Объект Date

Введение

JavaScript-объект Date позволяет работать с датами.

Объект Date, представляет собой момент времени. Объект содержит число **миллисекунд, прошедших с 1 января 1970 г. UTC**.

> ***Важно***. Целое число, представляющее собой **количество миллисекунд, прошедших с начала 1970 года**, называется **таймстамп** (англ. *timestamp*).
>
> ***Метка таймстамп:*** Thu Jan 01 1970 00:00:00 GMT+0300 (Москва, стандартное время).

По умолчанию, JavaScript будет использовать временную зону браузера и **отображать дату в виде полной текстовой строки**, например:

`Tue Nov 29 2022 11:00:18 GMT+0300 (Москва, стандартное время).`

Объект Date создается при помощи конструктора **new Date**(). Существует **четыре** способа создания нового объекта даты:

1. new **Date**()
2. new **Date**(milliSecond)
3. new **Date**(dateString)
4. new **Date**(year, month[, day[, hour[, minute[, second[, milliSecond]]]]])

new Date()

Конструктор new **Date**() создает новый объект даты с **текущими** датой и временем.

example_1. Конструктор new Date()

```
<script>
    // создаем специальный объект Date
    var d = new Date();
    // Thu Dec 15 2022 10:29:45 GMT+0300 (Москва, стандартное время)
    console.log(d);
</script>
```

new Date(milliSecond)

Параметр

- **milliSecond –** целое значение, представляющее количество миллисекунд, **прошедших** с 1 января 1970 00:00:00 по UTC (эпоха Unix).

example_2. Конструктор new Date(milliSecond)

```
<script>
    // метка timestamp
    console.log("Thu Jan 01 1970 00:00:00 GMT+0300 (Москва, стандартное время)");
    // на вход функции Date подадим количество миллисекунд,
    // прошедших с начала эпохи Unix
    // 24*60*60*1000 = 1 сутки
    var d = new Date(24*60*60*1000);
```

```
    // метка timestamp + 1 сутки
    console.log(d);
</script>
```

new Date(dateString)

Параметр

- **dateString –** строковое значение, представляющее дату. Строка должна быть в одном из форматов, распознаваемых методом Date.**parse**() (совместимые со стандартами временные метки).

example_3. Конструктор new Date(dateString)

```
<script>
    // корректная строка 1
    var strDate1 = "Tue Nov 29 2022 11:52:30 GMT+0300";
    // корректная строка 2
    var strDate2 = "10-23-2011";
    // корректная строка 3
    var strDate3 = "March 08, 1997";
    // корректная строка 4 ??
    var strDate4 = "Сент 10, 2021";
    // вывод значения объекта Date
    console.log(new Date(strDate1));
    console.log(new Date(strDate2));
    console.log(new Date(strDate3));
    console.log(new Date(strDate4));
</script>
```

new Date(year, month[, day[, hour[, minute[, second[, millisecond]]]]])

Конструктор new **Date**(year, month, ...) создает новый объект даты с заданными датой и временем.

Семь числовых параметров определяют год, месяц, день, часы, минуты, секунды, миллисекунды (именно в таком **порядке**).

Параметры

- **year** – целое значение, представляющее год. Значения с 0 по 99 отображаются на года с 1900 по 1999. Смотрите пример ниже.
- **month** – целое значение, представляющее месяц, начинается с 0 для января и кончается 11 для декабря.
- **day – необязательный** параметр. Целое значение, представляющее день месяца.
- **hour** – **необязательный** параметр. Целое значение, представляющее часы дня.
- **minute** – **необязательный** параметр. Целое значение, представляющее минуты времени.
- **second** – **необязательный** параметр. Целое значение, представляющее секунды времени.
- **milliSecond** – **необязательный** параметр. Целое значение, представляющее миллисекунды времени.

> ***Внимание*.**
>
> - В JavaScript нумерация месяцев идет **от 0 до 11**. Январь - 0. Декабрь - 11.
> - Нельзя **опускать параметр месяца**. Если задается один параметр, то он будет интерпретироваться как **миллисекунды**.

example_4. Конструктор new Date(year, month[, ...])

```
<script>
    // 2 числовых параметра определяют: -год, -месяц
    var d2 = new Date(2022, 11);
    console.log(d2);
    // 3 числовых параметра определяют: -год, -месяц, -день
    var d3 = new Date(2022, 11, 12);
    console.log(d3);
    // 4 числовых параметров определяют: -год, -месяц, -день, -часы
    var d4 = new Date(2022, 11, 22, 10);
    console.log(d4);
    // 5 числовых параметров определяют: -год, -месяц, -день, -часы, -минуты
    var d5 = new Date(2022, 11, 2, 10, 33);
    console.log(d5);
    // 6 числовых параметров определяют: -год, -месяц, -день, -часы, -минуты, -секунды
    var d6 = new Date(2022, 11, 4, 10, 33, 30);
    console.log(d6);
    // 7 числовых параметров определяют: -год, -месяц, -день, -часы, -минуты, -секунды, -миллисекунды
    var d7 = new Date(2022, 11, 21, 10, 33, 30, 55);
    console.log(d7);
</script>
```

Методы-аксессоры объекта Date

Когда объект **Date** создан, вы можете оперировать им при помощи его методов.

Методы объекта Date позволяют **получать** и **устанавливать** год, месяц, день, час, минуты, секунды и миллисекунды в объекте даты, используя как локальное время, так и UTC или GMT.

Мы уже достаточно много говорили об объектах и работали с объектами.

Есть **два типа свойств** объекта.

- **Свойства-данные** (*data properties*). В уроках и самостоятельных работах мы достаточно много работали со свойствами объектов.
- **Свойства-аксессоры** (*accessor properties*). По своей сути это функции, которые используются для **присвоения** (сеттеры) и **получения** (геттеры) значения. Во внешнем коде они выглядят как обычные свойства объекта.

Свойства-аксессоры представлены методами:

- **геттер** (от англ. get - **получить**) — это метод, который получает значение определённого свойства.
- **сеттер** (от англ. set — **присвоить**) — это метод, который присваивает значение определённому свойству объекта.

Вы можете определить геттеры и сеттеры для любых из встроенных или определённых вами объектов, которые поддерживают добавление новых свойств.

Методы геттеры

Следующие методы можно использовать для получения информации из объекта даты.

Метод	Описание

getTime()	Получить время (количество миллисекунд, прошедших с 1 января 1970)
getFullYear()	Получить год в формате четырех цифр (гггг)
getMonth()	Получить номер месяца (0-11)
getDate()	Получить число месяца (1-31)
getDay()	Получить номер дня недели (0-6)
getHours()	Получить час (0-23)
getMinutes()	Получить минуты (0-59)
getSeconds()	Получить секунды (0-59)
getMilliseconds()	Получить миллисекунды (0-999)

Метод getTime()

Метод **getTime**() возвращает количество **миллисекунд**, прошедших с 1 января 1970:

```
var d = new Date();

console.log(d.getTime());
```

Метод getFullYear()

Метод **getFullYear**() возвращает год в формате четырех цифр:

```
var d = new Date();

console.log(d.getFullYear());
```

Метод getMonth()

Метод **getMonth**() возвращает номер месяца (0-11):

```
var d = new Date();

console.log(d.getMonth());
```

В JavaScript первый месяц (январь) имеет номер 0, таким образом у последнего, декабря, будет номер 11.

Чтобы возвращалось название месяца, можно использовать массив с именами месяцев и метод **getMonth**():

```
var d = new Date();
var months = ["Январь", "Февраль", "Март", "Апрель", "Май", "Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь", "Декабрь"];
console.log(months[d.getMonth()]);
```

Метод getDate()

Метод **getDate**() возвращает число месяца (1-31):

```
var d = new Date();
console.log(d.getDate());
```

Метод getDay()

Метод **getDay**() возвращает номер дня недели (0-6):

```
var d = new Date();
console.log(d.getDay());
```

В JavaScript первым днем недели (0) считается "воскресенье", даже если в стране, где находится пользователь, первым днем недели считается "понедельник".

Чтобы возвращалось название дня недели, можно использовать массив с днями недели и метод **getDay**():

```
var d = new Date();
var days = ["Воскресенье", "Понедельник", "Вторник", "Среда", "Четверг", "Пятница", "Суббота"];
```

```
console.log(d.getDay());
```

example_5. Методы геттеры

```
<script>
    // текстовое название месяцев
    var months = ["Январь", "Февраль", "Март", "Апрель", "Май",
    "Июнь", "Июль", "Август", "Сентябрь", "Октябрь", "Ноябрь",
    "Декабрь"];
    // текстовое название дней недели
    var days = ["Воскресенье", "Понедельник", "Вторник",
    "Среда", "Четверг", "Пятница", "Суббота"];
    // получение строки текущего времени
    let d = new Date();
    // получение отдельных компонентов текущего времени
    let time = d.getTime();
    let fullYear = d.getFullYear();
    let month = months[d.getMonth()];
    let date = d.getDate();
    let day = days[d.getDay()];
    // вывод в консоль компонентов
    console.log(time);
    console.log(fullYear);
    console.log(month);
    console.log(date);
    console.log(day);
    // подготовка форматированной строки для вывода в браузер
    let strDate = `
        Микросекунд: ${time} <p>
```

```
            Год: ${fullYear} <p>
            Месяц: ${month} <p>
            Число: ${date} <p>
            День недели: ${day}
        `;
        // вставка в браузер строки с компонентами времени
        document.getElementById("demo").innerHTML = strDate;
</script>
```

Метод getHours()

Метод **getHours**() возвращает час (0-23):

```
var d = new Date();
console.log(d.getHours());
```

Метод getMinutes()

Метод **getMinutes**() возвращает минуты (0-59):

```
var d = new Date();
console.log(d.getMinutes());
```

Метод getSeconds()

Метод **getSeconds**() возвращает секунды (0-59):

```
var d = new Date();
console.log(d.getSeconds());
```

Метод getMilliseconds()

Метод **getMilliseconds**() возвращает миллисекунды (0-999):

```
var d = new Date();
console.log(d.getMilliseconds());
```

example_6. Получение компонентов даты

```
<script>
    // получаем текущее время
    var d = new Date();
    // получаем компоненты времени
    let hour = d.getHours();
    let min = d.getMinutes();
    let sec = d.getSeconds();
    let msec = d.getMilliseconds();
    // вывод компонентов в консоль
    console.log(hour);
    console.log(min);
    console.log(sec);
    console.log(msec);
    // подготовка строки с компонентами времени
    let strTime = `
        Часы: ${hour} <p>
        Минуты: ${min} <p>
        Секунды: ${sec} <p>
        Милисекунды: ${msec}
    `;
    // вывод строки в браузер
    document.getElementById("demo").innerHTML = strTime;
```

```
</script>
```

Методы сеттеры

Помимо **получения** данных в объекте Date также есть методы, которые позволяют **изменить значения даты и времени** (год, месяц, день, час, минуты, секунды, миллисекунды).

Метод	Описание
setTime()	Устанавливает время (количество миллисекунд, прошедших с 1 января 1970)
setFullYear()	Устанавливает год (также можно установить месяц и день)
setMonth()	Устанавливает месяц (0-11)
setDate()	Устанавливает день (1-31)
setHours()	Устанавливает час (0-23)
setMinutes()	Устанавливает минуты (0-59)
setSeconds()	Устанавливает секунды (0-59)
setMilliseconds()	Устанавливает миллисекунды (0-999)

Общий принцип работы с сеттерами, следующий:

1. **создать** объект Date;
2. на полученном объекте **установить** новый компонент времени.

Метод setTime()

Метод **setTime**() устанавливает время объекта **Date** в значение, представленное количеством **миллисекунд**, прошедших с 1 января 1970 00:00:00 по UTC.

В следующем примере получим количество миллисекунд от объекта заданной даты и установим новую дату по полученным миллисекундам.

```
var d = new Date('July 1, 2000');
```

```
msec = d.getTime();
var newd = new Date(msec);
console.log(d);
console.log(newd);
```

Метод setFullYear()

Метод **setFullYear**() устанавливает в объекте даты год.

В следующем примере создадим объект Date и на полученном объекте установим 2020 год:

```
var d = new Date();
d.setFullYear(2020);
console.log(d);
```

При желании при помощи метода **setFullYear**() также можно установить месяц и день:

```
var d = new Date();
d.setFullYear(2020, 11, 3);
console.log(d);
```

Метод setMonth()

Метод **setMonth**() устанавливает в объекте даты месяц (0-11):

```
var d = new Date();
d.setMonth(11);
console.log(d);
```

Метод setDate()

Метод **setDate**() устанавливает в объекте даты день (1-31):

```
var d = new Date();
d.setDate(20);
console.log(d);
```

Кроме этого, метод **setDate**() может использоваться для добавления дней к дате:

```
var d = new Date();
d.setDate(d.getDate() + 50);
console.log(d);
```

Если при добавлении дней происходит сдвиг месяцев или года, то эти изменения в объекте Date происходят **автоматически**.

example_7. Методы сеттеры

```
<script>
    // новый объект Date с текущими датой и временем
    var d = new Date();
    // на готовом объекте меняем компонент -время
    d.setTime(3600000); // 3600000 - 1 час
    console.log(d);
    // на готовом объекте меняем компонент -год
    d.setFullYear(1971);
    console.log(d);
    // на готовом объекте меняем компонент -месяц
    d.setMonth(3);
    console.log(d);
    // на готовом объекте меняем компонент -день
```

```
    d.setDate(10);

    console.log(d);
</script>
```

Метод setHours()

Метод **setHours**() устанавливает в объекте даты час (0-23):

```
var d = new Date();
d.setHours(22);
console.log(d)
```

Метод setMinutes()

Метод **setMinutes**() устанавливает в объекте даты минуты (0-59):

```
var d = new Date();
d.setMinutes(30);
console.log(d);
```

Метод setSeconds()

Метод **setSeconds**() устанавливает в объекте даты секунды (0-59):

```
var d = new Date();
d.setSeconds(30);
console.log(d);
```

example_8. Установка компонентов даты

```
<script>
    // новый объект Date с текущими датой и временем
```

```
    var d = new Date();
    // текущий объект Date, но 22 часа
    d.setHours(22);
    console.log(d);
    // текущий объект Date, но 30 минут
    d.setMinutes(30);
    console.log(d);
    // текущий объект Date, но 45 секунд
    d.setSeconds(45);
    console.log(d);
</script>
```

Отображение даты

По умолчанию, JavaScript будет отображать дату в формате полной текстовой строки: Wed Mar 25 2015 03:00:00 GMT+0300.

Когда вы выводите объект даты в HTML, он **автоматически преобразуется в строку при помощи метода toString**().

Пример:

```
d = new Date();
console.log(d);
```

То же самое:

```
d = new Date();
console.log(d.toString());
```

Метод **toUTCString**() преобразует дату в строку UTC (стандарт отображения даты).

```
var d = new Date();

console.log(d.toUTCString());
```

Метод **toDateString**() преобразует дату в более читабельный формат:

```
var d = new Date();

console.log(d.toDateString());
```

example_9. Строковое представление даты

```
<script>

     // создание объекта текущей даты

     d = new Date();

     // формируем строку для вывода

     let date = `

          Текущая дата <b>new Date()</b>: ${d} </d> <p>

          Метод <b>toString()</b>: ${d.toString()} <p>

          Метод <b>toUTCString()</b>: ${d.toUTCString()} <p>

          Метод <b>toDateString()</b>: ${d.toDateString()} <p>

     `;

     // вывод строки в браузер

     document.write(date);

</script>
```

Короткая запись даты

Короткая запись даты имеет следующую форму **ММ/ДД/ГГГГ**,

где:

- **ГГГГ** — полный год
- **ММ** — номер месяца
- **ДД** — день.

```
var d = new Date("12/25/2021");

console.log(d);
```

Поведение браузера при формате записи ДД/ММ/ГГГГ и ГГГГ/ММ/ДД **не определено**. Некоторые браузеры попытаются угадать формат. Некоторые вернут значение **NaN**.

example_10. Короткая запись даты

```
<script>

    // ММ/ДД/ГГГГ

    var d = new Date("12/25/2021"); // текущая дата

    console.log(d);

    // ДД/ММ/ГГГГ

    var d = new Date("25/12/2021"); // NaN

    console.log(d);

    // ГГГГ/ММ/ДД

    var d = new Date("2021/12/25"); // текущая дата

    console.log(d);

</script>
```

Альтернативная форма записи даты имеет следующий вид **ММ-ДД-ГГГГ**,

```
var d = new Date("12-25-2021");

console.log(d);
```

Поведение браузера при формате записи ДД-ММ-ГГГГ или ГГГГ-ММ-ДД также **не определено**. Некоторые браузеры попытаются угадать формат. Некоторые вернут значение **NaN**.

example_11. Альтернативная форма записи короткой даты

```
<script>
```

```
    // ММ-ДД-ГГГГ
    var d = new Date("12-25-2021"); // текущая дата
    console.log(d);
    // ДД-ММ-ГГГГ
    var d = new Date("25-12-2021"); // NaN
    console.log(d);
    // ГГГГ-ММ-ДД
    var d = new Date("2021-12-25"); // текущая дата
    console.log(d);
</script>
```

В некоторых браузерах указание месяца и дня **без начального нуля** может привести к **ошибке**.

Длинная запись даты

Длинная запись даты имеет следующий синтаксис **ММММ ДД ГГГГ**:

```
var d = new Date("Mar 25 2015");
```

Месяц и день могут быть в любом порядке:

```
var d = new Date("25 Mar 2015");
```

Месяц может записываться либо полностью (January), либо сокращенно (Jan):

```
var d1 = new Date("January 25 2015");
var d2 = new Date("Jan 25 2015");
```

Запятые игнорируются. Имена регистронезависимы:

```
var d = new Date("JANUARY, 25, 2015");
```

example_12. Длинная запись даты

```
<script>
     var d1 = new Date("January 25 2015");
     var d2 = new Date("Jan 25 2015");
     // запятые игнорируются
     // имена регистронезависимы
     var d3 = new Date("JANUARY, 25, 2015");
     // так не работает
     var d4 = new Date("Январь, 25, 2015");
     console.log(d1);
     console.log(d2);
     console.log(d3);
     console.log(d4);
</script>
```

Парсинг даты

Корректно составленную строку даты можно при помощи метода **Date.parse**() преобразовать в **миллисекунды**. Метод **Date.parse**() возвращает количество миллисекунд, прошедших с 1 января 1970 до заданной даты:

```
var msec = Date.parse("Oct 20, 2012");
console.log(msec);
```

В последствии эти миллисекунды можно преобразовать в объект даты:

```
var msec = Date.parse("March 12, 2018");
var d = new Date(msec);
console.log(d);
```

example_13. Парсинг даты

```
<script>
    // парсим корректную строку времени в миллисекунды
    var msec = Date.parse("March 12 2021");
    console.log(msec);
    // миллисекунды преобразуем в дату
    console.log(d = new Date(msec));
    // дату преобразуем в миллисекунды
    console.log(d.getTime());
</script>
```

Отображение разности дат

Часто появляется необходимость расчета разности дат.

После определения разности в миллисекундах, вы можете получить количество секунд, разделив миллисекунды на 1000, а затем преобразовать результат в целое число, это удаляет дробную часть, представляющую миллисекунды.

```
var a = new Date(2023, 1); // дата 1
var b = new Date(2022, 1); // дата 2
var d = (a-b); // разность в миллисекундах
var seconds = parseInt(d/1000);
```

Затем можно получить минуты (minutes), разделив seconds на 60 и преобразовав его в целое число.

```
var minutes = parseInt(seconds/60);
```

Разделив minutes на 60 и преобразовав результат в целое число получаем часы, hours.

```
var hours = parseInt(minutes/60);
```

Ну, и получить количество дней можно разделив hours на 24.

```
var days = parseInt(hours/24);
```

Например, чтобы получить **количество дней** в заданном количестве миллисекунд, необходимо разделить на 86 400 000 (количество миллисекунд в дне: 1000 х 60 секунд х 60 минут х 24 часа).

example_14. Подсчет и вывод разности дат

```
<script>
	// первая метка времени
	var a = new Date(2023, 0);
	// вторая метка времени
	var b = new Date(2022, 0);
	// вычисляем разность в миллисекундах
	var dif = a.getTime() - b.getTime();
	console.log("Расчетное время в миллисекундах: ", dif);
	// разность меток  можно вычислить и так
	// var dif = a - b;
	// console.log(dif);
	// всего дней
	console.log("Всего дней: ", dif / (1000 * 60 * 60 * 24));
	// всего часов
	console.log("Всего часов: ", dif / (1000 * 60 * 60));
	// всего минут
	console.log("Всего минут: ", dif / (1000 * 60));
	// всего секунд
	console.log("Всего секунд: ", dif / (1000));
</script>
```

Автоматическая корректировка дат

JavaScript имеет очень интересную и полезную особенность: если при создании объекта Date был указан **некорректный** момент времени - он **автоматически будет пересчитан в корректный**.

Давайте посмотрим это на примере (господа, я сейчас все объясню...).

Как известно, Барон Мюнхгаузен много сделал для своего родного города, например, дал городу еще один весенний день - тридцать второе мая (ну не идиоты же мы, чтобы отказываться от лишнего дня в году). Как мы раньше наивно полагали, даты 32 мая не существует. Максимально возможный день мая - 31. Получается, что новая дата имеет лишний день. Или нет?

Попробуем проверить утверждение барона с помощью JS-объекта Date.

example_15. Автоматическая корректировка Мюнхгаузена

```
<script>

    // зададим дату - 32 мая

    let date = new Date(2018, 4, 32);

    // браузер исправил на первое июня

    // Fri Jun 01 2018 00:00:00 GMT+0300 (Москва, стандартное
    время)

    console.log(date);

</script>
```

JavaScript просто прибавил лишний день к следующему месяцу.

Можно указывать не только **лишние** дни, но и месяцы. При этом **следует помнить, что месяцы начинаются с нуля**, а значит последний корректный месяц - 11-тый. Если указать 12-тый месяц, то получится январь следующего года:

```
let date = new Date(2018, 12, 1); // указываем 12-тый месяц
```

```
console.log(date); // получим 1 января 2019 года
```

Описанная корректировка работает и в **меньшую сторону**. Самым **минимальным днем месяца является день с номером 1**. Поэтому, если указать день с номером 0, то получится последний день предыдущего месяца:

```
let date = new Date(2018, 1, 0); // указываем нулевой день
console.log(date); // получим 31 января
```

Минимальный месяц имеет номер 0. Это значит, что минус первый день попадает во 2 день с конца предыдущего месяца, а минус первый месяц попадает просто в последний месяц предыдущего года.

example_16. Автоматическая корректировка отрицательных значений

```
<script>
    // определим дату с отрицательным числом
    var dat = new Date(2022, 12, -5);
    // браузер додумывает за пользователя,
    // что тот имел ввиду
    // часто успешно
    console.log(dat);
</script>
```

Задание № 19

Напишите скрипт, который при загрузке HTML-страницы, выводит в браузер текущий **день** недели, **число**, **месяц** и **год**. Для вывода дня недели предусмотреть текстовый **массив**.

Примерный **формат** вывода:

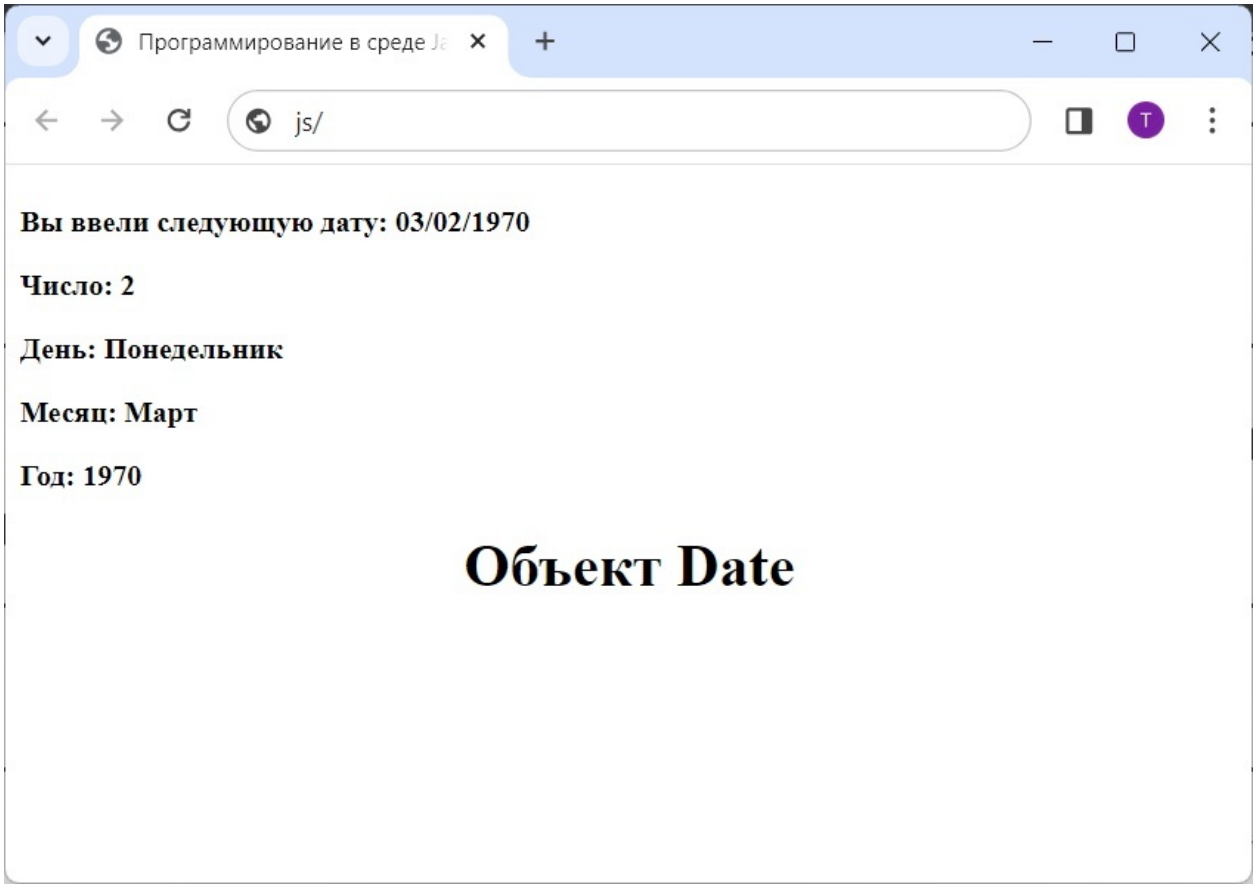
```
Сегодня: 21-10-2022 (Понедельник)
```

Задание № 20

Напишите скрипт, который при загрузке HTML-страницы **запрашивает** дату вашего рождения и выводит на страницу **день** недели, **число**, **месяц** и **год** этой даты.

Для вывода месяца и дня недели организовать текстовые массивы.

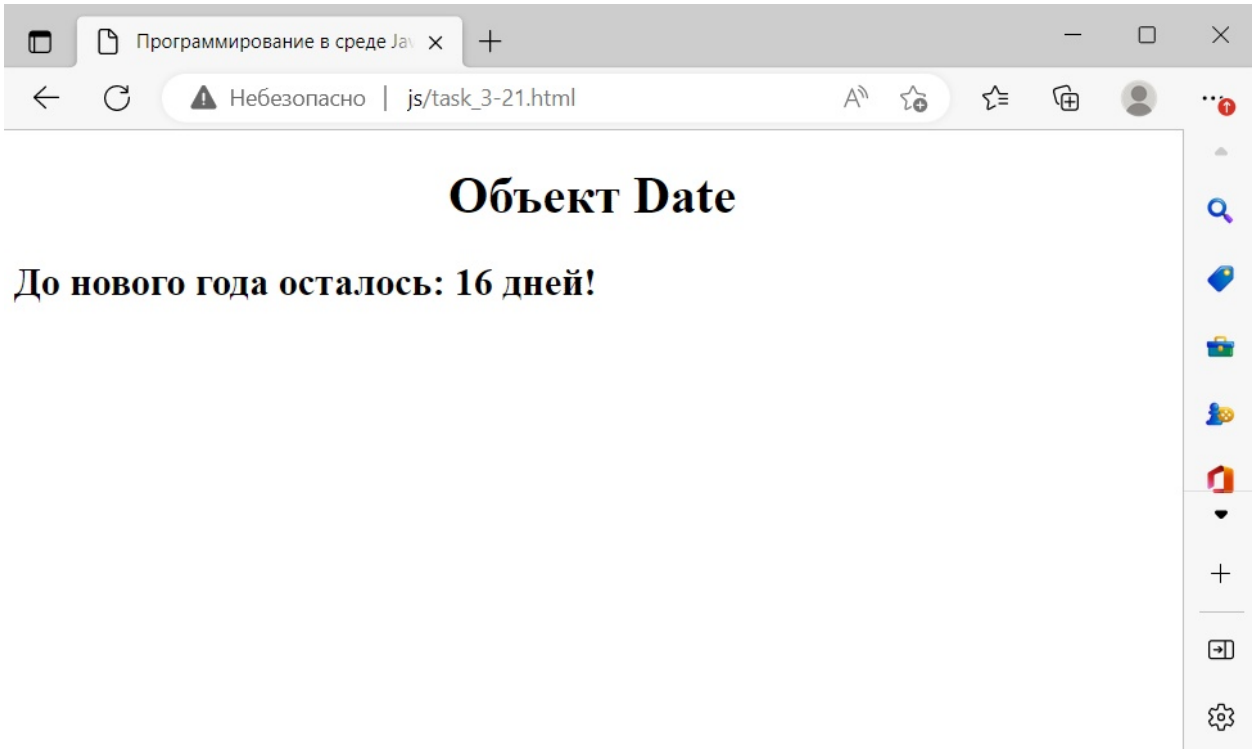
Примерный **формат** вывода компонентов даты представлен на рисунке.



Задание № 21

Напишите скрипт, подсчитывающий сколько дней осталось до ближайшего Нового года. **Выведите** расчетное значение в предназначенный блок HTML-страницы.

Примерный **формат** вывода представлен на рисунке.



Задание № 22

Учитывая возможности **автоматической корректировки даты**, определите, **в какую дату** JavaScript преобразует следующий момент времени:

1)

```
let date = new Date(2023, 11, 35);

console.log(date); // ?
```

2)

```
let date = new Date(2023, -3, 5);

console.log(date); // ?
```

3)

```
let date = new Date(2023, -3, -10);

console.log(date); // ?
```

Напишите **скрипт**, **проверьте** расчетные данные.