**МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ**

федеральное государственное бюджетное   
образовательное учреждения высшего образования

**«КУЗБАССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ   
УНИВЕРСИТЕТ ИМ. Т. Ф. ГОРБАЧЕВА»**

филиал КузГТУв г. Прокопьевске

**Методические указания   
для выполнения практической работы**   
по дисциплине «Информационные технологии» по теме  
«JavaScript»  
для студентов очной формы обучения направления подготовки 09.03.02 «Информационные системы и технологии», профиль – 01 «Системная интеграция и автоматизация информационных процессов»

Составитель: А. С. Абрамович

Рассмотрены и утверждены  
на заседании кафедры  
Протокол № X от X.X.2021 г.

Рекомендованы к печати  
учебно-методической комиссией

Протокол № X от X.X.2021 г.

Электронная копия находится  
в библиотеке филиала КузГТУ  
в г. Прокопьевске

Прокопьевск 2021

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 4](#_Toc91728619)

[Что такое JavaScript? 5](#_Toc91728620)

[Что может JavaScript в браузере? 5](#_Toc91728621)

[Чего НЕ может JavaScript в браузере? 6](#_Toc91728622)

[Что делает JavaScript особенным? 8](#_Toc91728623)

[Языки «над» JavaScript 8](#_Toc91728624)

[Скрипт 9](#_Toc91728625)

[Расположение тегов <script> 11](#_Toc91728626)

[Функция 12](#_Toc91728627)

[Вызов функции 12](#_Toc91728628)

[Переменные 12](#_Toc91728629)

[Циклы в JavaScript 13](#_Toc91728630)

[Задание 16](#_Toc91728631)

[Список литературы 18](#_Toc91728632)

**Рецензент:**

Старший преподаватель кафедры информационных технологий, машиностроения и автотранспорта

**Иноземцев М.А.**

# ВВЕДЕНИЕ

Данные методические указания разработаны для обучающихся направления подготовки ВО 09.03.02 Информационные системы и технологии, профиль – «Системная интеграция и автоматизация информационных процессов». Позволяют освоить практические навыки и закрепить теоретические навыки по дисциплине «Информационные технологии» по теме «JavaScript».

# [Что такое JavaScript?](https://learn.javascript.ru/intro" \l "chto-takoe-javascript)

Изначально JavaScript был создан, чтобы «сделать веб-страницы живыми».

Программы на этом языке называются скриптами. Они могут встраиваться в HTML и выполняться автоматически при загрузке веб-страницы.

Скрипты распространяются и выполняются, как простой текст. Им не нужна специальная подготовка или компиляция для запуска.

Это отличает JavaScript от другого языка – [Java](https://ru.wikipedia.org/wiki/Java).

Когда JavaScript создавался, у него было другое имя – «LiveScript». Однако, язык Java был очень популярен в то время, и было решено, что позиционирование JavaScript как «младшего брата» Java будет полезно.

Со временем JavaScript стал полностью независимым языком со своей собственной спецификацией, называющейся [ECMAScript](http://ru.wikipedia.org/wiki/ECMAScript), и сейчас не имеет никакого отношения к Java.

Сегодня JavaScript может выполняться не только в браузере, но и на сервере или на любом другом устройстве, которое имеет специальную программу, называющуюся [«движком» JavaScript](https://ru.wikipedia.org/wiki/%D0%94%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript).

У браузера есть собственный движок, который иногда называют «виртуальная машина JavaScript».

Разные движки имеют разные «кодовые имена». Например:

* [V8](https://ru.wikipedia.org/wiki/V8_(%D0%B4%D0%B2%D0%B8%D0%B6%D0%BE%D0%BA_JavaScript)) – в Chrome и Opera.
* [SpiderMonkey](https://ru.wikipedia.org/wiki/SpiderMonkey) – в Firefox.
* …Ещё есть «Trident» и «Chakra» для разных версий IE, «ChakraCore» для Microsoft Edge, «Nitro» и «SquirrelFish» для Safari и т.д.

Эти названия полезно знать, так как они часто используются в статьях для разработчиков. Мы тоже будем их использовать. Например, если «функциональность X поддерживается V8», тогда «Х», скорее всего, работает в Chrome и Opera.

# [Что может JavaScript в браузере?](https://learn.javascript.ru/intro" \l "chto-mozhet-javascript-v-brauzere)

Современный JavaScript – это «безопасный» язык программирования. Он не предоставляет низкоуровневый доступ к памяти или процессору, потому что изначально был создан для браузеров, не требующих этого.

Возможности JavaScript сильно зависят от окружения, в котором он работает. Например, [Node.JS](https://ru.wikipedia.org/wiki/Node.js) поддерживает функции чтения/записи произвольных файлов, выполнения сетевых запросов и т.д.

В браузере для JavaScript доступно всё, что связано с манипулированием веб-страницами, взаимодействием с пользователем и веб-сервером.

Например, в браузере JavaScript может:

* Добавлять новый HTML-код на страницу, изменять существующее содержимое, модифицировать стили.
* Реагировать на действия пользователя, щелчки мыши, перемещения указателя, нажатия клавиш.
* Отправлять сетевые запросы на удалённые сервера, скачивать и загружать файлы (технологии [AJAX](https://ru.wikipedia.org/wiki/AJAX) и [COMET](https://ru.wikipedia.org/wiki/Comet_(%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5))).
* Получать и устанавливать куки, задавать вопросы посетителю, показывать сообщения.
* Запоминать данные на стороне клиента («local storage»).

# [Чего НЕ может JavaScript в браузере?](https://learn.javascript.ru/intro#chego-ne-mozhet-javascript-v-brauzere)

Возможности JavaScript в браузере ограничены ради безопасности пользователя. Цель заключается в предотвращении доступа недобросовестной веб-страницы к личной информации или нанесения ущерба данным пользователя.

Примеры таких ограничений включают в себя:

* JavaScript на веб-странице не может читать/записывать произвольные файлы на жёстком диске, копировать их или запускать программы. Он не имеет прямого доступа к системным функциям ОС.

Современные браузеры позволяют ему работать с файлами, но с ограниченным доступом, и предоставляют его, только если пользователь выполняет определённые действия, такие как «перетаскивание» файла в окно браузера или его выбор с помощью тега <input>.

Существуют способы взаимодействия с камерой/микрофоном и другими устройствами, но они требуют явного разрешения пользователя. Таким образом, страница с поддержкой JavaScript не может незаметно включить веб-камеру, наблюдать за происходящим и отправлять информацию в [ФСБ](https://ru.wikipedia.org/wiki/%D0%A4%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D1%81%D0%BB%D1%83%D0%B6%D0%B1%D0%B0_%D0%B1%D0%B5%D0%B7%D0%BE%D0%BF%D0%B0%D1%81%D0%BD%D0%BE%D1%81%D1%82%D0%B8_%D0%A0%D0%BE%D1%81%D1%81%D0%B8%D0%B9%D1%81%D0%BA%D0%BE%D0%B9_%D0%A4%D0%B5%D0%B4%D0%B5%D1%80%D0%B0%D1%86%D0%B8%D0%B8).

* Различные окна/вкладки не знают друг о друге. Иногда одно окно, используя JavaScript, открывает другое окно. Но даже в этом случае JavaScript с одной страницы не имеет доступа к другой, если они пришли с разных сайтов (с другого домена, протокола или порта).

Это называется «Политика одинакового источника» (Same Origin Policy). Чтобы обойти это ограничение, обе страницы должны согласиться с этим и содержать JavaScript-код, который специальным образом обменивается данными.

Это ограничение необходимо, опять же, для безопасности пользователя. Страница https://anysite.com, которую открыл пользователь, не должна иметь доступ к другой вкладке браузера с URL https://gmail.com и воровать информацию оттуда.

* JavaScript может легко взаимодействовать с сервером, с которого пришла текущая страница. Но его способность получать данные с других сайтов/доменов ограничена. Хотя это возможно в принципе, для чего требуется явное согласие (выраженное в заголовках HTTP) с удалённой стороной. Опять же, это ограничение безопасности.

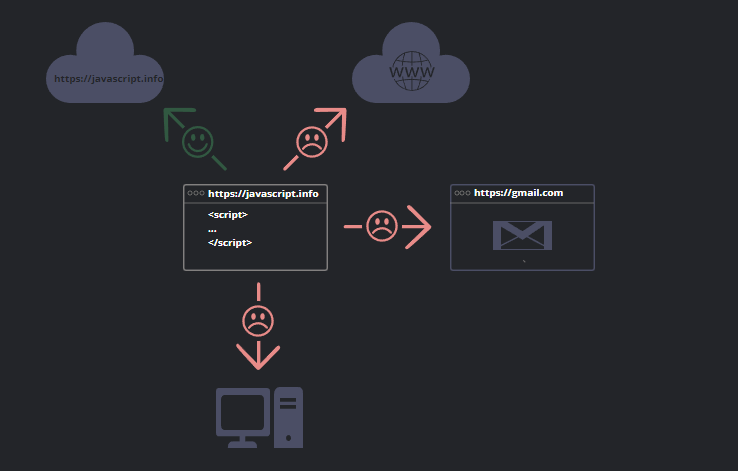


Рисунок 1 - Пример концепции взаимодействия скриптов с другими объектами.

Подобные ограничения не действуют, если JavaScript используется вне браузера, например — на сервере. Современные браузеры предоставляют плагины/расширения, с помощью которых можно запрашивать дополнительные разрешения.

# [Что делает JavaScript особенным?](https://learn.javascript.ru/intro#chto-delaet-javascript-osobennym)

Как минимум, три сильные стороны JavaScript:

* Полная интеграция с HTML/CSS.
* Простые вещи делаются просто.
* Поддерживается всеми основными браузерами и включён по умолчанию.

JavaScript – это единственная браузерная технология, сочетающая в себе все эти три вещи.

Вот что делает JavaScript особенным. Вот почему это самый распространённый инструмент для создания интерфейсов в браузере.

Хотя, конечно, JavaScript позволяет делать приложения не только в браузерах, но и на сервере, на мобильных устройствах и т.п.

# [Языки «над» JavaScript](https://learn.javascript.ru/intro#yazyki-nad-javascript)

Синтаксис JavaScript подходит не под все нужды. Разные люди хотят иметь разные возможности.

Это естественно, потому что проекты разные и требования к ним тоже разные.

Так, в последнее время появилось много новых языков, которые транспилируются (конвертируются) в JavaScript, прежде чем запустятся в браузере.

Современные инструменты делают транспиляцию очень быстрой и прозрачной, фактически позволяя разработчикам писать код на другом языке, автоматически преобразуя его в JavaScript «под капотом».

Примеры таких языков:

* [CoffeeScript](http://coffeescript.org/) добавляет «синтаксический сахар» для JavaScript. Он вводит более короткий синтаксис, который позволяет писать чистый и лаконичный код. Обычно такое нравится Ruby-программистам.
* [TypeScript](http://www.typescriptlang.org/) концентрируется на добавлении «строгой типизации» для упрощения разработки и поддержки больших и сложных систем. Разработан Microsoft.
* [Flow](http://flow.org/) тоже добавляет типизацию, но иначе. Разработан Facebook.
* [Dart](https://www.dartlang.org/) стоит особняком, потому что имеет собственный движок, работающий вне браузера (например, в мобильных приложениях). Первоначально был предложен Google, как замена JavaScript, но на данный момент необходима его транспиляция для запуска так же, как для вышеперечисленных языков.
* [Brython](https://brython.info/) транспилирует Python в JavaScript, что позволяет писать приложения на чистом Python без JavaScript.

Есть и другие. Но даже если мы используем один из этих языков, мы должны знать JavaScript, чтобы действительно понимать, что мы делаем.

# Скрипт

Под скриптом понимают программы, разработанные на языке программирования JavaScript.

Скрипт может быть 2 видов:

1.Внутренний, для использования которого достаточно внутри html-документа, а именно внутри тэга html, после двухкомпонентных тэгов <head> и <body> объявить двухкомпонентный тэг <script>. Структура html-документа с использованием внутреннего скрипта представлена далее.

<html>

<head>

</head>

<body>

</body>

<script>

</script>

</html>

Если JavaScript-кода много – его выносят в отдельный файл, который, как правило, имеет расширение .js.

2.Внешний скрипт, для использования которого достаточно создать файл с расширением .js и подключить его к нашей HTML-странице следующим образом:

Чтобы включить в HTML-документ JavaScript-кoд из внешнего файла, нужно использовать атрибут src (source) тега <script>. Его значением должен быть URL-aдpec файла, в котором содержится JS-код:

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

В этом примере указан абсолютный путь к файлу с именем script.js, содержащему скрипт (из корня сайта). Сам файл должен содержать только JavaScript-кoд, который иначе располагался бы между тегами <script> и </script>.

По аналогии с элементом <img> атрибуту src элемента <script> можно назначить полный URL-aдpec, не относящийся к домену текущей НТМL-страницы:

<script src=" [http://www.somesite.com/script.js"></script](http://www.somesite.com/script.js%22%3e%3c/script)>

Чтобы подключить несколько скриптов, используйте несколько тегов:

<script src="/scripts/script1.js"></script>

<script src="/scripts/script2.js"></script>

**Примечание:** элемент <script> с атрибутом src не может содержать дополнительный JаvаSсriрt-код между тегами <script> и </script>, хотя внешний сценарий выполняется, встроенный код игнорируется.

<!DOCTYPE html>  
<html>  
<head>  
<meta charset="utf-8">  
</head>  
<body>  
<script src="script.js">  
   alert('Привет, мир!');  
</script>  
<p>При наличии атрибута src внутренняя часть тега script игнорируется!</p>  
</body>  
</html>

Независимо от того, как JS-код включается в НТМL-документ, элементы <script> интерпретируются браузером в том порядке, в котором они расположены в HTML-документе. Сначала интерпретируется код первого элемента <script>, затем браузер приступает ко второму элементу <script> и т. д.

Внешние скрипты практичны, когда один и тот же код используется во многих разных веб-страницах. Браузер скачает js-файл один раз и в дальнейшем будет брать его из своего кеша, благодаря чему один и тот же скрипт, содержащий, к примеру, библиотеку функций, может использоваться на разных страницах без полной перезагрузки с сервера. Кроме этого, благодаря внешним скриптам, упрощается сопровождение кода, поскольку вносить изменения или исправлять ошибки приходится только в одном месте.

# Расположение тегов <script>

Вы уже знаете, что браузер читает HTML-документ сверху вниз и, начинает отображать страницу, показывая часть документа до тега <script>. Встретив тег <script>, переключается в JavaScript-режим и выполняет сценарий. Закончив выполнение, возвращается обратно в HTML-режим и отображает оставшуюся часть документа.

Это наглядно демонстрирует следующий пример. Метод alert() выводит на экран модальное окно с сообщением и приостанавливает выполнение скрипта, пока пользователь не нажмёт «ОК»:

<!DOCTYPE html>  
<html>  
<body>  
<p>Начало контента...</p>  
<script src="script.js"></script>  
<p>...Продолжение контента</p>  
</body>  
</html>

Если на странице используется много скриптов JavaScript, то могут возникнуть длительные задержки при загрузке, в течение которых пользователь видит пустое окно браузера. Поэтому считается хорошей практикой все ссылки нa javaScript-cцeнapии указывать после контента страницы перед закрывающим тегом <body>:

<body>

<!-- Контент страницы -->

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

<script src="script2.js"></script>

</body>

Такое расположение сценариев позволяет браузеру загружать страницу быстрее, так как сначала загрузится контент страницы, а потом будет загружаться код сценария.  
Для пользователей это предпочтительнее, потому что страница полностью визуализируется в браузере до обработки JavaScript-кoдa.

# Функция

Под функцией в JavaScript понимается фрагмент программного кода, который отвечает за выполнение отдельного метода на web-странице. Функция объявляется внутри двухкомпонентного тэга <script> ключевым словом function Наименование()

{

//Блок операторов тела функции

}

# Вызов функции

JavaScript – событийно – ориентированный язык программирования, который позволяет запускать функции, отвечающие за «динамику» на web-странице при наступлении определенных событий, например как нажатие кнопки, рассмотрим такой пример ниже.

Первым действием нужно создать управляющий элемент типа кнопка и в свойстве OnClick описать вызов необходимой нам функции:

<input type = “button” name = “button1” value = “Кнопка 1” OnClick = НаименованиеФункции()>

# Переменные

Переменная – это именованная область памяти, которая служит для хранения данных, которые могут меняться в процессе выполнения программы.

Любая переменная в JavaScript как и в других языках программирования состоит из типа данных и названия переменной. На первоначальных этапах в целях изучения типов данных и принципа работы с переменными мы будем использовать тип данных «var» и посмотрим различные вариации работы с переменными такого типа данных. Примеры объявления переменной:

var i = 1; - пример объявления переменной var, хранящей целочисленные значения

var i1 = “Информационные технологии”; - пример объявления переменной var, хранящей строковые значения

# Циклы в JavaScript

При написании скриптов зачастую встаёт задача сделать однотипное действие много раз.

Например, вывести товары из списка один за другим. Или просто перебрать все числа от 1 до 10 и для каждого выполнить одинаковый код.

Для многократного повторения одного участка кода предусмотрены *циклы*.

[**Цикл «while»**](https://learn.javascript.ru/while-for#tsikl-while)

Цикл while имеет следующий синтаксис:

while (condition) {

// код

// также называемый "телом цикла"

}

Код из тела цикла выполняется, пока условие condition истинно.

Например, цикл ниже выводит i, пока i < 3:

let i = 0;

while (i < 3) { // выводит 0, затем 1, затем 2

alert( i );

i++;

}

Одно выполнение тела цикла по-научному называется *итерация*. Цикл в примере выше совершает три итерации.

Если бы строка i++ отсутствовала в примере выше, то цикл бы повторялся (в теории) вечно. На практике, конечно, браузер не позволит такому случиться, он предоставит пользователю возможность остановить «подвисший» скрипт, а JavaScript на стороне сервера придётся «убить» процесс.

Любое выражение или переменная может быть условием цикла, а не только сравнение: условие while вычисляется и преобразуется в логическое значение.

Например, while (i) – более краткий вариант while (i != 0):

let i = 3;

while (i) { // когда i будет равно 0, условие станет ложным, и цикл остановится

alert( i );

i--;

}

сли тело цикла состоит лишь из одной инструкции, мы можем опустить фигурные скобки {…}

let i = 3;

while (i) alert(i--);

[**Цикл «do…while»**](https://learn.javascript.ru/while-for#tsikl-dowhile)

Проверку условия можно разместить под телом цикла, используя специальный синтаксис do..while:

do {

// тело цикла

} while (condition);

Цикл сначала выполнит тело, а затем проверит условие condition, и пока его значение равно true, он будет выполняться снова и снова.

Например:

let i = 0;

do {

alert( i );

i++;

} while (i < 3);

Такая форма синтаксиса оправдана, если вы хотите, чтобы тело цикла выполнилось **хотя бы один раз**, даже если условие окажется ложным. На практике чаще используется форма с предусловием: while(…) {…}.

[**Цикл «for»**](https://learn.javascript.ru/while-for#tsikl-for)

Более сложный, но при этом самый распространённый цикл — цикл for.

Выглядит он так:

for (начало; условие; шаг) {

// ... тело цикла ...

}

Давайте разберёмся, что означает каждая часть, на примере. Цикл ниже выполняет alert(i) для i от 0 до (но не включая) 3:

for (let i = 0; i < 3; i++) { // выведет 0, затем 1, затем 2

alert(i);

}

Рассмотрим конструкцию for подробней:

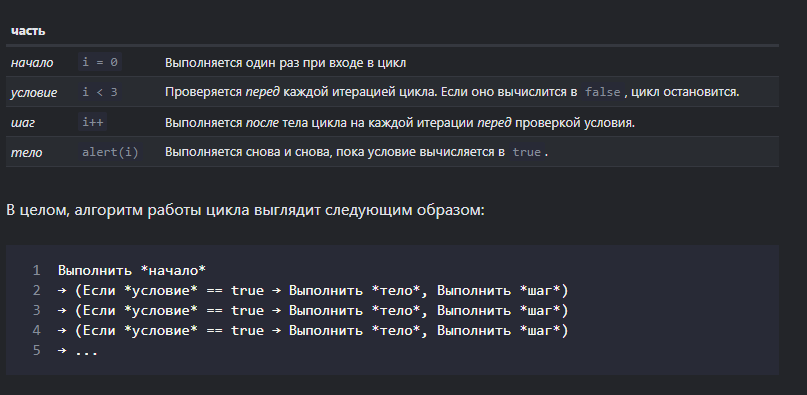


Рисунок 2 – Определение параметров цикла for.

Циклы позволяют решать задачи разных типов, касающиеся перебора коллекций значений, одной из задач данного спектра является задача перебора строки посимвольно, которая будет представлена ниже.

Язык программирования JavaScript довольно обширный, поддерживает работу с многими объектами, а также поддерживает работу с множеством различных методов, применяемых, например, к строкам и датам. Например, для того, чтобы получить длину строки достаточно к элементу управления, где хранится строковое значение применить напрямую или через переменную метод length(), пример использования которого представлен ниже:

var str = form1.text1.value;

var kolvo = str.length\*1;

Далее представлен пример, в котором с использованием цикла происходит посимвольный перебор значения, введенного в текстовое поле и извлечение из нее подстроки с использование метода substr(i,j), где i – с какого символа нужно начать перебор, j – сколько символ нужно извлечь, допустим, каждой итерацией цикла.

var str = form1.text1.value; //создали переменную для хранения значения, вводимого в текстовое поле

var kolvo = str.length\*1;//посчитали количество символов, которое введено в текстовое поле с помощью метода length

var i = 0\*1;// создали переменную – счетчик для цикла

for(i;i<=kolvo;i++)//создали сам цикл, который будет начинаться будет выполняться столько раз, пока переменная i не станет равна количеству строк, введенному в текстовое поле

{

var itog1 = "";

var itog2 = "";

var str2 = str.substr(i,1);//извлекаем подстроку из исходной строки по одному символу. Согласно условию цикла, переменная str2 будет при каждой итерации цикла принимать один символ исходной строки до тех пор пока значение i не станет равно переменной, хранящей данные о количестве символов, введенных в исходную строку.

# Задание

1. Сделать верстку страницы на языке разметки HTML для реализации калькулятора. Элементы управления должны соответствовать элементам управления, представленным на рисунке ниже.

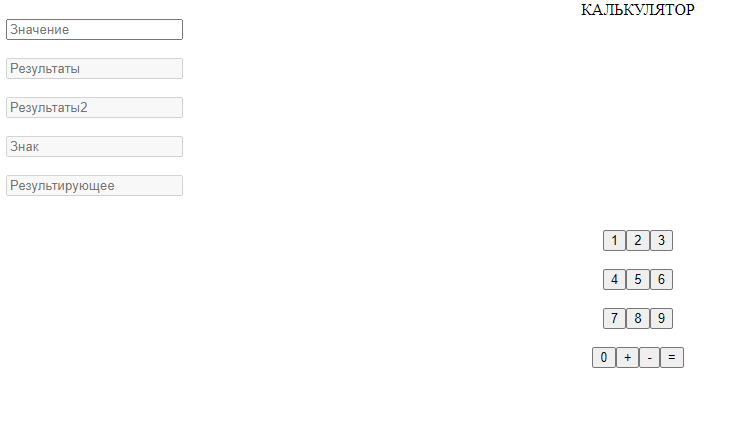


Рисунок 3 – Пример графического представления элементов управления на web-странице после верстки.

1. Написать функции для реализации сложения, деления, умножения и вычитания на web-странице в одной строке, как представлено на рисунке ниже.

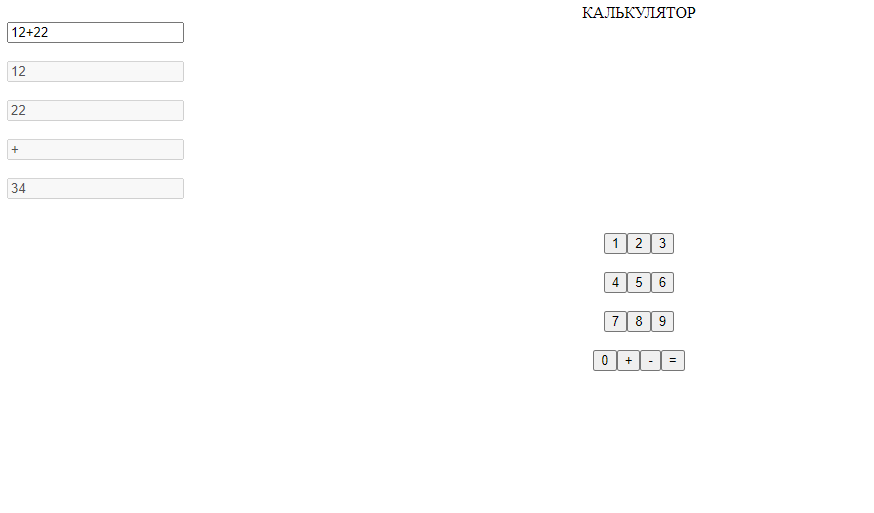


Рисунок 4 – Пример работы калькулятора и представления данных на примере управляющих элементов на web-странице.

**Примечание:** при выполнении задания обязательно использовать цикл for и оператор ветвления if (изучить самостоятельно).

# Список литературы

1. Информационные технологии и системы: Учебное пособие [Электронный ресурс]/ Е. Л. Федотова. - М.: ИД ФОРУМ: НИЦ ИНФРА-М, 2014. - 352 с. - Режим доступа: <http://znanium.com/bookread2.php?book=429113>
2. Введение в JavaScript [Электронный ресурс].- Режим доступа – Свободный.- URL: <https://learn.javascript.ru/intro.-> Дата обращения: 29.10.2021)

Составитель

**Абрамович Александр Сергеевич**

**Методические указания   
для выполнения практической работы**   
по дисциплине «Информационные технологии» по теме  
«JavaScript»  
для студентов очной формы обучения направления подготовки 09.03.02 «Информационные системы и технологии», профиль – 01 «Системная интеграция и автоматизация информационных процессов»

Сверстано и отпечатано в филиале КузГТУ в г. Прокопьевске.

653039, г. Прокопьевск, ул. Ноградская, 19а.

Формат 60×84 1/16. Объем X п. л. Заказ X.