

**JavaScript**

[JavaScript 2](#_Toc525326190)

[1 Как работает JavaScript? 2](#_Toc525326191)

[2 Java – это короткое название JavaScript? 2](#_Toc525326192)

[3 В чем писать JavaScript? 2](#_Toc525326193)

[4 Командная строка 2](#_Toc525326194)

[5 HTML-страница для примера 3](#_Toc525326195)

[6 Ввод-вывод данных 3](#_Toc525326196)

[6.1 Блоки кода 3](#_Toc525326197)

[6.2 Всплывающие сообщения 4](#_Toc525326198)

[6.3 Переменные 5](#_Toc525326199)

[6.4 Проверка условий в if 6](#_Toc525326200)

[6.5 Вывод на страницу document.write 7](#_Toc525326201)

[6.6 Задание 8](#_Toc525326202)

[7 Калькулятор НДС 8](#_Toc525326203)

[7.1 Элементы управления в HTML 9](#_Toc525326204)

[7.2 События. Функции 10](#_Toc525326205)

[7.3 Вывод в innerHTML 11](#_Toc525326206)

[7.4 Преобразование типов 13](#_Toc525326207)

[7.5 Элемент управления переключатель 16](#_Toc525326208)

[7.6 Задание 18](#_Toc525326209)

[8 Часы 18](#_Toc525326210)

[8.1 Скрипты во внешнем файле 19](#_Toc525326211)

[8.2 Работа с датой и временем 19](#_Toc525326212)

[8.3 Срабатывание по таймеру 21](#_Toc525326213)

[8.4 Задание 21](#_Toc525326214)

# JavaScript

## Как работает JavaScript?

Код JavaScript встраивается в web-страницу вместе с HTML и CSS. Код выполняет браузер непосредственно во время загрузки страницы. JavaScript может выполнять любые действия в текущем окне браузера, но не имеет доступа ни к системе за пределами этого окна, ни к другим окнам и страницам в браузере.

## Java – это короткое название JavaScript?

Нет. Это вообще разные языки. Внешне в коде могут быть сходства, т.к. оба языка C-подобные. Но и исполнение, и прочие особенности у них отличаются.

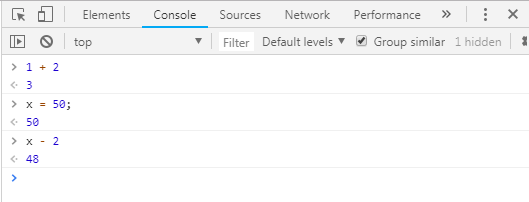
Если JavaScript слишком длинно, пишите **JS**.

## В чем писать JavaScript?

Там же, где HTML и CSS.

## Командная строка

В инструментах разработчика в браузере есть вкладка Console (F12 или Shift+Ctrl+I). Сюда выводятся сообщения об ошибках, и в нее можно писать любые команды JS.



## HTML-страница для примера

Поскольку JavaScript работает на страницах сайта, нам потребуется простенькая страничка:

<!DOCTYPE HTML>

<html lang="ru">

<head>

<meta charset="utf-8">

<title>My JavaScript </title>

</head>

<body>

<h1>JavaScript</h1>

<p>Начало документа...</p>

<p>...Конец документа</p>

</body>

</html>

Сохраните запасной экземпляр этого файла, он вам еще пригодится.

## Ввод-вывод данных

Пора создать первый JS-код. Обычно его выносят в отдельный файл, но в простых случаях можно разместить прямо в тексте странички.

### Блоки кода

Добавьте в конец блока <head> пару тегов <script></script>. Не забывайте про отступы. Кстати, программы на JS обычно называют ***скриптами***.

<**script**>

//здесь пишется программный код

</**script**>

Таких блоков на странице может быть сколько угодно и размещаться они могут где угодно, но чаще всего их помещают либо в head, либо в самый конец файла. Недостаток – синтаксис JavaScript не будет подсвечиваться, т.к. это файл HTML.

Раньше, когда JS еще не был так популярен, обязательным требованием было указать язык скрипта в атрибуте language:

<script **language** = "JavaScript">

//здесь пишется программный код

</script>

Но сейчас это необязательно.

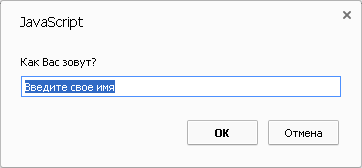
### Всплывающие сообщения

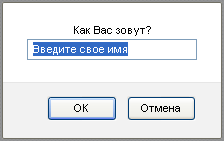
Простейший способ ввода-вывода – это всплывающие сообщения. Они появляются при загрузке страницы.

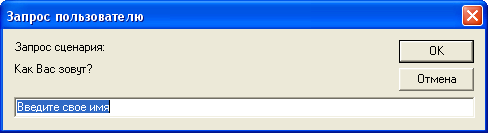
**prompt**(*текст\_сообщения*[,*значение\_по\_умолчанию*]);

(Не теряйте букву **p** в слове prom**p**t! PROMT – это российская программа-переводчик, а prompt по-английски «запрашивать»).

Его вид немного отличается в разных браузерах:

  
Google Chrome

  
Firefox

  
Internet Explorer

Вывод окна сообщения осуществляется через функцию alert:

**alert**(*текст\_сообщения*);

Сообщение alert выглядит аналогично, только без поля ввода.

Напишем простейшую программу. Запросим у пользователя его имя и выведем приветствие.

Начнем с вывода приветствия без имени. Текст пишется в кавычках (двойных или одинарных – не важно).

<script>

**alert**("Добро пожаловать!");

</script>

Запустите страницу в браузере и убедитесь, что сообщение появляется и после нажатия OK отображается ваша страничка.

### Переменные

Имя пользователя будем хранить в переменной userName. Переменные в JS указываются с ключевым словом var, тип не указывается (JS определяет тип данных автоматически, по значению). В конце команды не забываем ставить точку с запятой.

<script>

**var** userName;

alert("Добро пожаловать!");

</script>

Добавим запрос имени пользователя и его вывод в alert. Причем имя пользователя нужно вставить между «Добро пожаловать» и восклицательным знаком:

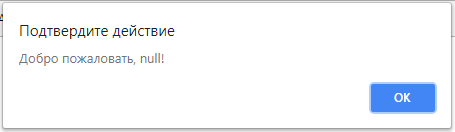
<script>

**var** userName = prompt("Привет! Как вас зовут?");

alert("Добро пожаловать, " + userName + "!");

</script>

Проверьте, как работает ваш скрипт. А что будет, если при вводе имени нажать «Отмена»?



Как-то обидно для пользователя. А если ничего не вводить и нажать OK?



Тоже не очень хорошо. Давайте добавим проверку, и если пользователь ничего не ввел, то покажем «Добро пожаловать, гость!»

### Проверка условий в if

Проверка условий, как и в других языках, выполняется в операторе if. Сначала добавим проверку для второго случая, он попроще. Здесь пользователь нажал OK и тем самым ввел пустую строку. Проверка на равенство состоит из двух = подряд, без пробела.

<script>

**var** userName = prompt("Привет! Как вас зовут?");

//если имя пользователя не указано, то называем его гостем

**if** (userName == "") {

alert("Добро пожаловать, гость!");

}

//иначе - приветствуем по имени

**else** {

alert("Добро пожаловать, " + userName + "!");

};

</script>

В первом случае мы обозвали дорогого пользователя null. null – это особая константа, означающая полное отсутствие значения (буквально ничего). null не равен ничему, кроме самого себя. Добавим в if сравнение с null. Если имя пользователя равно пустой строке **ИЛИ** null, то он просто гость. **ИЛИ** в JS обозначается двумя вертикальными чертами **||**.

//если имя пользователя не указано, то называем его гостем

**if** (userName == "" || userName == null) {

alert("Добро пожаловать, гость!");

}

//иначе - приветствуем по имени

**else** {

alert("Добро пожаловать, " + userName + "!");

};

Проверьте работу скрипта. Теперь все нормально?

Но здесь есть еще место для улучшения. Этого изменения пользователь не заметит, но нам надо думать еще и о дальнейшем развитии программы, и о других программистах.

Сейчас мы просто приветствуем пользователя. А если мы захотим еще где-то к нему обратиться? Опять писать длинный if? Каждый раз?

Давайте лучше запишем в userName, что его зовут «гость» и будем пользоваться где угодно безо всяких проверок.

//если имя пользователя не указано, то называем его гостем

**if** (userName == "" || userName == null) {

userName = "гость";

};

//приветствуем по имени

alert("Добро пожаловать, " + userName + "!");

Теперь блок else оказался не нужен – ввел пользователь имя, или не ввел, форма приветствия будет одинаковой.

### Вывод на страницу document.write

И, чтобы далеко не ходить, добавим вывод имени пользователя не только в виде сообщения, но и на страничку.

Самый простой (но как вы поймете на практике, не самый удобный) способ написать что-то на страницу – это использовать **document.write**(*текст\_на\_страницу*). В тексте можно использовать теги HTML, браузер их обработает. Причем выводиться будет строго в том месте странички, где написан document.write. Напомню, что наш скрипт находится в блоке head, который на странице не отображается.

Поэтому добавим еще тег script, но уже в теле страницы. Например, в начале.

<script>

**document.write**("<p>Уважаемый, " + userName + ", эта страница находится в разработке.</p>");

</script>

Убедитесь, что скрипт работает, и мы дважды обращаемся к пользователю: в alert и на странице.

**Мораль**: значения переменных видны всем блокам скриптов на странице, как будто это один сплошной код.

### Задание

Самостоятельно добавьте запрос пола пользователя и выводите на странице «уважаемый» или «уважаемая» соответственно.

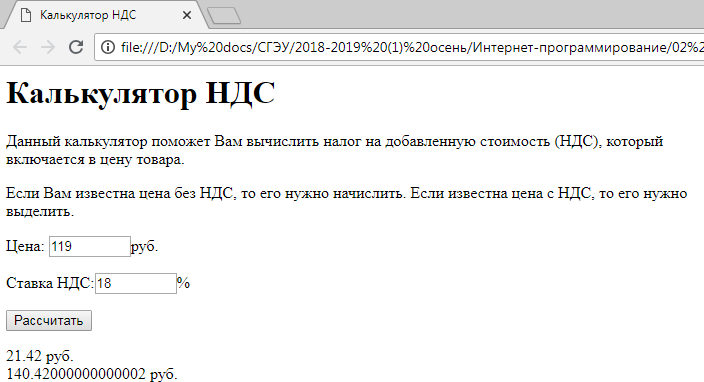
## Калькулятор НДС

Следующий этап – работа с числами, размещение кнопок и полей ввода на странице.

Самостоятельно создайте страницу с заголовком «Калькулятор НДС» и текстом:

Данный калькулятор поможет Вам вычислить налог на добавленную стоимость (НДС), который включается в цену товара.

Если Вам известна цена без НДС, то его нужно начислить. Если известна цена с НДС, то его нужно выделить.



Стили страницы настройте по своему усмотрению.

Если вы не знаете, как вычисляется НДС – спросите у Гугла или Яндекса.

### Элементы управления в HTML

В этот раз мы не будем пугать пользователя всякими окошками, а разместим все прямо на странице.

Для элементов управления в HTML используется непарный тег <input>. Тип элемента управления определяется атрибутом type.

Поле для ввода:

<**input** type="**text**" id="имя" size="ширина" maxlength="макс\_длина\_текста" value="значение">

Кнопка:

<**input** type="**button**" value="текст\_на\_кнопке">

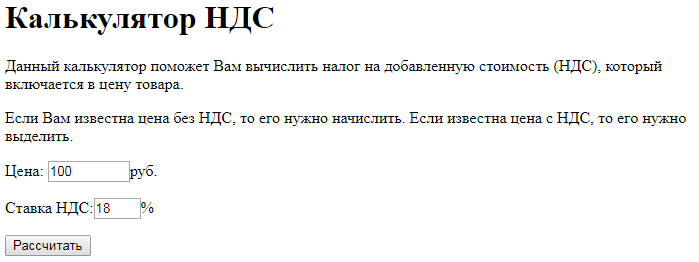
К input как и к любым другим тегам можно применять стили CSS.

Разместим на странице два поля для ввода цены и ставки НДС (по умолчанию 18%) и кнопку для вычисления.

<p>Цена: <input type="text" id="textPrice" value=100 size=7>руб.</p>

<p>Ставка НДС:<input type="text" id="textTaxRate" value=18 size=2>%</p>

<p><input type="button" value="Рассчитать"></p>



Убедитесь, что страница отображается корректно. Кнопка пока не работает, ведь мы не привязали к ней никакого действия.

### События. Функции

Всё, что происходит на странице – нажатие на кнопку, движение мышкой, загрузка, прокрутка и т.п. – является **событиями**. События могут относиться ко всей странице целиком или к какому-то конкретному элементу.

Нас интересует событие onClick – нажатие на кнопку «Рассчитать». При этом должен запуститься код JavaScript для расчета. Чтобы браузер смог найти нужный код, у него должно быть какое-то имя. Блок кода, который имеет свое имя, образует **функцию**.

Во-первых, добавим заготовку скрипта с функцией calcuateVAT() (вычисли НДС, VAT = value added tax). Функция объявляется ключевым словом function, далее пишется имя функции, круглые скобки (в них могут быть аргументы) и в фигурных скобках – тело функции.

<script>

**function** calculateVAT() {

}

</script>

Теперь нужно подключить эту функцию к событию onclick найшей кнопки.

<p><input type="button" value="Рассчитать" **onclick="calculateVAT()"**></p>

В самой функции сначала необходимо получить значения (value), которые пользователь ввел в текстовые поля textPrice и textTaxRate.

**function** calculateVAT() {

**var** price = textPrice.value; //цена без НДС

**var** taxRate = textTaxRate.value; //ставка налога

}

Вычисляем НДС и цену с НДС.

**function** calculateVAT() {

**var** price = textPrice.value; //цена без НДС

**var** taxRate = textTaxRate.value; //ставка налога

**var** VAT = price \* taxRate / 100; //НДС = цена без НДС \* ставка НДС / 100%

**var** priceVAT = price + VAT; //цена с НДС = цена без НДС + НДС

}

Осталось вывести результат на страницу. Но document.write() здесь не подойдет – он не умеет очищать страницу и после каждого нажатия на кнопку будет дописывать новый результат в конец, оставляя старые на месте. Нам же нужен один текущий результат.

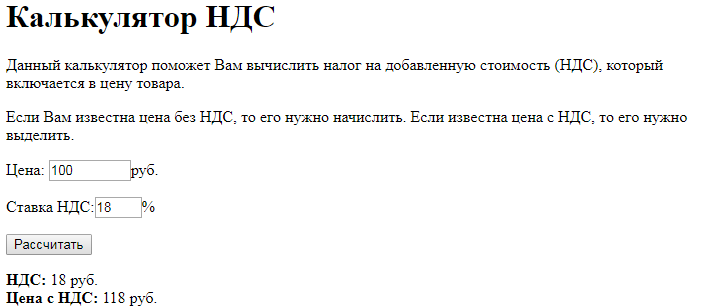
### Вывод в innerHTML

К счастью, JavaScript может обратиться к любому элементу на странице по его id. Добавим на страницу два тега div после кнопки «Рассчитать»: один для вывода НДС, другой для вывода цены с НДС.

<**div** id="divVAT"><b>НДС:</b> 18 руб.</div>

<**div** id="divPriceVAT"><b>Цена с НДС:</b> 118 руб.</div>

Как видите, по id этих элементов понятно, который за что отвечает, и то, что это именно теги div, а не просто переменные.



Добавляем в функцию вывод результатов расчета в эти div. Все, что находится между открывающим и закрывающим тегами, называется innerHTML. Сюда можно выводить любой текст и теги HTML.

**function** calculateVAT() {

**var** price = textPrice.value; //цена без НДС

**var** taxRate = textTaxRate.value; //ставка налога

**var** VAT = price \* taxRate / 100; //НДС = цена без НДС \* ставка НДС / 100%

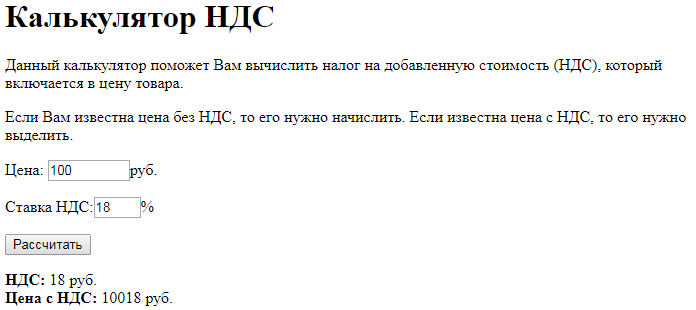
**var** priceVAT = price + VAT; //цена с НДС = цена без НДС + НДС

divVAT.innerHTML = "<b>НДС:</b> " + VAT + " руб.";

divPriceVAT.innerHTML = "<b>Цена с НДС:</b> " + priceVAT + " руб.";

}

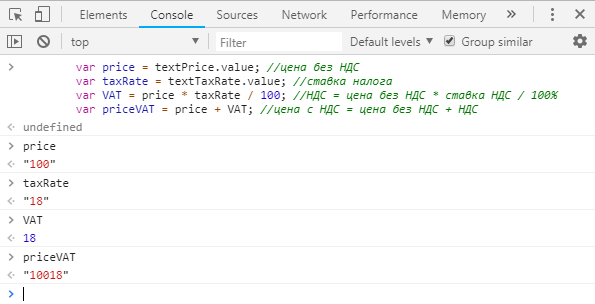
Проверим результат:



Видим, что НДС вычислен правильно: 18% от 100 руб. = 18 руб. А вот с ценой с НДС получилось что-то не то: 100 + 18 = 118, а не 10018!

Проблема в том, что JavaScript – язык с динамической типизацией. Мы не указывали тип данных для переменных, и браузер пытается сам «угадать», какие значения мы помещаем в переменные.

Давайте откроем консоль браузера (F12) и проверим работу нашей функции. Просто скопируйте в консоль текст из функции и выполните код (Enter). Затем наберите имена переменных, чтобы вывести их значения (автозавершение подсказок стрелкой вправо на клавиатуре).



### Преобразование типов

Что мы видим? В переменные price и taxRate помещены значения "100" и "18", в кавычках – это текст. Логично, ведь мы взяли их из текстовых полей.

Но почему тогда в VAT получилось число 18? Да потому, что в формуле price \* taxRate / 100 анализатор JavaScript распознал арифметические действия \* / и число 100 и, таким образом, «понял», что нужно текст "100" и "18" преобразовать в числа 100 и 18, а потом уже считать. Поэтому тут никаких проблем.

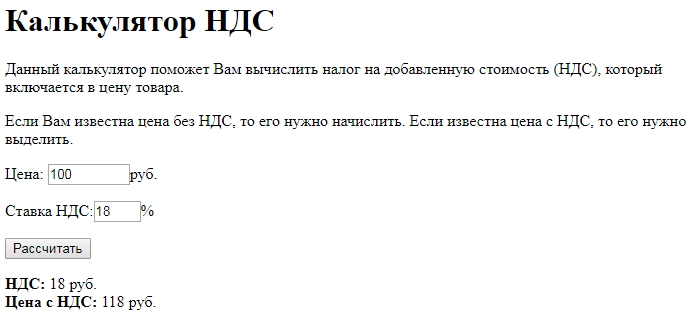
Но почему тогда не получилось вычислить priceVAT? Дело в том, что плюс в price + VAT – операция неоднозначная. Плюс может означать и сложение чисел, и конкатенацию («склеивание») строк. Поскольку переменная price является текстом, а VAT – числом, то JS решает, что это конкатенация, преобразует VAT в строку "18" и склеивает "100" и "18".

Лучше всего сразу заставить JS понять, что price и taxRate – это числа, а не текст. Тогда в дальнейшем все расчеты с ними будут правильными. Сделать это очень просто – поставьте + перед текстовыми значениями. Это унарная математическая операция, противоположность знака – (например, +(2 + 3) = 5, а -(2 + 3) = -5). Унарный плюс, в отличие от предыдущего, бинарного, применим только к числам, поэтому текст преобразуется в число.

**var** price = **+**textPrice.value; //цена без НДС

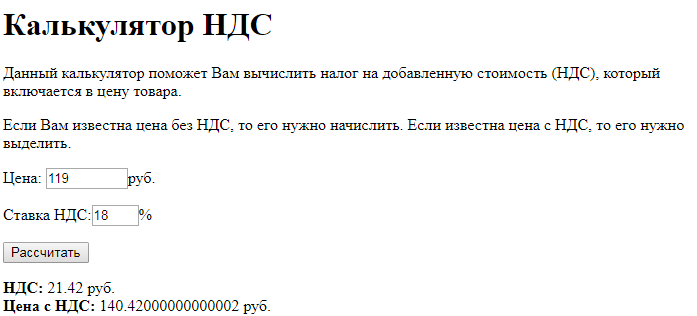
**var** taxRate = **+**textTaxRate.value; //ставка налога

Проверяем, теперь все верно:

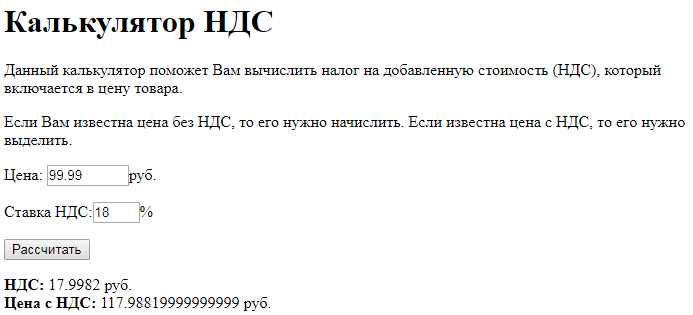


Вы все еще думаете, что это здорово – когда не надо указывать тип данных переменной?

Тогда попробуйте ввести цену 119 руб.



А если 99,99 руб.? (В JavaScript дробная часть отделяется точкой).



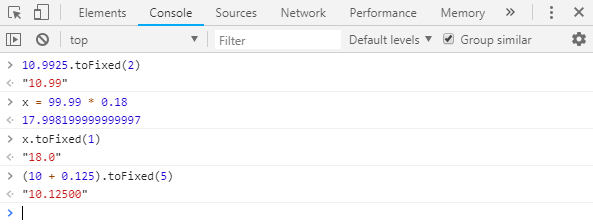
Знакомьтесь, это ошибки округления, связанные с двоичным представлением дробей. Приготовьте 1,191111111111 копеек сдачи.

Запомните как молитву: ***при работе с денежными суммами обязательно округление*** (либо использование специальных типов данных, таких как Decimal и Currency – но в JS их нет).

Округление выполняется математической функцией Math.round(*число*). Но округлять можно только до целых, а у нас есть 2 знака после запятой. Можно, конечно, перевести рубли в копейки, округлить, а потом опять поделить на 100, но получается длинно и непонятно:

**var** VAT = Math.round(price \* 100 \* taxRate / 100) / 100;

К счастью, есть еще метод toFixed, который переводит число в строку с заданным количеством знаков после запятой. Этот метод вызывается непосредственно для числа, переменной или формулы. Например, попробуйте в командной строке:

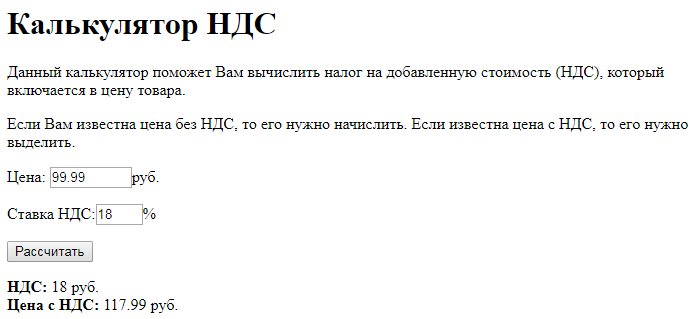


**var** VAT = +(price \* taxRate / 100).**toFixed**(2); //НДС = цена без НДС \* ставка НДС / 100%

**var** priceVAT = +(price + VAT).**toFixed**(2); //цена с НДС = цена без НДС + НДС

Примечание: чтобы переменные VAT и priceVAT остались числовыми, пришлось добавить еще + перед формулой.

Теперь расчеты корректные:



### Элемент управления переключатель

В описании к нашему калькулятору мы указали, что НДС можно вычислять в две стороны. Мы пока реализовали только вычисление по цене без НДС. На практике чаще требуется, наоборот, из цены, в которую уже включен НДС, выделить сам НДС и исходную цену товара.

Добавим в наш калькулятор выбор направления расчета. Лучше всего для этого подойдет переключатель radio.

Кроме идентификаторов id элементам управления можно назначать имена name. Идентификаторы разных элементов не должны совпадать, а имена могут быть одинаковыми. В частности, у radio, которые отвечают на один вопрос, имена должны совпадать. Еще одно свойство checked указывает, что данный переключатель отмечен.

<p><input type="radio" id="radioDirection0" name="radioDirection" checked>Начислить НДС</p>

<p><input type="radio" id="radioDirection1" name="radioDirection">Выделить НДС</p>



Таким образом, переключатель «Начислить НДС» имеет идентификатор radioDirection0, переключатель «Выделить НДС» – radioDirection1, и оба они вместе называются radioDirection.

Для проверки, установлен ли конкретный переключатель, в коде JS можно использовать свойство checked.

**function** calculateVAT() {

**var** price = +textPrice.value; //цена без НДС

**var** taxRate = +textTaxRate.value; //ставка налога

//начисление НДС

**if** (radioDirection0.checked) {

**var** VAT = +(price \* taxRate / 100).toFixed(2); //НДС = цена без НДС \* ставка НДС / 100%

**var** priceVAT = +(price + VAT).toFixed(2); //цена с НДС = цена без НДС + НДС

divPriceVAT.innerHTML = "<b>Цена с НДС:</b> " + priceVAT + " руб.";

}

//выделение НДС

**else** {

**var** VAT = +(price / (100 + taxRate) \* taxRate).toFixed(2); //НДС = цена c НДС / (100% + ставка НДС) \* ставка НДС

**var** priceVAT = +(price - VAT).toFixed(2); //цена без НДС = цена с НДС - НДС

divPriceVAT.innerHTML = "<b>Цена без НДС:</b> " + priceVAT + " руб.";

}

divVAT.innerHTML = "<b>НДС:</b> " + VAT + " руб.";

}



### Задание

Самостоятельно добавьте проверку правильности ввода данных в поля.

## Часы

Разместим на нашей странице цифровые часы, которые будут показывать текущую дату и время.

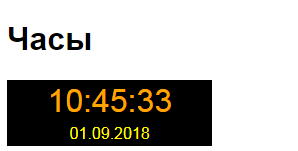
В этом скрипте не будет никаких элементов управления. Часы просто будут выводиться на страницу.

Создайте пустую страницу HTML. Разместите на ней заголовок «Часы» и два элемента div: один для вывода времени, другой для вывода даты.

<div id="divTime">10:45:33</div>

<div id="divDate">01.09.2018</div>

Настройте стили, чтобы часы выглядели примерно так:



### Скрипты во внешнем файле

В этот раз вынесем код JavaScript, как положено, в отдельный файл. Создайте в одной папке с вашей страницей файл «**time.js**». Обратите внимание на расширение **.js**.

В конец HTML-файла страницы добавьте ссылку на ваш файл.

<script src="time.js"></script>

Все, теперь код в файле «**time.js**» доступен вашей странице, как будто он размещен прямо в ней.

### Работа с датой и временем

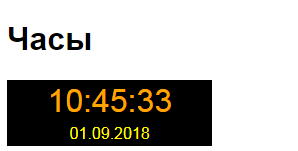
В этом скрипте не будет никаких элементов управления. Часы просто будут выводиться на страницу.

Создайте пустую страницу HTML. Разместите на ней заголовок «Часы» и два элемента div: один для вывода времени, другой для вывода даты.

<div id="divTime">10:45:33</div>

<div id="divDate">01.09.2018</div>

Настройте стили, чтобы часы выглядели примерно так:



Добавим в файл «**time.js**» код, который загрузит текущие дату и время в наши блоки div.

Дата и время Date в JavaScript – это особая структура, т.е. сложный объект, состоящий из нескольких значений и методов обработки. Чтобы поместить такую структуру в переменную, нужно создать ее новый экземпляр с ключевым словом new.

**var** d = **new** Date();

Теперь в переменной d находятся текущие дата и время, с которыми можно выполнять множество действий, например:

* getYear() – узнать год (за 0 считается 1900 год);
* getMonth() - номер месяца (0 – янв; 1 – фев, ...);
* getDate() - число;
* getDay() - номер дня недели (0 – вс, 1 – пн, ...);
* getHours() - часы;
* getMinutes() - минуты;
* getSeconds() - секунды.

Например, можно вывести время так:

divTime.innerHTML = d.getHours() + ':' + d.getMinutes() + ':' + d.getSeconds();

Но нас интересует дата и время целиком. Причем, хотелось бы вывести их в принятом у нас формате (например, в США используют 12-часовой формат времени и формат MM/DD/YYYY для даты, что для нас совсем непривычно).

Для этого у Date тоже есть специальные методы. И будет гораздо короче:

divTime.innerHTML = d.toLocaleTimeString();

divDate.innerHTML = d.toLocaleDateString();

Запустите страницу, и убедитесь, что теперь на нее выводится текущее время и дата.

### Срабатывание по таймеру

Однако, пока что наши часы не «тикают», а просто отображают время, когда страница была загружена.

Чтобы обновлять наши часы, требуется «завернуть» написанный код JS в функцию (назовем ее showDateTime())

function showDateTime()

{

**var** d = **new** Date();

divTime.innerHTML = d.toLocaleTimeString();

divDate.innerHTML = d.toLocaleDateString();

}

Однако функция не будет срабатывать автоматически. Нужно вызвать ее при загрузке страницы – по событию onload тега body.

<body onload="showDateTime()">

Теперь добавим в конец функции отложенный вызов ее самой с помощью команды setTimeout(имя функции, задержка). Обратите внимание, в аргументах setTimeout имя функции указывается в виде текста, а задержка задается в миллисекундах (1000 мс = 1 с).

function showDateTime()

{

**var** d = **new** Date();

divTime.innerHTML = d.toLocaleTimeString();

divDate.innerHTML = d.toLocaleDateString();

setTimeout("showDateTime()", 1000);

}

Все, теперь часы должны тикать.

### Задание

В зависимости от времени суток, выводите на страницу приветствие «Доброе утро!», «Добрый день!», «Добрый вечер!», «Доброй ночи!».

Сделайте в CSS два набора стилей для блока div с часами: дневной и ночной скин. Переключайте их в зависимости от времени суток.

## Бегущая строка

Создадим строку текста, которая будет перемещаться вдоль нижнего края экрана. Для этого нам потребуется совместное использование CSS и JavaScript.

### Подготовка страницы

Бегущую строку мы разместим в двух вложенных тегах div: один будет служить фоном, а другой текстом.

Создайте страницу со следующим содержимым:

<BODY>

<h1>JavaScript. Задание 4</h1>

<div class=scroll\_back id=myscrollback

<div class="scroll" id="myscroll">Бегущая строка</div>

</div>

<BODY>

и таблицу CSS для нее:

.scroll

{

position: absolute;

left: 0px;

top: 0px;

font-family: Arial;

font-size: 10pt;

color: yellow;

white-space: nowrap;

}

.scroll\_back

{

position: absolute;

left: 0px;

height: 20px;

width: 100%;

background-color: black;

overflow: hidden;

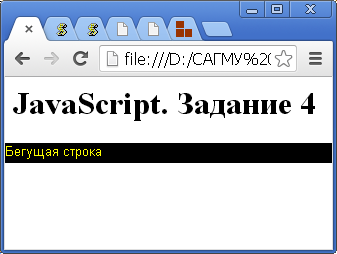
}

Обратите внимание на свойства position: absolute, white-space: nowrap и overflow: hidden.

position: absolute означает, что мы будем задавать положение наших элементов в абсолютных единицах: класс .scroll\_back относительно всего окна, а .scroll относительно scroll\_back.

white-space: nowrap показывает, что текст не нужно переносить, даже если он не умещается в одну строку.

overflow: hidden значит, что не умещающийся текст следует обрезать.



### Код JavaScript

Пока что наша бегущая строка не двигается и располагается вверху страницы. Размещение текста будет выполнять JavaScript. Самостоятельно подключите файл JS к странице.

Сначала добавим функцию для организации прокрутки текста.

var pos = 0;//глобальная переменная для хранения текущего положения начала строки

function ScrollText()

{

//если pos не вылезает за границы myscrollback

if (pos < myscrollback.offsetWidth)

{

pos++ //увеличиваем pos на 1

}

else

{

//иначе – перемещаем строку в начало, за левый край страницы

pos = -myscroll.offsetWidth

}

//собственно перемещение текста

myscroll.style.left = pos + "px";

//перезапуск каждые 20мс (50fps)

setTimeout("ScrollText()",20);

}

Обратите внимание – ширину элемента можно узнать с помощью свойства offsetWidth, а высоту, соответственно через offsetHeight.

Добавим функцию, которая «прилепит» бегущую строку к нижнему краю листа:

function BottomText()

{

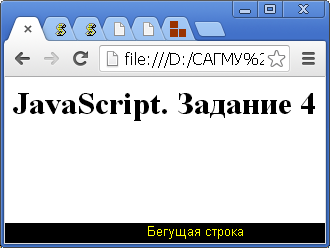
myscrollback.style.top = (window.innerHeight - 20) + "px"

}

Обе их будем вызывать по onload:

<BODY onload="ScrollText(); BottomText()">

Теперь бегущая строка работает.



Но если вы попытаетесь изменить размер окна браузера, то заметите, что бегущая строка немного «не успевает» за окном.

Чтобы этого избежать добавим вызов BottomText еще и при изменении размера окна:

<BODY onload="ScrollText(); BottomText()" onresize="BottomText()">

Замечание: это учебный пример. Для бегущей строки существует специальный тег <marquee>. Но его возможности более ограничены.

### Задание

Самостоятельно измените скрипт так, чтобы цвет бегущей строки менялся после каждого прокручивания.

Измените скрипт так, чтобы добежав до правого края, строка начинала движение в обратную сторону.

## Галерея

Создадим галерею, в которой будут отображаться уменьшенные копии картинок, причем при клике на картинку она отобразится в полном размере в отдельном поле, а при наведении мыши – немного увеличится.

### Подготовка страницы

На странице разместим уменьшенные копии рисунков в виде горизонтальной таблицы и под таблицей – текущую картинку.



Исходный код страницы:

<h1>JavaScript. Задание 5</h1>

<p>Галерея:</p>

<table height=120>

<tr valign=center>

<td><img src="15-1461.jpg" height=70>

</td>

<td><img src="018052.jpg" height=70>

<td>

<td><img src="69049.jpg" height=70>

</td>

<td><img src="144066.jpg" height=70>

</td>

<td><img src="454044.jpg" height=70>

</td>

</tr>

</table>

<p align=center><img id=big\_pic src="15-1461.jpg"></p>

Все картинки в галерее отображаются с одинаковой высотой (70), независимо от реального размера. В big\_pic будет отображаться картинка в реальном размере.

*Замечание*: для экономии трафика и ускорения загрузки страницы правильнее хранить уменьшенные картинки в виде отдельных файлов. А уже когда пользователь нажал на картинку, загружать ее полный вариант.

### Обработка событий

Для выполнения задания нам потребуется три функции:

* для загрузки большой картинки (onClick).
* для увеличения картинки в галерее при наведении мыши (onMouseEnter).
* для обратного уменьшения картинки в галерее, когда мышь убрана (onMouseLeave).

Имя загружаемой картинки будем передавать в функцию в виде параметра:

function ShowPic(filename)

{

big\_pic.src = filename

}

Функцию будем вызывать по событию onclick маленьких картинок.

onclick="ShowPic('15-1461.jpg')"

Текстовый параметр передаем в одинарных кавычках.

Для уменьшения и увеличения картинки нам нужно знать, какую картинку увеличивать. Саму картинку также можно передать в качестве параметра.

function ZoomPic(pic)

{

pic.height = 100

}

function UnZoomPic(pic)

{

pic.height = 70

}

Но как ее идентифицировать? Можно задать каждой маленькой картинке имя (id) и передавать его, например, для первой картинки:

<img id=pic1 src="15-1461.jpg" height=70

onclick="ShowPic('15-1461.jpg')"

onmouseenter="ZoomPic(pic1)"

onmouseleave="UnZoomPic(pic1)">

Но это не очень удобно, особенно, когда картинок много.

В JavaScript используется указатель на текущий объект this (аналог self в Дельфи).

<img src="15-1461.jpg" height=70

onclick="ShowPic('15-1461.jpg')"

onmouseenter="ZoomPic(this)"

onmouseleave="UnZoomPic(this)">

this – это тот объект, с которым произошло событие.

### Задание

Самостоятельно измените код так, чтобы в ShowPic автоматически передавался текущий рисунок (т.е. не приходилось вручную писать имя рисунка в onclick). При клике выведите на экран не только сам рисунок, но и имя файла в виде ссылки.

## Литература

1. Актуальный обновляемый учебник, один из наиболее полных: <https://learn.javascript.ru/>
2. Справочник JavaScript от Mozilla: <https://developer.mozilla.org/ru/docs/JavaScript>
3. Канал "JavaScript решает": <https://www.youtube.com/channel/UCP-xJwnvKCGyS-nbyOx1Wmg/>
4. Хорошие, но старые уроки: <http://uroki-javascript.ru/>