, Aug 8, 2012

How can be initialized from unix time (seconds since 1970)?

Comment by robert.l...@gmail.com, Aug 20, 2012

Any chance you will be adding support for ....toString('MMMM dS, yyyy')? as in August 2nd, 2012 since the link from "See FormatSpecifiers for more info." seemed to indicate S would be S The ordinal suffix ("st, "nd", "rd" or "th") of the current day. "st, "nd", "rd" or "th" But in practice I got August 2S, 2012

Comment by robert.l...@gmail.com, Aug 20, 2012

.toString('h:mm tt') returns 0:00 AM instead of 12:00 AM

Comment by ruben...@gmail.com, Aug 21, 2012

to louiseze...@gmail.com , Get the latest version from SVN http://datejs.googlecode.com/svn/trunk/

Comment by bitmapve...@gmail.com, Aug 25, 2012

How to get days on a month excluding Sunday(for example)?

I have try it trough a looping but i get stuck. any ideas? here is my code

```
var daysinmonth=Date.getDaysInMonth(2012,7);
var sundays=0;

for(i=1;i<=daysinmonth;i++){
        if(i.is().sunday()){
                sundays++;
        }
}
activedays = daysinmonth - sundays;
console.log(activedays);
```

Comment by vekec...@gmail.com, Aug 27, 2012

In response to ecoldwell's problem:

There appears to be a problem displaying and parsing 12 o'clock. "12pm" cannot be parsed. "12am" is parsed as 12pm. Midnight is displayed as "0:00:00 AM" instead of "12:00:00 AM"

Here's a link to the solution: http://code.google.com/p/datejs/issues/detail?id=134

Just follow the 'fix' instructions in the 'Please provide any additional information below.' section. It worked for me!

Comment by TimChau...@gmail.com, Oct 3, 2012

This script & documentation has nice ideas but is a mess... In the alpha version, the compare method is absent. In the release version, the parse method is absent.

Comment by anag...@gmail.com, Dec 9, 2012

How can you disable all dates except those which you specific in an array?

► Sign in to add a comment

Terms - Privacy - Project Hosting Help

Powered by Google Project Hosting