



**Кафедра «Автоматизированные станочные системы»**  
**Dept. of Automated Manufacturing Systems**

---

# ЯЗЫК JAVASCRIPT

**Язык JavaScript** поддерживается всеми современными браузерами. Java-апплет представляет собой исполняемый код для так называемой виртуальной машины Java, встроенной в браузер.

Технология Java включает две составляющих: одноименный язык программирования и платформу Java, которая, в свою очередь, состоит из виртуальной машины Java (Java VM) и интерфейса прикладного программирования (Java API).

**Недостаток** программ для любой виртуальной машины – **низкая производительность**.

Исполняемый модуль для виртуальной машины Java именуется **байт-кодом** (или иногда **J-кодом**).

С помощью языка программирования Java можно создавать программы двух типов: **приложения** (applications) и **апплеты** (applets).

**Java-приложения** представляют собой полнофункциональные программы, которые выполняются под управлением виртуальной Java-машины, которая, в свою очередь, работает под управлением операционной системы.

**Java-апплет** есть байт-код для Java-машины, которая работает под управлением браузера.

Для помещения апплета на Web-страницу разработчик сайта должен:

- подготовить его байт-код с помощью какого-либо средства разработки,
- разместить файл с байт-кодом в каталоге Web-сервера,
- указать его URL в тексте разрабатываемой страницы в специальном теге **<applet>**.

Браузер клиента по завершению загрузки страницы с апплетом запускает свою виртуальную машину, которая начинает выполнение апплета.

**Апплет не имеет возможности доступа к локальным ресурсам клиентского компьютера.**

---

Все операции, которые можно исполнять в программе на JavaScript, описывают действия над объектами, которыми являются элементы рабочей области браузера (окно, документ, статусная строка...) и контейнеры языка HTML.

**НЕТ:**

- создаваемых программистом классов объектов,
- наследования в JavaScript.

**ИМЕЕТСЯ:**

- объекты с набором свойств и методов,
- обычные функции, больше похожие на процедуры из традиционных языков программирования,
- события.

Скрипт включается в HTML-документ с помощью тега **<script>**:

```
<script>  
<script language = "JavaScript">  
<! - -  
ТЕЛО СКРИПТА  
// - - >  
</script>
```

JavaScript-код может вставляться в элемент **<BODY>**.

Исполняется такой сценарий сразу после загрузки web-документа,

**например:**

```
<script type="text/javascript">  
{document.write("Эту строку вывел сценарий")}  
</script>
```

JavaScript-код можно вставлять в элемент **<HEAD>**.

---

Такой сценарий запускается в ответ на какое-либо событие, генерируемое системой или пользователем.

**Например:**

```
<HTML>
<HEAD>
<TITLE> Сценарий JavaScript </TITLE>
<script type="text/javascript">
function wrt()
{document.write("Эту строку вывел сценарий")}
</script>
</HEAD>
<BODY>
<INPUT type="button" value="Вывести строку"
onClick="wrt()"/>
</BODY>
</HTML>
```

JavaScript-код можно поместить в отдельном файле с расширением .js. Далее в разделе <HEAD> следует указать адрес этого файла следующим образом:

```
<script type='text/javascript' src='wrt_str.js'>  
</script>
```

Внешний сценарий также может запускаться и сразу после загрузки web-документа, или по какому-либо событию.

Сценарий можно разместить и непосредственно в обработчике события, **например**:

```
<INPUT type="button" value="Вывести строку"  
onClick="document.write("Эту строку вывел сценарий")"/>
```

## Типы данных в JavaScript

Тип	Описание
Булевские	Имеют значения true или false
Целочисленный	Могут быть выражены в десятичном, восьмеричном и шестнадцатеричном формате. Например: 45, 071, 0x7d8
С плавающей точкой	Определяют десятичное число с дробной частью, могут быть представлены в стандартной и экспоненциальной форме. Например: 3405.23, -123.09, 2.90E-12
Строки	Представляет символы Unicode, заключённых в двойные или одинарные кавычки, например: "Строка", 'Строчка'
Специальные символы	Управляющие символы: \\n перевод на новую строку \\t горизонтальная табуляция \\v вертикальная табуляция \\ обратная косая черта \' одинарная кавычка \" двойная кавычка

Переменные в JavaScript, как и в Pascal, задаются с помощью ключевого слова **var**:

```
var a = 27;
```

**JavaScript является языком со свободной типизацией.**

Одной и той же переменной можно присваивать различные типы данных:

```
var a = 27;  
a = 15.5;  
a = "String 1";
```

В Java предусмотрены два способа включения комментариев:

```
// Это однострочный комментарий  
/*а это много -  
строчный комментарий*/
```

## Арифметические действия

Оператор	Наименование	Пример
+	Сложение	$x + y$
-	Вычитание	$x - y$
*	Умножение	$x * y$
/	Деление	$x / y$
%	Взятие остатка (MOD)	$x \% y$
++	Увеличение на единицу	$x++$
--	Уменьшение на единицу	$y --$

Часть операций (++, --) являются унарными. К ним относятся префиксное и постфиксное возрастание ++; префиксное и постфиксное уменьшение --, унарный плюс; унарный минус.

## Оператор присваивания

Оператор	Пример	Эквивалентное выражение
=	$x = y$	$x = y$
+=	$x += y$	$x = x + y$
--	$x -= y$	$x = x - y$
*=	$x *= y$	$x = x * y$
/=	$x /= y$	$x = x / y$
%=	$x \% = y$	$x = x \% y$

## Логические операции

Оператор	Название	Пример
===	Строгое равенство	$x === y$
==	Равно	$x == y$
!=	Не равно	$x != y$
>	Больше	$x > y$
>=	Больше или равно	$x >= y$
<	Меньше	$x < y$
<=	Меньше или равно	$x <= y$
!	Отрицание (не)	$!x$
&&	и	$x \&\& y$
	или	$x    y$

Для конкатенации (объединения) строк используется оператор +:  
`var a = "a" + "b" + "c";`

Переменная a будет содержать строку "abc".

## Функции и события

---

**Функция (function)** - это группа операторов, предназначенных для определенной цели и объединенных под общим именем.

Функция имеет следующий общий вид:

```
function имяфункции([аргументы])  
{  
операторы;  
}
```

**Функции автоматически запускаться на выполнение не могут.**

Функции в языке JavaScript могут вызываться несколькими способами.

---

Указание имени функции непосредственно в блоке SCRIPT:

```
<script language="JavaScript">
// вызов функции
yourMessage();
// описание функции
function yourMessage()
{
alert("Наша первая функция");
}
</script>
```

Одну функцию можно вызывать из другой, но для "запуска" всего процесса обычно применяются события.

## Самые нужные события:

---

**Событие onLoad.** Это событие происходит после загрузки страницы в окне браузера. Оно считается состоявшимся только после полного завершения загрузки всей страницы, включая изображения. Событие, происходящее при выгрузке страницы (переходе на другую страницу или закрытии окна браузера), называется **onUnload**.

**Событие onClick.** Это событие происходит после щелчка мышью в определенном месте страницы. Множество элементов страницы (гиперссылки, изображения, кнопки и пр.) могут реагировать на событие **onClick**.

**Событие onMouseover.** Это событие происходит после наведения курсора мыши на определенный элемент страницы. Событие **onMouseover** можно связать практически с любым объектом Web-страницы (текстом, изображением, кнопками, гиперссылками и т.д.).

**Событие onMouseout.** Это событие происходит в тех случаях, когда курсор мыши отводится от объекта.

Пример функции, выводящей текущее время:

```
function announceTime()  
{  
  //get the date, the hour, minutes, and seconds  
  var the_date = new Date();  
  var the_hour = the_date.getHours();  
  var the_minute = the_date.getMinutes();  
  var the_second = the_date.getSeconds();  
  var the_time = the_hour + ":" + the_minute +  
  ":" + the_second;  
  alert("Время: " +the_time);  
}
```

Свяжем вызов данной функции с событием onmouseover абзаца (тег <p>):

```
<p onmouseover="announceTime();" >Который час?</p>
```

Теперь достаточно подвести курсор к фразе "Который час?" – и выскочит окошко с текущим временем.

## Массивы в JavaScript

Объекты и массивы (всегда динамические) создаются с помощью оператора **new**. Нумерация элементов массива начинается с нуля.

В JavaScript допускается возможность хранить различные типы данных в одном массиве:

```
var a = new Array();  
a[0] = 8;  
a[1] = 4.9;  
a[2] = "String 1";
```

С помощью JavaScript можно так же имитировать и многомерные массивы:

```
var a = new Array();  
a[0] = new Array();  
a[0][0] = "1,1";  
a[0][1] = "1,2";  
a[1] = new Array();  
a[1][0] = "2,1";  
a[1][1] = "2,2";
```

Для работы с массивами в JavaScript применяются **методы**:

**Метод join()**. Метод join() позволяет объединить элементы массива в одну строку. Он является обратным методу **split()**, который применяется к объектам типа "текстовая строка".

**Например:**

```
b = "http://intuit.ru/help/index.html";
```

```
b = split('/');
```

Получили массив b:

```
b[0]=http:
```

```
b[1]=
```

```
b[2]=intuit.ru
```

```
b[3]=help
```

```
b[4]=index.html
```

Объединили элементы массива b обратно в строку:

```
l=b.join("/");
```

Получили в результате:

```
http://intuit.ru/help/index.html
```

**Метод reverse().** Метод reverse() применяется для изменения на противоположный порядка элементов массива внутри массива.

**Пример:**

Предположим, массив натуральных чисел упорядочен по возрастанию:

```
a = new Array(1,2,3,4,5);
```

Упорядочим его по убыванию: `a.reverse();`

Результат: a[0]=5 a[1]=4 a[2]=3 a[3]=2 a[4]=1

**Метод sort().** Метод sort() позволяет отсортировать элементы массива в соответствии с некоторой функцией сортировки, чье имя используется в качестве аргумента метода.

**Например:**

```
a = new Array(1,6,9,9,3,5);
```

```
function g(a,b)
```

```
{
```

```
if(a > b) return 1;
```

```
if(a < b) return -1;
```

```
if(a==b) return 0;
```

```
}
```

```
b = a.sort(g);
```

В результате выполнения этого кода получим массив следующего вида:

```
b[0]=1 b[1]=3 b[2]=5 b[3]=6 b[4]=9 b[5]=9
```

Тройцкий Д.И.

Ввод данных можно осуществить с помощью функции **prompt**, а вывод - с помощью функции **alert**.

Например:

```
<html>
<head>
<title>Script</title>
</head>
<body>
<h1>Script</h1>
<script language = "JavaScript">
<!--
var a = prompt("Input A: ", 10);
var b = prompt("Input B: ", "");
a = parseFloat(a);
b = parseFloat(b);
var c = a + b;
alert(c);
//-->
</script>
</body>
</html>
```

Первый аргумент функции **prompt** является строкой, которая будет использована в качестве приглашения пользователю, а второй – значением по умолчанию. Если не предполагается использовать значение по умолчанию, то необходимо задать пустую строку "".

Так как функция **prompt** возвращает строку, для того, чтобы преобразовать ее в число, используются функции **parseFloat** и **parseInt**, преобразующие текст в переменную вещественного и целого типа соответственно.