

Раздел

Типы данных

Навигация по уроку

Создание

Получение компонентов
датыУстановка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

Date.now()

Бенчмаркинг

Разбор строки с датой

Итого


Задачи (8)

Комментарии

Поделиться



Редактировать на GitHub

 → [Язык программирования JavaScript](#)
→ [Типы данных](#) 24-го сентября 2020

Дата и время

Встречайте новый встроенный объект: [Date](#). Он содержит дату и время, а также предоставляет методы управления ими.

Например, его можно использовать для хранения времени создания/изменения, для измерения времени или просто для вывода текущей даты.

Создание

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:

```
1 let now = new Date();  
2 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.

```
1 // 0 соответствует 01.01.1970 UTC+0  
2 let Jan01_1970 = new Date(0);  
3 alert( Jan01_1970 );  
4  
5 // теперь добавим 24 часа и получим 02.01.1970 UTC+0  
6 let Jan02_1970 = new Date(24 * 3600 * 1000);  
7 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()` (см. ниже).

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

```
1 // 31 декабря 1969 года  
2 let Dec31_1969 = new Date(-24 * 3600 * 1000);  
3 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`, который мы рассмотрим позже.

```
1 let date = new Date("2017-01-26");  
2 alert(date);  
3 // Время не указано, поэтому оно ставится в полночь по  
4 // меняется в соответствии с часовым поясом места выпол  
5 // Так что в результате можно получить  
6 // Thu Jan 26 2017 11:00:00 GMT+1100 (восточно-австрали  
7
```

Раздел

Типы данных

Навигация по уроку

Создание

Получение компонентов
даты

Установка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

Date.now()

Бенчмаркинг

Разбор строки с датой

Итого

Задачи (8)

Комментарии

Поделиться



Редактировать на GitHub



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.

Например:

```
1 new Date(2011, 0, 1, 0, 0, 0, 0); // // 1 Jan 2011, 00:00:00  
2 new Date(2011, 0, 1); // то же самое, так как часы и пр
```

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

```
1 let date = new Date(2011, 0, 1, 2, 3, 4, 567);  
2 alert( date ); // 1.01.2011, 02:03:04.567
```

Получение компонентов даты

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

getFullYear()

Получить год (4 цифры)

getMonth()

Получить месяц, от 0 до 11.

getDate()

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

getHours(), getMinutes(), getSeconds(), 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()

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.

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()`, `getUTCMonth()`, `getUTCDay()`. Для их использования требуется после "get" подставить "UTC".

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

```
1 // текущая дата
2 let date = new Date();
3
4 // час в вашем текущем часовом поясе
5 alert( date.getHours() );
6
7 // час в часовом поясе UTC+0 (лондонское время без пере-
8 alert( date.getUTCHours() );
```

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

`getTime()`

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

`getTimezoneOffset()`

Возвращает разницу в минутах между местным часовым поясом и UTC:

```
1 // если вы в часовом поясе UTC-1, то выводится 60
2 // если вы в часовом поясе UTC+3, выводится -180
3 alert( new Date().getTimezoneOffset() );
```

Установка компонентов даты

Следующие методы позволяют установить компоненты даты и времени:

- `setFullYear(year, [month], [date])`
- `setMonth(month, [date])`
- `setDate(date)`
- `setHours(hour, [min], [sec], [ms])`
- `setMinutes(min, [sec], [ms])`
- `setSeconds(sec, [ms])`
- `setMilliseconds(ms)`
- `setTime(milliseconds)` (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:

```
1 let today = new Date();
2
3 today.setHours(0);
4 alert(today); // выводится сегодняшняя дата, но значени-
5
6 today.setHours(0, 0, 0, 0);
7 alert(today); // всё ещё выводится сегодняшняя дата, но
```

Автоисправление даты

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

Пример:

Раздел

Типы данных

Навигация по уроку

Создание

Получение компонентов
даты

Установка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

`Date.now()`

Бенчмаркинг

Разбор строки с датой

Итого

Задачи (8)

Комментарии

Поделиться



Редактировать на GitHub

Раздел

Типы данных

Навигация по уроку

Создание

Получение компонентов
даты

Установка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

Date.now()

Бенчмаркинг

Разбор строки с датой

Итого

Задачи (8)

Комментарии

Поделиться



Редактировать на GitHub



```
1 let date = new Date(2013, 0, 32); // 32 Jan 2013
2 alert(date); // ...1st Feb 2013!
```

Неправильные компоненты даты автоматически распределяются по остальным.

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:

```
1 let date = new Date(2016, 1, 28);
2 date.setDate(date.getDate() + 2);
3
4 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»:

```
1 let date = new Date();
2 date.setSeconds(date.getSeconds() + 70);
3
4 alert( date ); // выводит правильную дату
```

Также можно установить нулевые или даже отрицательные значения. Например:

```
1 let date = new Date(2016, 0, 2); // 2 Jan 2016
2
3 date.setDate(1); // задать первое число месяца
4 alert( date );
5
6 date.setDate(0); // первый день месяца -- это 1, так что
7 alert( date ); // 31 Dec 2015
```

Преобразование к числу, разность дат

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

```
1 let date = new Date();
2 alert(+date); // количество миллисекунд, то же самое, что
```

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

Этот приём можно использовать для измерения времени:

```
1 let start = new Date(); // начинаем отсчёт времени
2
3 // выполняем некоторые действия
4 for (let i = 0; i < 100000; i++) {
5   let doSomething = i * i * i;
6 }
7
8 let end = new Date(); // заканчиваем отсчёт времени
9
10 alert( `Цикл отработал за ${end - start} миллисекунд` )
```

Date.now()

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

Раздел

[Типы данных](#)

Навигация по уроку

Создание

Получение компонентов
даты

Установка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

Date.now()

Бенчмаркинг

Разбор строки с датой

Итого

Задачи (8)

Комментарии

Поделиться



Редактировать на GitHub



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:

```
1 let start = Date.now(); // количество миллисекунд
2
3 // выполняем некоторые действия
4 for (let i = 0; i < 100000; i++) {
5   let doSomething = i * i * i;
6 }
7
8 let end = Date.now(); // заканчиваем отсчёт времени
9
10 alert( `Цикл отработал за ${end - start} миллисекунд` )
```

Бенчмаркинг

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, измеряющие производительность, также называют «бенчмарками» (benchmark).



```
1 // есть date1 и date2, какая функция быстрее вернёт раз
2 function diffSubtract(date1, date2) {
3   return date2 - date1;
4 }
5
6 // или
7 function diffGetTime(date1, date2) {
8   return date2.getTime() - date1.getTime();
9 }
```



Обе функции делают буквально одно и то же, только одна использует явный метод `date.getTime()` для получения даты в миллисекундах, а другая полагается на преобразование даты в число. Результат их работы всегда один и тот же.

Но какая функция быстрее?

Для начала можно запустить их много раз подряд и засечь разницу. В нашем случае функции очень простые, так что потребуется хотя бы 100000 повторений.

Проведём измерения:

```
1 function diffSubtract(date1, date2) {
2   return date2 - date1;
3 }
4
5 function diffGetTime(date1, date2) {
6   return date2.getTime() - date1.getTime();
7 }
8
9 function bench(f) {
10  let date1 = new Date(0);
11  let date2 = new Date();
12 }
```

Раздел

Типы данных

Навигация по уроку

Создание

Получение компонентов
даты

Установка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

Date.now()

Бенчмаркинг

Разбор строки с датой

Итого

Задачи (8)

Комментарии

Поделиться



Редактировать на GitHub



```
13 let start = Date.now();
14 for (let i = 0; i < 100000; i++) f(date1, date2);
15 return Date.now() - start;
16 }
17
18 alert( 'Время diffSubtract: ' + bench(diffSubtract) + '!' );
19 alert( 'Время diffGetTime: ' + bench(diffGetTime) + 'мс' );
```

Вот это да! Метод `getTime()` работает ощутимо быстрее! Всё потому, что не производится преобразование типов, и интерпретаторам такое намного легче оптимизировать.

Замечательно, это уже что-то. Но до хорошего бенчмарка нам ещё далеко.

Представьте, что при выполнении `bench(diffSubtract)` процессор параллельно делал что-то ещё, также потребляющее ресурсы. А к началу выполнения `bench(diffGetTime)` он это уже завершил.

Достаточно реалистичный сценарий в современных многопроцессорных операционных системах.

В итоге у первого бенчмарка окажется меньше ресурсов процессора, чем у второго. Это может исказить результаты.

Для получения наиболее достоверных результатов тестирования производительности весь набор бенчмарков нужно запускать по несколько раз.

Например, так:

```
1 function diffSubtract(date1, date2) {
2   return date2 - date1;
3 }
4
5 function diffGetTime(date1, date2) {
6   return date2.getTime() - date1.getTime();
7 }
8
9 function bench(f) {
10  let date1 = new Date(0);
11  let date2 = new Date();
12
13  let start = Date.now();
14  for (let i = 0; i < 100000; i++) f(date1, date2);
15  return Date.now() - start;
16 }
17
18 let time1 = 0;
19 let time2 = 0;
20
21 // bench(upperSlice) и bench(upperLoop) поочередно запу
22 for (let i = 0; i < 10; i++) {
23   time1 += bench(diffSubtract);
24   time2 += bench(diffGetTime);
25 }
26
27 alert( 'Итоговое время diffSubtract: ' + time1 );
28 alert( 'Итоговое время diffGetTime: ' + time2 );
```

Современные интерпретаторы JavaScript начинают применять продвинутые оптимизации только к «горячему коду», выполняющемуся несколько раз (незачем оптимизировать то, что редко выполняется). Так что в примере выше первые запуски не оптимизированы должным образом. Нелишним будет добавить предварительный запуск для «разогрева»:

```
1 // добавляем для "разогрева" перед основным циклом
2 bench(diffSubtract);
3 bench(diffGetTime);
4
5 // а теперь тестируем производительность
```

Раздел

Типы данных

Навигация по уроку

Создание

Получение компонентов
даты

Установка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

Date.now()

Бенчмаркинг

Разбор строки с датой

Итого

Задачи (8)

Комментарии

Поделиться



Редактировать на GitHub



```
6 for (let i = 0; i < 10; i++) {
7   time1 += bench(diffSubtract);
8   time2 += bench(diffGetTime);
9 }
```

⚠ Будьте осторожны с микробенчмарками

Современные интерпретаторы JavaScript выполняют множество оптимизаций. Они могут повлиять на результаты «искусственных тестов» по сравнению с «нормальным использованием», особенно если мы тестируем что-то очень маленькое, например, работу оператора или встроенной функции. Поэтому если хотите серьезно понять производительность, пожалуйста, изучите, как работают интерпретаторы JavaScript. И тогда вам, вероятно, уже не понадобятся микробенчмарки.

Отличный набор статей о V8 можно найти на <http://mrale.ph>.

Разбор строки с датой

Метод `Date.parse(str)` 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:

```
1 let ms = Date.parse('2012-01-26T13:51:50.417-07:00');
2
3 alert(ms); // 1327611110417 (таймстамп)
```

Можно тут же создать объект `new Date` из таймстампа:

```
1 let date = new Date( Date.parse('2012-01-26T13:51:50.417-07:00') );
2
3 alert(date);
```

Итого

- Дата и время в JavaScript представлены объектом `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.

Раздел

[Типы данных](#)

Навигация по уроку

Создание

Получение компонентов
даты

Установка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

Date.now()

Бенчмаркинг

Разбор строки с датой

Итого

Задачи (8)

Комментарии

Поделиться



[Редактировать на GitHub](#)



- 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()`, возвращающий количество миллисекунд с начала загрузки страницы с точностью до микросекунд (3 цифры после точки):

```
1 alert(`Загрузка началась ${performance.now()}мс назад`);
2 // Получаем что-то вроде: "Загрузка началась 34731.2600
3 // .26 -- это микросекунды (260 микросекунд)
4 // корректными являются только первые три цифры после т
```

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`.

✓ Задачи

Создайте дату

важность: 5

Создайте объект `Date` для даты: 20 февраля 2012 года, 3 часа 12 минут. Временная зона – местная.

Для вывода используйте `alert`.

решение

Покажите день недели

важность: 5

Напишите функцию `getWeekDay(date)`, показывающую день недели в коротком формате: «ПН», «ВТ», «СР», «ЧТ», «ПТ», «СБ», «ВС».

Например:

```
1 let date = new Date(2012, 0, 3); // 3 января 2012 года
2 alert( getWeekDay(date) );      // нужно вывести "ВТ"
```

[Открыть песочницу с тестами для задачи.](#)

решение

День недели в европейской нумерации

важность: 5

В Европейских странах неделя начинается с понедельника (день номер 1), затем идёт вторник (номер 2) и так до воскресенья (номер 7). Напишите функцию `getLocalDay(date)`, которая возвращает «европейский» день недели для даты `date`.

```
1 let date = new Date(2012, 0, 3); // 3 января 2012 года
2 alert( getLocalDay(date) );      // вторник, нужно пок
```

[Открыть песочницу с тестами для задачи.](#)

решение

Раздел

[Типы данных](#)

Навигация по уроку

Создание

Получение компонентов
даты

Установка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

Date.now()

Бенчмаркинг

Разбор строки с датой

Итого

Задачи (8)

Комментарии

Поделиться



[Редактировать на GitHub](#)



Какой день месяца был много дней назад?

важность: 4

Создайте функцию `getDateAgo(date, days)`, возвращающую число, которое было `days` дней назад от даты `date`.

К примеру, если сегодня двадцатое число, то `getDateAgo(new Date(), 1)` вернёт девятнадцатое и `getDateAgo(new Date(), 2)` – восемнадцатое.

Функция должна надёжно работать при значении `days=365` и больших значениях:

```
1 let date = new Date(2015, 0, 2);
2
3 alert( getDateAgo(date, 1) ); // 1, (1 Jan 2015)
4 alert( getDateAgo(date, 2) ); // 31, (31 Dec 2014)
5 alert( getDateAgo(date, 365) ); // 2, (2 Jan 2014)
```

P.S. Функция не должна изменять переданный ей объект `date`.

[Открыть песочницу с тестами для задачи.](#)

решение

Последнее число месяца?

важность: 5

Напишите функцию `getLastDayOfMonth(year, month)`, возвращающую последнее число месяца. Иногда это 30, 31 или даже февральские 28/29.

Параметры:

- `year` – год из четырёх цифр, например, 2012.
- `month` – месяц от 0 до 11.

К примеру, `getLastDayOfMonth(2012, 1) = 29` (високосный год, февраль).

[Открыть песочницу с тестами для задачи.](#)

решение

Сколько сегодня прошло секунд?

важность: 5

Напишите функцию `getSecondsToday()`, возвращающую количество секунд с начала сегодняшнего дня.

Например, если сейчас 10:00, и не было перехода на зимнее/летнее время, то:

```
1 getSecondsToday() == 36000 // (3600 * 10)
```

Функция должна работать в любой день, т.е. в ней не должно быть конкретного значения сегодняшней даты.

решение

Сколько секунд осталось до завтра?

важность: 5

Создайте функцию `getSecondsToTomorrow()`, возвращающую количество секунд до завтрашней даты.

Например, если сейчас 23:00, то:

```
1 getSecondsToTomorrow() == 3600
```

P.S. Функция должна работать в любой день, т.е. в ней не должно быть конкретного значения сегодняшней даты.

решение

Форматирование относительной даты

важность: 4

Напишите функцию `formatDate(date)`, форматирующую `date` по следующему принципу:

- Если спустя `date` прошло менее 1 секунды, вывести "прямо сейчас".
- В противном случае, если с `date` прошло меньше 1 минуты, вывести "n сек. назад".
- В противном случае, если меньше часа, вывести "m мин. назад".
- В противном случае, полная дата в формате "DD.MM.YY HH:mm". А именно: "день.месяц.год часы:минуты", всё в виде двух цифр, т.е. 31.12.16 10:00.

Например:

```
1 alert( formatDate(new Date(new Date - 1)) ); // "прямо
2
3 alert( formatDate(new Date(new Date - 30 * 1000)) ); //
4
5 alert( formatDate(new Date(new Date - 5 * 60 * 1000)) );
6
7 // вчерашняя дата вроде 31.12.2016, 20:00
8 alert( formatDate(new Date(new Date - 86400 * 1000)) );
```

Открыть песочницу с тестами для задачи.

решение

Проводим курсы по JavaScript и фреймворкам. 

Комментарии

перед тем как писать...

Раздел

Типы данных

Навигация по уроку

Создание

Получение компонентов
даты

Установка компонентов
даты

Автоисправление даты

Преобразование к числу,
разность дат

`Date.now()`

Бенчмаркинг

Разбор строки с датой

Итого

Задачи (8)

Комментарии

Поделиться



Редактировать на GitHub