[Date](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date). Он содержит дату и время, а также предоставляет методы управления ими.

**[Создание](https://learn.javascript.ru/date" \l "sozdanie)**

Для создания нового объекта Date нужно вызвать конструктор new Date() с одним из следующих аргументов:

**new Date()**

Без аргументов – создать объект Date с текущими датой и временем:

let now = new Date();

alert( now ); // показывает текущие дату и время

**new Date(milliseconds)**

Создать объект Date с временем, равным количеству миллисекунд (тысячная доля секунды), прошедших с 1 января 1970 года UTC+0.

// 0 соответствует 01.01.1970 UTC+0

let Jan01\_1970 = new Date(0);

alert( Jan01\_1970 );

// теперь добавим 24 часа и получим 02.01.1970 UTC+0

let Jan02\_1970 = new Date(24 \* 3600 \* 1000);

alert( Jan02\_1970 );

Целое число, представляющее собой количество миллисекунд, прошедших с начала 1970 года, называется таймстамп (англ. timestamp).

Это – легковесное численное представление даты. Из таймстампа всегда можно получить дату с помощью new Date(timestamp) и преобразовать существующий объект Date в таймстамп, используя метод date.getTime()

Формула для вычисления количества миллисекунд в часе:

Количество миллисекунд в одной секунде: 1000 мс;

Количество секунд в одной минуте: 60 сек;

Количество минут в одном часе: 60 минут.

Количество миллисекунд в часе = (60 минут \* 60 секунд \* 1000 мс)

Количество миллисекунд в часе = 60 \* 60 \* 1000 = 3600000 мс

# Сколько миллисекунд в одном году

# Сколько миллисекунд в одном году

**31 536 000 000 мс или 31 622 400 000 мс если год високосный**

В одном году 31 536 000 000 мс или 31 622 400 000 мс если год високосный. Чтобы посчитать сколько миллисекунд в году, нужно перемножить количество миллисекунд в секунде, количество секунд в минуте, количество минут в часе, количество часов в сутках и результат умножить на количество дней в году.

**Расчет количества миллисекунд в году**  
1 000 мс \* 60 секунд \* 60 минут \* 24 часа \* 365 дней = 31 536 000 000 мс  
1 000 мс \* 60 секунд \* 60 минут \* 24 часа \* 366 дней = 31 622 400 000 мс (високосный год).

Датам до 1 января 1970 будут соответствовать отрицательные таймстампы, например:

// 31 декабря 1969 года

let Dec31\_1969 = new Date(-24 \* 3600 \* 1000);

alert( Dec31\_1969 );

**new Date(datestring)**

Если аргумент всего один, и это строка, то из неё «прочитывается» дата. Алгоритм разбора – такой же, как в Date.parse

let date = new Date("2017-01-26");

alert(date);

// Время не указано, поэтому оно ставится в полночь по Гринвичу и

// меняется в соответствии с часовым поясом места выполнения кода

// Так что в результате можно получить

// Thu Jan 26 2017 11:00:00 GMT+1100 (восточно-австралийское время)

// или

// Wed Jan 25 2017 16:00:00 GMT-0800 (тихоокеанское время)

**new Date(year, month, date, hours, minutes, seconds, ms)**

Создать объект Date с заданными компонентами в местном часовом поясе. Обязательны только первые два аргумента.

* year должен состоять из четырёх цифр. Для совместимости также принимаются 2 цифры и рассматриваются как 19xx, к примеру, 98 здесь это тоже самое, что и 1998, но настоятельно рекомендуется всегда использовать 4 цифры.
* month начинается с 0 (январь) по 11 (декабрь).
* Параметр date здесь представляет собой день месяца. Если параметр не задан, то принимается значение 1.
* Если параметры hours/minutes/seconds/ms отсутствуют, их значением становится 0.

Например:

new Date(2011, 0, 1, 0, 0, 0, 0); // // 1 Jan 2011, 00:00:00

new Date(2011, 0, 1); // то же самое, так как часы и проч. равны 0

Максимальная точность – 1 мс (до 1/1000 секунды):

let date = new Date(2011, 0, 1, 2, 3, 4, 567);

alert( date ); // 1.01.2011, 02:03:04.567

**[Получение компонентов даты](https://learn.javascript.ru/date" \l "poluchenie-komponentov-daty)**

Существуют методы получения года, месяца и т.д. из объекта Date:

[**getFullYear()**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getFullYear)

Получить год (4 цифры)

[**getMonth()**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getMonth)

Получить месяц, **от 0 до 11**.

[**getDate()**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getDate)

Получить день месяца, от 1 до 31, что несколько противоречит названию метода.

[**getHours()**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getHours)**, [getMinutes()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getMinutes), [getSeconds()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getSeconds), [getMilliseconds()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getMilliseconds)**

Получить, соответственно, часы, минуты, секунды или миллисекунды.

**Никакого getYear(). Только getFullYear()**

Многие интерпретаторы JavaScript реализуют нестандартный и устаревший метод getYear(), который порой возвращает год в виде двух цифр. Пожалуйста, обходите его стороной. Если нужно значение года, используйте getFullYear().

Кроме того, можно получить определённый день недели:

[**getDay()**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getDay)

Вернуть день недели от 0 (воскресенье) до 6 (суббота). Несмотря на то, что в ряде стран за первый день недели принят понедельник, в JavaScript начало недели приходится на воскресенье.

Однако существуют и их UTC-варианты, возвращающие день, месяц, год для временной зоны UTC+0: [getUTCFullYear()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getUTCFullYear), [getUTCMonth()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getUTCMonth), [getUTCDay()](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getUTCDay). Для их использования требуется после "get" подставить "UTC".

Если ваш местный часовой пояс смещён относительно UTC, то следующий код покажет разные часы:

// текущая дата

let date = new Date();

// час в вашем текущем часовом поясе

alert( date.getHours() );

// час в часовом поясе UTC+0 (лондонское время без перехода на летнее время)

alert( date.getUTCHours() );

Помимо вышеприведённых методов, существуют два особых метода без UTC-варианта:

[**getTime()**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getTime)

Для заданной даты возвращает таймстамп – количество миллисекунд, прошедших с 1 января 1970 года UTC+0.

[**getTimezoneOffset()**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getTimezoneOffset)

Возвращает разницу в минутах между UTC и местным часовым поясом:

// если вы в часовом поясе UTC-1, то выводится 60

// если вы в часовом поясе UTC+3, выводится -180

alert( new Date().getTimezoneOffset() );

## [Установка компонентов даты](https://learn.javascript.ru/date" \l "ustanovka-komponentov-daty)

Следующие методы позволяют установить компоненты даты и времени:

* [setFullYear(year, [month], [date])](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/setFullYear)
* [setMonth(month, [date])](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/setMonth)
* [setDate(date)](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/setDate)
* [setHours(hour, [min], [sec], [ms])](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/setHours)
* [setMinutes(min, [sec], [ms])](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/setMinutes)
* [setSeconds(sec, [ms])](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/setSeconds)
* [setMilliseconds(ms)](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/setMilliseconds)
* [setTime(milliseconds)](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/setTime) (устанавливает дату в виде целого количества миллисекунд, прошедших с 01.01.1970 UTC)

У всех этих методов, кроме setTime(), есть UTC-вариант, например: setUTCHours().

Как мы видим, некоторые методы могут устанавливать сразу несколько компонентов даты, например: setHours. Если какая-то компонента не указана, она не меняется.

Пример:

let today = new Date();

today.setHours(0);

alert(today); // выводится сегодняшняя дата, но значение часа будет 0

today.setHours(0, 0, 0, 0);

alert(today); // всё ещё выводится сегодняшняя дата, но время будет ровно 00:00:00.

## [Автоисправление даты](https://learn.javascript.ru/date" \l "avtoispravlenie-daty)

Автоисправление – это очень полезная особенность объектов Date. Можно устанавливать компоненты даты вне обычного диапазона значений, а объект сам себя исправит.

Пример:

let date = new Date(2013, 0, 32); // 32 Jan 2013 ?!?

alert(date); // ...1st Feb 2013!

Предположим, нам требуется увеличить дату «28 февраля 2016» на два дня. В зависимости от того, високосный это год или нет, результатом будет «2 марта» или «1 марта». Нам об этом думать не нужно. Просто прибавляем два дня. Объект Date позаботится об остальном:

let date = new Date(2016, 1, 28);

date.setDate(date.getDate() + 2);

alert( date ); // 1 Mar 2016

Эту возможность часто используют, чтобы получить дату по прошествии заданного отрезка времени. Например, получим дату «спустя 70 секунд с текущего момента»:

let date = new Date();

date.setSeconds(date.getSeconds() + 70);

alert( date ); // выводит правильную дату

Также можно установить нулевые или даже отрицательные значения. Например:

let date = new Date(2016, 0, 2); // 2 Jan 2016

date.setDate(1); // задать первое число месяца

alert( date );

date.setDate(0); // первый день месяца -- это 1, так что выводится последнее число предыдущего месяца

alert( date ); // 31 Dec 2015

## [Преобразование к числу, разность дат](https://learn.javascript.ru/date" \l "preobrazovanie-k-chislu-raznost-dat)

Если объект Date преобразовать в число, то получим таймстамп по аналогии с date.getTime():

let date = new Date();

alert(+date); // количество миллисекунд, то же самое, что date.getTime()

Важный побочный эффект: даты можно вычитать, в результате получаем разность в миллисекундах.

Этот приём можно использовать для измерения времени:

let start = new Date(); // начинаем отсчёт времени

// выполняем некоторые действия

for (let i = 0; i < 100000; i++) {

let doSomething = i \* i \* i;

}

let end = new Date(); // заканчиваем отсчёт времени

alert( `Цикл отработал за ${end - start} миллисекунд` );

## [Date.now()](https://learn.javascript.ru/date" \l "date-now)

Если нужно просто измерить время, объект Date нам не нужен.

Существует особый метод Date.now(), возвращающий текущую метку времени.

Семантически он эквивалентен new Date().getTime(), однако метод не создаёт промежуточный объект Date. Так что этот способ работает быстрее и не нагружает сборщик мусора.

Данный метод используется из соображений удобства или когда важно быстродействие, например, при разработке игр на JavaScript или других специализированных приложений.

Вероятно, предыдущий пример лучше переписать так:

let start = Date.now(); // количество миллисекунд с 1 января 1970 года

// выполняем некоторые действия

for (let i = 0; i < 100000; i++) {

let doSomething = i \* i \* i;

}

let end = Date.now(); // заканчиваем отсчёт времени

alert( `Цикл отработал за ${end - start} миллисекунд` ); // вычитаются числа, а не даты

## [Бенчмаркинг](https://learn.javascript.ru/date" \l "benchmarking) – не дошло

Будьте внимательны, если хотите точно протестировать производительность функции, которая зависит от процессора.

Например, сравним две функции, вычисляющие разницу между двумя датами: какая сработает быстрее?

Подобные вычисления, замеряющие производительность, также называют «бенчмарками» (benchmark).

// есть date1 и date2, какая функция быстрее вернёт разницу между ними в миллисекундах?

function diffSubtract(date1, date2) {

return date2 - date1;

}

// или

function diffGetTime(date1, date2) {

return date2.getTime() - date1.getTime();

## [Разбор строки с датой](https://learn.javascript.ru/date" \l "razbor-stroki-s-datoy)

Метод [Date.parse(str)](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/parse) считывает дату из строки.

Формат строки должен быть следующим: YYYY-MM-DDTHH:mm:ss.sssZ, где:

* YYYY-MM-DD – это дата: год-месяц-день.
* Символ "T" используется в качестве разделителя.
* HH:mm:ss.sss – время: часы, минуты, секунды и миллисекунды.
* Необязательная часть 'Z' обозначает часовой пояс в формате +-hh:mm. Если указать просто букву Z, то получим UTC+0.

Возможны и более короткие варианты, например, YYYY-MM-DD или YYYY-MM, или даже YYYY.

Вызов Date.parse(str) обрабатывает строку в заданном формате и возвращает таймстамп (количество миллисекунд с 1 января 1970 года UTC+0). Если формат неправильный, возвращается NaN.

Например:

let ms = Date.parse('2012-01-26T13:51:50.417-07:00');

alert(ms); // 1327611110417 (таймстамп)

Можно тут же создать объект new Date из таймстампа:

let date = new Date( Date.parse('2012-01-26T13:51:50.417-07:00') );

alert(date);

## [Итого](https://learn.javascript.ru/date" \l "itogo)

* Дата и время в JavaScript представлены объектом [Date](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date). Нельзя создать «только дату» или «только время»: объекты Date всегда содержат и то, и другое.
* Счёт месяцев начинается с нуля (да, январь – это нулевой месяц).
* Дни недели в getDay() также отсчитываются с нуля, что соответствует воскресенью.
* Объект Date самостоятельно корректируется при введении значений, выходящих за рамки допустимых. Это полезно для сложения/вычитания дней/месяцев/недель.
* Даты можно вычитать, и разность возвращается в миллисекундах. Так происходит, потому что при преобразовании в число объект Date становится таймстампом.
* Используйте Date.now() для быстрого получения текущего времени в формате таймстампа.

Учтите, что, в отличие от некоторых других систем, в JavaScript таймстамп в миллисекундах, а не в секундах.

Порой нам нужно измерить время с большей точностью. Собственными средствами JavaScript измерять время в микросекундах (одна миллионная секунды) нельзя, но в большинстве сред такая возможность есть. К примеру, в браузерах есть метод [performance.now()](https://developer.mozilla.org/ru/docs/Web/API/Performance/now), возвращающий количество миллисекунд с начала загрузки страницы с точностью до микросекунд (3 цифры после точки):

alert(`Загрузка началась ${performance.now()}мс назад`);

// Получаем что-то вроде: "Загрузка началась 34731.26000000001мс назад"

// .26 –- это микросекунды (260 микросекунд)

// корректными являются только первые три цифры после точки, а остальные -- это ошибка точности

В Node.js для этого предусмотрен модуль microtime и ряд других способов. Технически почти любое устройство или среда позволяет добиться большей точности, просто её нет в объекте Date.