**Дата и время**

Встроенный объект: Date

## Создание

Содание new Date(milliseconds) объект Date с временем, равным количеству **миллисекунд** (*таймстамп* (англ. timestamp)), прошедших с **1 января 1970 года** UTC+0., без аргументов – создать объект Date с текущими датой и временем:

let now = **new Date();**

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

Датам до 1 января 1970 будут соответствовать отрицательные таймстампы.

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: значение 2013 корректно, 98 – нет.
* **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

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

Date.now()

## Получение компонентов даты

[**getFullYear()**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getFullYear) - Получить год (4 цифры) (**getYear**- год в виде двух цифр. Нежелательно использовать)

[**getMonth()**](https://developer.mozilla.org/ru/docs/Web/JavaScript/Reference/Global_Objects/Date/getMonth) - Получить месяц, **от 0 до 11**. Счёт месяцев начинается с нуля (да, январь – это нулевой месяц).

[**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.

**getDay()** - Вернуть день недели от 0 (воскресенье) до 6 (суббота).

Все вышеперечисленные методы возвращают значения в соответствии с местным часовым поясом. 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).

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

## Измерить время

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.

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` );

## Разбор строки с датой

Метод **Date.parse(str)** считывает дату из строки. Формат строки должен быть следующим: **YYYY-MM-DDTHH:mm:ss.sssZ**, где 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);

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

* [**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().

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.

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

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

alert( date ); // 1 Mar 2016

## Преобразование к числу, разность дат

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` );

В этом примере лучше использовать **Date.now()** вместо **new Date**

**Строковое** сравнение дат:

let date1 = '2020-12-01';

let date2 = '2019-12-01';

Сравнение **объектов**:

let date1 = new Date(2020, 1, 1);

let date2 = new Date(2021, 1, 1);

console.log(date1 > date2); // выведет false

## Точность имерения

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(), 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.