Шпаргалка: введение в JavaScript

Переменные, операция присваивания

```
// двойной слеш применяют, чтобы написать комментарий на одной строке

/*
Вот так
можно писать
длинные комментарии.
*/
```

Переменная — это контейнер, в который можно положить какое-нибудь значение и потом его извлечь.

```
let bear; // объявлена переменная bear
```

```
bear = 'Мишка'; // в JavaScript текст берут в кавычки
```

```
// все переменные объявлены независимо друг от друга
let bear = 'Мишка';
let fox = 'Лисичка';
let rabbit = 'Кролик';
```

```
// варианты записи:
let bear = 'Мишка', fox = 'Лисичка', rabbit = 'Кролик';

// или
let bear = 'Мишка',
    fox = 'Лисичка',
    rabbit = 'Кролик';
```

`console.log` — узнать, какое значение в переменной.

```
console.log("Лев"); // выведет «Лев»

let bear = 'Мишка';
console.log(bear); // выведет «Мишка»
```

```
let bear = 'Мишка';
let fox = 'Лиса';

console.log(bear, fox, 'Кролик'); // «Мишка Лиса Кролик»
```

`undefined` — это специальное значение в JavaScript — «не определено».

```
let owl;
console.log(owl); // выведет undefined
```

Когда применяют присваивание `=`, в переменной сохраняется указанное значение.

```
bear = fox; // теперь в переменной bear хранится значение 'Лисичка'
```

Константы — это те же переменные, но без возможности изменить значение.

Чтобы объявить константу, примени слово `const`:

```
const lion = 'Лев';
lion = "Мишка"; // ошибка — константу невозможно изменить

const tiger; // ошибка — константу нельзя объявить без значения
```

В названиях переменных и констант нельзя применять зарезервированные слова. Зарезервированные слова — это элементы языка

программирования. Например, `let` или `const`:

```
let let; // ошибка
```

Также в названиях переменных и констант запрещены дефисы:

```
let bear-1; // ошибка
let bear_1; // нижнее подчёркивание использовать можно
```

Названия переменных и констант нельзя начинать с цифр или спецсимволов:

```
let 1bear; // ошибка
let !bear; // ошибка
```

Спецсимволы — это чаще всего любые знаки препинания: `. , ! : "`. В JavaScript названия переменных и констант можно начинать с `_` и `$`:

```
let _bear = 'Мишка';
let $lion = 'Лев';
```

Переменные, которые начинаются с прописной (верхний регистр) и строчной (нижний регистр) буквы, — это разные переменные:

```
let bear = 'Мишка';
let Bear = 'Медведь';
let BEAR = 'МЕДВЕДЬ';

// bear, Bear, BEAR — три разные переменные
console.log(bear, Bear, BEAR); // «Мишка Медведь МЕДВЕДЬ»
```

Если название переменной состоит из нескольких слов, в JavaScript принято писать их в стиле camelCase — когда первое слово пишется с маленькой буквы, а каждое следующее — с большой.

```
let bearAge = 20;
let bearWeight = 90;
let bearFathersName = 'Степан';
```

Элементарные типы данных

Оператор typeof

`typeof` определяет тип данных, который сейчас хранится в переменной.

```
let age = 20;
let name = "Иван";
let isStudent = true;
let someVariable;
typeof age; // вернёт number
typeof name; // вернёт string
typeof isStudent; // вернёт boolean
typeof someVariable; // вернёт undefined
```

Number

В JavaScript и целые, и дробные числа относятся к одному типу — **number**. Дробные числа записывают через точку.

```
let age = 18;
let height = 176.05;
typeof age; // вернёт number
typeof height; // вернёт number
```

String

Значение типа **string** — строка, в которой может быть любой текст.

Текст записывают в кавычках: можно применять одинарные кавычки, двойные кавычки и грависы. Грависы ` — это обратные кавычки.

```
let word = 'Привет!';
console.log(word); // выведет «Привет!»
word = "Пока!";
console.log(word); // выведет «Пока!»
word = `И снова привет!`;
console.log(word); // выведет «И снова привет!»
```

Важно открывать и закрывать значение одинаковыми кавычками.

```
let phrase = 'Привет!"; // ошибка
phrase = "Пока!`; // ошибка
```

Если в тексте уже есть кавычки какого-то типа, строку лучше обособлять кавычками другого типа.

```
let hiBaby = 'Привет, "Малыш"!';
console.log(hiBaby); // выведет «Привет, "Малыш"!»
```

Boolean

Значения типа **boolean** делят на **true** (верно) и **false** (ложно).

```
let isEmploee = false; // не является сотрудником
typeof isEmployee; // "boolean"
console.log(isEmploee); // выведет false — ложь
```

```
isEmploee = true;
console.log(isEmploee); // выведет true — истина
```

Объекты

Object

Чтобы в одной переменной хранить несколько пар «имя — значение», применяют тип данных **object**.

```
let dog = {
  nickname : 'Бобби',
  breed : 'Овчарка',
  age : 8,
  weight : 22
};

console.log(typeof dog); // object
```

Информация оформлена в парах «имя (или ключ) — значение». Получить значение по ключу можно двумя способами:

```
console.log(dog.breed); // выведет «Овчарка»
console.log(dog['breed']); // выведет «Овчарка»

console.log(dog.height); // выведет undefined
console.log(dog['height']); // выведет undefined
```

Array

Массив (**array**) — это упрощённая реализация объекта. Вместо пары «ключ — значение» массив хранит в одной переменной множество значений. Массив можно объявить квадратными скобками:

```
let dog = ['Бобби', 'Овчарка', 8, 22];
console.log(typeof dog); // object
```

Или создать его оператором `new`:

```
let dog = new Array ('Бобби', 'Овчарка', 8, 22);
```

К значениям массива можно обратиться через порядковый номер элемента. Нумерация начинается с 0 — в примере выводится третий элемент массива:

```
console.log(dog[2]); // выведет 8
```

Если обратиться по номеру, по которому нет значения, результат будет `undefined`.

```
console.log(dog[30]); // undefined
```

`length` выведет длину массива:

```
console.log(dog.length); // 4
```

Арифметические операторы

Оператор сложения `+`

```
console.log(12 + 13) // выведет 25
```

Оператор вычитания `-`

```
console.log(12 - 13) // выведет -1
```

Оператор умножения `*`

```
console.log(15 * 2) // выведет 30
```

Оператор деления `/`

```
console.log(30 / 6) // выведет 5
console.log(3 / 2) // выведет 1.5
console.log(10 / 3) // выведет 3.33333...
```

Приоритет операций

Когда в одной строчке кода встречаются несколько операторов, умножение `*` и деление `/` выполняются первыми. У них приоритет выше, чем у `-` и `+`:

```
console.log(24 / 24 - 12); // выведет -11
console.log(1 + 5 * 2); // выведет 11
```

Оператор группировки `()` — объединяет несколько операторов в группу. Если нужно поднять приоритет какой-то операции, оператор и операнды берут в скобки:

```
console.log(24 / (24 - 12)); // выведет 2
```

Оператор возведения в степень `**` — операнд слева возводится в степень операнда справа:

```
console.log(2 ** 4); // выведет 16
```

Оператор получения остатка от деления `%` — возвращает остаток после деления левого операнда на правый.

```
console.log(5 % 2); // выведет 1 (5 — нечётное)
console.log(6 % 2); // выведет 0 (6 — чётное)
```

Оператор унарный минус `-` — возвращает число с отрицательным знаком.

```
let a = 5;
a = -a;
console.log(a); // выведет -5
a = -a;
console.log(a); // выведет 5
```

Строковые операторы

Конкатенация строк

Строки можно соединять. Процесс объединения — это **конкатенация**. Её выполняют оператором `+`:

```
console.log("маша" + " и " + "медведь") // выведет «маша и медведь»
console.log('Hello,' + ' world!') // выведет Hello, world!
console.log(`Яндекс.` + `Маршруты`) // выведет «Яндекс.Маршруты»
```

Обратные кавычки — грависы могут объединять символы на нескольких строчках. Например:

```
console.log(`ЗЕЛЕНОГРАД 23
ТВЕРЬ 132
С.-ПЕТЕРБУРГ 679`);

// ЗЕЛЕНОГРАД 23
// ТВЕРЬ 132
// С.-ПЕТЕРБУРГ 679
```

«Складывать» можно не только строки или числа, но и строки с числами. Перед этим типы данных автоматически преобразуются к одному общему.

```
console.log(100 + " рублей"); // выведет «100 рублей»
```

Если сложить два числа: одно — в строковом, а другое — в числовом типах, результат приведётся к строковому типу.

```
console.log(100 + "500"); // выведет 100500
```

Если вычитать число из строки или строку из числа, результат приведётся к числовому типу данных.

```
console.log(1000 - "100"); // выведет 900
console.log("1000" - 100); // выведет 900
```

Операторы сравнения и логические операторы

Оператор `==` сравнивает значения и возвращает true, если значения равны, а false — если не равны.

```
console.log(12 == 12); // выведет true
console.log(12 == "12"); // выведет true
console.log("Привет" == "Привет"); // выведет true
console.log("Привет" == "Привет!"); // выведет false
```

Есть обратный оператор — `!=`, или «не равно».

```
console.log(15.1 != 15.1); // выведет false
console.log(12.1 != "12.1"); // выведет false
console.log("Bye" != "Bye"); // выведет false
console.log("Bye" != "Bye!"); // выведет true
```

Строго равно `===`, строго не равно `!==`

```
console.log(12 === 12); // выведет true
console.log(12 === "12"); // выведет false
```

```
console.log(12.22 !== "12.22"); // выведет true
console.log(12.22 !== 12.22); // выведет false
```

Больше `>`, меньше `<`

```
console.log(100 > 100); // выведет false
console.log(100 > "100"); // выведет false
console.log(100 > 99.9); // выведет true
console.log(100 > "99.9"); // выведет true

console.log(100 < 100); // выведет false
console.log(100 < "100"); // выведет false
console.log(100 < 99.9); // выведет false
console.log(100 < "99.9"); // выведет false
```

Больше или равно `>=`, меньше или равно `<=`

```
console.log(100 >= 100); // выведет true
console.log(100 <= "100"); // выведет true
console.log(100 >= 99) // выведет true
console.log(100 <= "99.9"); // выведет false
```

В JavaScript значение false соответствует нулю, а true — единице:

```
console.log(true == 1); // выведет true
console.log(false == 0); // выведет true
console.log(true != false); // выведет true
```

Но типы у них разные, поэтому эти значения не равны.

```
console.log(true === 1); // выведет false
console.log(false === 0); // выведет false
console.log(false !== 0); // выведет true
```

Логические операторы

Логическое ИЛИ `||`

Оператор «логическое ИЛИ» вернёт true, если истинно хотя бы одно выражение из группы выражений.

```
let x = 1, y = 2;
console.log(x > 1 || y > 1); // выведет true
console.log(x > 2 || y > 2); // выведет false
console.log("1.1" === "1.1" || "1.1" === 1.1); // выведет true
```

Логическое И `&&`

Оператор «логическое И» возвращает true, когда истинны все выражения из группы выражений.

```
let x = 1, y = 2;
console.log(x > 1 && y > 1); // выведет false
console.log(x > 0 && y > 0); // выведет true
console.log("1.1" === "1.1" && "1.1" === 1.1); // выведет false
```

Логическое отрицание `!`

Оператор «логическое отрицание» — это операция, которая может сделать ложь из истины и наоборот. Пример:

```
let x = true;
console.log(!x); // выведет false
console.log(!!x); // выведет true
```

Диалоги

Метод alert

Выводит текст в отдельном диалоговом окне в браузере. Пример кода:

```
alert("Hello, world!");
```

Метод prompt

Создаёт диалоговое окно, в котором просит пользователя ввести текст. Например, это выражение запросит у пользователя ввести значение переменной:

```
let x = prompt("Введите X");
```

Если стандартное значение не указано и пользователь оставил поле ввода пустым, то метод вернёт `null`. `null` — это специальное значение в JavaScript, синоним слова «ничего».

```
console.log(null == undefined); // выведет true
console.log(null === undefined); // выведет false
```

Эти значения не равны: у них разные типы.

```
typeof(undefined); // выведет "undefined"
typeof(null); // выведет "object"
```

Метод confirm

Создаёт диалоговое окно, в котором запрашивает у пользователя согласие. Если нажать «Да», вернётся значение true, а если «Нет» — вернётся false:

```
confirm("Пойдём завтра на работу?");
```

Приведение типов

Number() и String()

Можно привести значение к другому типу через специальные методы.

Метод `Number()` преобразовывает данные в число.

```
let text = '255';
console.log(typeof(text)); // выведет string
let red = Number(text);
console.log(typeof(red)); // выведет number
```

`String()` — преобразовывает данные в строку.

```
let red = 255;
console.log(typeof(red)); // выведет number
let text = String(red);
console.log(typeof(text)); // выведет string
```

Приведение унарным плюсом

Если плюс стоит перед строкой, он не складывает операнды. Вместо этого он приводит строку к числу. Такой оператор — это **унарный плюс**.

```
console.log(typeof(+'33')); // number
console.log(typeof('33')); // string
```