JavaScript на веб-страницах

* **Содержание:**
* [1. Подключение сценариев к html-документу](https://html5book.ru/osnovy-javascript/#part1)
* [2. Типы данных и переменные в JavaScript](https://html5book.ru/osnovy-javascript/#part2)
* [2.1. Переменные в JavaScript](https://html5book.ru/osnovy-javascript/#part3)
* [2.2. Типы данных переменных](https://html5book.ru/osnovy-javascript/#part4)
* [2.3. Глобальные и локальные переменные](https://html5book.ru/osnovy-javascript/#part5)

1. Подключение сценариев к html-документу

Сценарии JavaScript бывают **встроенные**, т.е. их содержимое является частью документа, и **внешние**, хранящиеся в отдельном файле с расширением .js. Сценарии можно внедрить в html-документ следующими способами:

**В виде гиперссылки**.  
Для этого нужно разместить код в отдельном файле и включить ссылку на файл в заголовок

<head>

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

</head>

или тело страницы.

<body>

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

</body>

Этот способ обычно применяется для сценариев большого размера или сценариев, многократно используемых на разных веб-страницах.

**В виде обработчика события**.  
Каждый html-элемент имеет JavaScript-события, которые срабатывают в определенный момент. Нужно добавить необходимое событие в html-элемент как атрибут, а в качестве значения этого атрибута указать требуемую функцию. Функция, вызываемая в ответ на срабатывание события, является **обработчиком события**. В результате срабатывания события исполнится связанный с ним код. Этот способ применяется в основном для коротких сценариев, например, можно установить смену цвета фона при нажатии на кнопку:

<script>

var colorArray = ["#5A9C6E", "#A8BF5A", "#FAC46E", "#FAD5BB", "#F2FEFF"]; // создаем массив с цветами фона

var i = 0;

function changeColor(){

document.body.style.background = colorArray[i];

i++;

if( i > colorArray.length - 1){

i = 0;

}

}

</script>

<button onclick="changeColor();">Change background</button>

**Внутрь элемента** <script>.  
Элемент <script> может вставляться в любое место документа. Внутри тега располагается код, который выполняется сразу после прочтения браузером, или содержит описание функции, которая выполняется в момент ее вызова. Описание функции можно располагать в любом месте, главное, чтобы к моменту ее вызова код функции уже был загружен.

Обычно код JavaScript размещается в заголовке документа (элемент <head>) или после открывающего тега <body>. Если скрипт используется после загрузки страницы, например, код счетчика, то его лучше разместить в конце документа:

<footer>

<script>

document.write("Введите свое имя");

</script>

</footer>

</body>

2. Типы данных и переменные в JavaScript

Компьютеры обрабатывают информацию — данные. Данные могут быть представлены в различных формах или типах. Большая часть функциональности JavaScript реализуется за счет простого набора объектов и типов данных. Функциональные возможности, связанные со строками, числами и логикой, базируются на строковых, числовых и логических типах данных. Другая функциональная возможность, включающая регулярные выражения, даты и математические операции, осуществляется с помощью объектов RegExp, Date и Math.

**Литералы** в JavaScript представляют собой особый класс типа данных, фиксированные значения одного из трех типов данных — строкового, числового или логического:

"это строка"

3.14

true

alert("Hellow"); // "Hellow" - это литерал

var myVariable = 15; // 15 - это литерал

**Примитивный тип данных** является экземпляром определенного типа данных, таких как строковый, числовой, логический, null и undefined.

2.1. Переменные в JavaScript

Данные, обрабатываемые сценарием JavaScript, являются **переменными**. Переменные представляют собой именованные контейнеры, хранящие данные (значения) в памяти компьютера, которые могут изменяться в процессе выполнения программы. Переменные имеют **имя**, **тип** и **значение**.

Имя переменной, или **идентификатор**, может включать только буквы a-z, A-Z, цифры 0-9 (цифра не может быть первой в имени переменной), символ $ (может быть только первым символом в имени переменной или функции) и символ подчеркивания \_, наличие пробелов не допускается. Длина имени переменной не ограничена. Можно, но не рекомендуется записывать имена переменных буквами русского алфавита, для этого они должны быть записаны в Unicode.

В качестве имени переменной нельзя использовать ключевые слова JavaScript. Имена переменных в JavaScript чувствительные к регистру, что означает, что переменная var message; и var Message; — разные переменные.

Переменная создается (объявляется) с помощью ключевого слова var, за которым следует имя переменной, например, var message;. Объявлять переменную необходимо перед ее использованием.

Переменная **инициализируется** значением с помощью операции присваивания =, например, var message="Hellow";, т.е. создается переменная message и в ней сохраняется ее **первоначальное** значение "Hellow". Переменную можно объявлять без значения, в этом случае ей присваивается значение по умолчанию undefined. Значение переменной может изменяться во время исполнения скрипта. Разные переменные можно объявлять в одной строке, разделив их запятой:  
var message="Hellow", number\_msg = 6, time\_msg = 50;

2.2. Типы данных переменных

JavaScript является нетипизированным языком, тип данных для конкретной переменной при ее объявлении указывать не нужно. Тип данных переменной зависит от значений, которые она принимает. Тип переменной может изменяться в процессе совершения операций с данными (**динамическое приведение типов**). Преобразование типов выполняется автоматически в зависимости от того, в каком контексте они используются. Например, в выражениях, включающих числовые и строковые значения с оператором +, JavaScript преобразует числовые значения в строковые:

var message = 10 + " дней до отпуска";

// вернет "10 дней до отпуска"

Получить тип данных, который имеет переменная, можно с помощью оператора typeof. Этот оператор возвращает строку, которая идентифицирует соответствующий тип.

typeof 35; // вернет "number"

typeof "text"; // вернет "string"

typeof true; // вернет "boolean"

typeof [1, 2, 4]; // вернет "object"

typeof undefined; // вернет "undefined"

typeof null; // вернет "object"

Все типы данных в JavaScript делятся на две группы — **простые** типы данных *(primitive data types)* и **составные** типы данных *(composite data types)*.

К **простым** типам данных относят строковый, числовой, логический, null и underfined.

2.2.1. Строковый тип (string)

Используется для хранения строки символов, заключенных в двойные или одинарные кавычки. Пустой набор символов, заключенный в одинарные или двойные кавычки, является пустой строкой. Число, заключенное в кавычки, также является строкой.

var money = ""; // пустая строка, ноль символов

var work = 'test';

var day = "Sunday";

var x = "150";

В строку в двойных кавычках можно включить одиночную кавычку и наоборот. Кавычка того же типа отключается с помощью символа обратного слэша \ (так называемая **escape-последовательность**):

document.writeln("\"Доброе утро, Иван Иваныч!\"\n");

// выведет на экран "Доброе утро, Иван Иваныч!"

Строки можно сравнивать, а также объединять с помощью операции конкатенации +. Благодаря автоматическому приведению типов можно объединять числа и строки. Строки являются постоянными, после того, как строка создана, она не может быть изменена, но может быть создана новая строка путем объединения других строк.

2.2.2. Числовой тип (number)

Используется для числовых значений. Числа в языке JavaScript бывают двух типов: целые числа *(integer)* и числа с плавающей точкой *(floating-point number)*. Целочисленные величины могут быть положительными, например 1, 2, и отрицательными, например –1, –2, или равными нулю. 1 и 1.0 — одно и то же значение. Большинство чисел в JavaScript записываются в десятичной системе счисления, также может использоваться восьмеричная и шестнадцатеричная системы.

В **десятичной** системе значения числовых переменных задаются с использованием арабских цифр 1, 2, 3, 4, 5, 6, 7, 8, 9, 0.

В **восьмеричном** формате числа представляет собой последовательность, содержащая цифры от 0 до 7 и начинающаяся с префикса 0.

Для **шестнадцатеричного** формата добавляется префикс 0x (0X), за которым следует последовательность из цифр от 0 до 9 или букв от a (A) до f (F), соответствующие значениям от 10 до 15.

var a = 120; // целое десятичное числовое значение

var b = 012; // восьмеричный формат

var c = 0xfff; // шестнадцатеричный формат

var d = 0xACFE12; // шестнадцатеричный формат

Числа с плавающей точкой представляют собой числа с дробной десятичной частью, либо это числа, выраженные в экспоненциальном виде. Экспоненциальная запись чисел предполагает следующий вид: число с дробной десятичной частью, за ним следует буква e, которая может быть указана как в верхнем, так и в нижнем регистре, далее — необязательный знак + или - и целая экспонента.

var a = 6.24; // вещественное число

var b = 1.234E+2; // вещественное число, эквивалентно 1.234 Х 10²

var c = 6.1e-2; // вещественное число, эквивалентно 6.1 Х 10‾²

2.2.3. Логический тип (boolean)

Данный тип имеет два значения, true (истина), false (ложь). Используется для сравнения и проверки условий.

var answer = confirm("Вам понравилась эта статья?\n Нажмите ОК. Если нет, то нажмите Cancel.");

if (answer == true)

{

alert("Спасибо!");

}

Также существуют специальные типы простых значений:  
**нулевой тип** — данный тип имеет одно значение null, которое используется для представления несуществующих объектов.

**неопределенный тип** — тип переменной underfined означает отсутствие первоначального значения переменной, а также несуществующее свойство объекта.

**Составные типы данных** состоят из более чем одного значения. К ним относятся объекты и особые типы объектов — массивы и функции. Объекты содержат свойства и методы, массивы представляют собой индексированный набор элементов, а функции состоят из коллекции инструкций.

2.3. Глобальные и локальные переменные

Переменные по области видимости делятся на **глобальные** и **локальные**. **Область видимости** представляет собой часть сценария, в пределах которой имя переменной связано с этой переменной и возвращает ее значение. Переменные, объявленные внутри тела функции, называются **локальными**, их можно использовать только в этой функции. Локальные переменные создаются и уничтожаются вместе с соответствующей функцией.

Переменные, объявленные внутри элемента <script>, или внутри функции, но без использования ключевого слова var, называются **глобальными**. Доступ к ним может осуществляться на протяжении всего времени, пока страница загружена в браузере. Такие переменные могут использоваться всеми функциями, позволяя им обмениваться данными.

Глобальные переменные попадают в **глобальное пространство имен**, которое является местом взаимодействия отдельных компонентов программы. Не рекомендуется объявлять переменные таким способом, так как аналогичные имена переменных уже могут использоваться любым другим кодом, вызывая сбой в работе скрипта.

Глобальное пространство в JavaScript представляется глобальным объектом window. Добавление или изменение глобальных переменных автоматически обновляет глобальный объект. В свою очередь, обновление глобального объекта автоматически приводит к обновлению глобального пространства имен.

Если глобальная и локальная переменная имеют одинаковые имена, то локальная переменная будет иметь преимущество перед глобальной.

Локальные переменные, объявленные внутри функции в разных блоках кода, имеют одинаковые области видимости. Тем не менее, рекомендуется помещать объявления всех переменных в начале функции.