1. Арифметические операторы

Арифметические операторы предназначены для выполнения математических операций, они работают с числовыми операндами (или переменными, хранящими числовые значения), возвращая в качестве результата числовое значение.

Если один из операндов является строкой, интерпретатор JavaScript попытается преобразовать его в числовой тип, а после выполнить соответствующую операцию. Если преобразование типов окажется невозможным, будет получен результат `NaN` (не число).

ТАБЛИЦА 1. АРИФМЕТИЧЕСКИЕ ОПЕРАТОРЫ

Оператор/Операция	Описание	Приоритет
`+` Сложение	Складывает числовые операнды. Если один из операндов — строка, то результатом выражения будет строка.	12
`-` Вычитание	Выполняет вычитание второго операнда из первого.	12
`-` Унарный минус	Преобразует положительное число в отрицательное, и наоборот.	14
`*` Умножение	Умножает два операнда.	13
`/` Деление	Делит первый операнд на второй. Результатом деления может являться как целое, так и	13

	число с плавающей точкой.	
`%` Деление по модулю (остаток от деления)	Вычисляет остаток, получаемый при целочисленном делении первого операнда на второй. Применяется как к целым числам, так и числам с плавающей точкой.	13

```JavaScript
var x = 5, y = 8, z;
z = x + y; // вернет 13
z = x - y; // вернет -3
z = - y; // вернет -8
z = x * y; // вернет 40
z = x / y; // вернет 0.625
z = y % x; // вернет 3
```

2. Операторы присваивания

Операторы присваивания используются для присваивания значений переменным. Комбинированные операторы позволяют сохранить первоначальное и последующее значение в одной переменной.

ТАБЛИЦА 2. ОПЕРАТОРЫ ПРИСВАИВАНИЯ

Оператор/Операция	Описание	Приоритет
`=` Присваивание	Используется для присваивания значения переменной.	2
`+=`, `-=`, `*=`, `/=`, `%=` Комбинированный оператор	Выполняет присваивание с операцией. Между первым и вторым операндом выполняется соответствующая операция, затем результат присваивается первому операнду.	2

```JavaScript
var a = 5; // присваиваем переменной a числовое значение 5
var b = "hellow"; // сохраняем в переменной b строку hellow
```

```
var m = n = z = 10; // присваиваем переменным m, n, z числовое значение 10
x += 10; // равнозначно x = x + 10;
x -= 10; // равнозначно x = x - 10;
x *= 10; // равнозначно x = x * 10;
x /= 10; // равнозначно x = x / 10;
x %= 10; // равнозначно x = x % 10;
```

3. Операторы инкремента и декремента

Операции **инкремента и декремента** являются унарными и производят увеличение и уменьшение значения операнда на единицу. В качестве операнда может быть переменная, элемент массива, свойство объекта. Чаще всего такие операции используются для увеличения счетчика в цикле.

ТАБЛИЦА 3. ОПЕРАТОРЫ ИНКРЕМЕНТА И ДЕКРЕМЕНТА

Оператор/Операция	Описание	Приоритет
`++x` Префиксный инкремент	Увеличивает операнд на единицу.	14
`x++` Постфиксный инкремент	Прибавляет к операнду единицу, но результатом выражения будет являться первоначальное значение операнда.	14
`--x` Префиксный декремент	Уменьшает на единицу операнд, возвращая уменьшенное значение.	14
`x--` Постфиксный декремент	Уменьшает на единицу операнд, возвращая первоначальное значение.	14

JavaScript

```
var x = y = m = n = 5, z, s, k, l;
z = ++x * 2; /* в результате вычислений вернет значение z = 12, x = 6, т.е. значение x сначала увеличивается на 1, а после выполняется операц
s = y++ * 2; /* в результате вычислений вернет значение s = 10, y = 6, т.е. сначала выполняется операция умножения, а после в переменной y сохраня
```

```
k = --m * 2; // вернет значение k = 8, m = 4
l = n-- * 2; // вернет значение l = 10, n = 4
```

4. Операторы сравнения

Операторы сравнения используются для сопоставления операндов, результатом выражения может быть одно из двух значений — `true` или `false`. Операндами могут быть не только числа, но и строки, логические значения и объекты. Однако сравнение может выполняться только для чисел и строк, поэтому операнды, не являющиеся числами или строками, преобразуются.

Если оба операнда не могут быть успешно преобразованы в числа или строки, операторы всегда возвращают `false`.

Если оба операнда являются строками/числами или могут быть преобразованы в строки/числа, они будут сравниваться как строки/числа.

Если один операнд является строкой/преобразуется в строку, а другой является числом/преобразуется в число, то оператор попытается преобразовать строку в число и выполнить сравнение чисел. Если строка не является числом, она преобразуется в значение `NaN` и результатом сравнения будет `false`.

Чаще всего операции сравнения используются при организации ветвлений в программах.

ТАБЛИЦА 4. ОПЕРАТОРЫ СРАВНЕНИЯ

Оператор/ Операция	Описание	Приоритет
`==` Равенство	Проверяет две величины на совпадение, допуская преобразование типов. Возвращает `true`, если операнды совпадают, и `false`, если они различны.	9
`!=` Неравенство	Возвращает `true`, если операнды не равны	9

`===` Идентичность	Проверяет два операнда на «идентичность», руководствуясь строгим определением совпадения. Возвращает `true`, если операнды равны без преобразования типов.	9
`!==` Неидентичность	Выполняет проверку идентичности. Возвращает `true`, если операнды не равны без преобразования типов.	9
`>` Больше	Возвращает `true`, если первый операнд больше второго, в противном случае возвращает `false`.	10
`>=` Больше или равно	Возвращает `true`, если первый операнд не меньше второго, в противном случае возвращает `false`.	10
`<` Меньше	Возвращает `true`, если первый операнд меньше второго, в противном случае возвращает `false`.	10
`<=` Меньше или равно	Возвращает `true`, если первый операнд не больше второго, в противном случае возвращает `false`.	10

JavaScript

```
5 == "5"; // вернет true
5 != -5.0; // вернет true
5 === "5"; // вернет false
false === false; // вернет true
1 !== true; // вернет true
1 != true; // вернет false, так как true преобразуется в 1
3 > -3; // вернет true
3 >= "4"; // вернет false
```

5. Логические операторы

Логические операторы позволяют комбинировать условия, возвращающие логические величины. Чаще всего используются в условном выражении `if`.

ТАБЛИЦА 5. ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

Оператор/ Операция	Описание	Приоритет
`&&` Логическое И	Возвращает `true`, только если оба операнда истинны. При выполнении операции сначала проверяется значение первого операнда. Если оно имеет значение `false`, то значение второго оператора не учитывается и результату выражения присваивается `false`.	5
`||` Логическое ИЛИ	Возвращает `true`, если хотя бы один операнд истинен, т.е. проверяет истинность как минимум одного условия.	4
`!` Логическое НЕ	Изменяет значение оператора на обратное - с `true` на `false` и наоборот.	14

JavaS

```
(2 < 3) && (3===3); // вернет true, так как выражения в обеих скобках дают true
(x < 10 && x > 0); // вернет true, если значение x принадлежит промежутку от 0 до 10
```

```
!false; // вернет true
```

6. Побитовые операторы

Побитовые операторы работают с операндами как с 32-битной последовательностью нулей и единиц и возвращают числовое значение, означающее результат операции, записанное в десятичной системе счисления. В качестве операндов рассматриваются целые числа, дробная часть операнда отбрасывается. Побитовые операции могут использоваться, например, при шифровании данных, для работы с флагами, разграничения прав доступа.

ТАБЛИЦА 6. ПОБИТОВЫЕ ОПЕРАТОРЫ

Оператор/ Операция	Описание	Приоритет
`&` Побитовый И	Если оба бита равны `1`, то результирующий бит будет равен `1`. В противном случае результат равен `0`.	8
`|` Побитовый ИЛИ	Если один из операндов содержит в позиции `1`, результат тоже будет содержать `1` в этой позиции, в противном случае результат в этой позиции будет равен `0`.	6
`^` Исключающее ИЛИ	Если одно, и только одно значение содержит `1` в какой-либо позиции, то и результат будет содержать `1` в этой позиции, в противном случае результат в этой позиции будет равен `0`.	7
`~` Отрицание	Выполняется операция побитового отрицания над двоичным представлением значения выражения. Любая позиция, содержащая `1` в исходном выражении, заменяется на `0`. Любая позиция, содержащая `0` в исходном выражении, становится равной `0`. Положительные числа начинаются с `0`, отрицательные - с `-1`, поэтому `~ n == -(n+1)`.	14
`<<` Побитовый сдвиг влево	Оператор сдвигает биты первого операнда влево на число битовых позиций, установленных вторым операндом. Для заполнения позиций справа используются нули. Возвращают результат того же типа, что левый операнд.	11

`>>` Побитовый сдвиг вправо	Оператор сдвигает биты первого операнда вправо на число битовых позиций, установленных вторым операндом. Цифры, сдвинутые за пределы диапазона, удаляются. Самый старший бит (32й) не меняется, чтобы сохранить знак результата. Если первый операнд положителен, старшие биты результата заполняются нулями; если первый операнд отрицателен, старшие биты результата заполняются единицами. Сдвиг значения вправо на одну позицию эквивалентен делению на 2 (с отбрасыванием остатка), а сдвиг вправо на две позиции эквивалентен делению на 4 и т. д.	11
`>>>` Побитовый сдвиг вправо без учета знака	Оператор сдвигает биты первого операнда вправо на число битовых позиций, установленных вторым операндом. Слева добавляются нули независимо от знака первого операнда. Цифры, сдвинутые за пределы диапазона, удаляются.	11

JavaScript

```
var x = 9, y = 5, z = 2, s = -5, result; // 9 эквивалентно 1001, 5 эквивалентно 0101
result = x & y; // вернет 1 (эквивалентно 0001)
result = x | y; // вернет 13 (эквивалентно 1101)
result = x ^ y; // вернет 12 (эквивалентно 1100)
result = ~ y; // вернет -6 (эквивалентно 1100)
result = x << y; // вернет 288 (эквивалентно 100100000)
result = x >> z; // вернет 2 (эквивалентно 10)
result = s >>> z; // вернет 1073741822 (эквивалентно 11111111111111111111111111111110)
```

7. Строковые операторы

Существует несколько операторов, которые работают со строками особым образом.

ТАБЛИЦА 7. СТРОКОВЫЕ ОПЕРАТОРЫ

Оператор/ Операция	Описание	Приори

`+` Конкатенация	Оператор работает слева направо, выполняя объединение строк. Если первый операнд является строкой, последующие операнды будут преобразованы в строки и далее выполнится их объединение.	12
`+=` Конкатенация с присваиванием	Выполняется объединение двух строк и результат присваивается переменной.	12
`>`, `<`, `>=`, `<=`, `==` Сравнение	Строки сравниваются по алфавиту, буквы в верхнем регистре всегда меньше букв в нижнем регистре. Сравнение строк основывается на номерах символов, указанных в стандарте Unicode, где прописные буквы идут раньше, чем строчные.	10

JavaScript

```
"1" + "10"; // вернет "110"
"1" + 10; // вернет "110"
2 + 5 + " цветных карандашей"; // вернет "7 цветных карандашей"
"Цветных карандашей " + 2 + 5; // вернет "Цветных карандашей 25"
"1" > "10"; // вернет false
"10" <= 10; // вернет true
"CCCP" == "cccp"; // вернет false
x = "micro"; x+= "soft"; // вернет "microsoft"
```

8. Специальные операторы

ТАБЛИЦА 8. СПЕЦИАЛЬНЫЕ ОПЕРАТОРЫ

Оператор/ Операция	Описание	Приоритет
`.` Обращение к свойству	Осуществляет доступ к свойству объекта.	15
`,`	Вычисляет несколько независимых выражений, записанных в одну строку.	1

Множественное вычисление		
`[]` Индексация массива	Осуществляет доступ к элементам массива или свойствам объекта.	15
`()` Вызов функции, группировка	Группирует операции или вызывает функцию.	15
`typeof` Определение типа данных	Унарный оператор, возвращает тип данных операнда.	14
`instanceof` Проверка типа объекта	Оператор проверяет, является ли объект экземпляром определенного класса. Левый операнд должен быть объектом, правый - должен содержать имя класса объектов. Результат будет `true`, если объект, указанный слева, представляет собой экземпляр класса, указанного справа, в противном случае - `false`.	10
`in` Проверка наличия свойства	В качестве левого операнда должна быть строка, а правым - массив или объект. Если левое значение является свойством объекта, вернется результат `true`.	10
`new` Создание объекта	Оператор создает новый объект с неопределенными свойствами, затем вызывает функцию-конструктор для его инициализации (передачи параметров). Также может применяться для создания массива.	1
`delete` Удаление	Оператор позволяет удалять свойство из объекта или элемент из массива. Возвращает `true`, если удаление прошло успешно, в противном случае `false`. При удалении элемента массива его длина не меняется.	14
`void` Определение выражения без возвращаемого значения	Унарный оператор, отбрасывает значение операнда и возвращает `underfined`.	14
`?:` Операция условного	Тернарный оператор, позволяет организовать простое ветвление. В выражении участвуют три операнда, первый должен быть логическим значением или преобразовываться в него, а второй и	3

выражения	третий - любыми значениями. Если первый операнд равен `true`, то условное выражение примет значение второго операнда; если `false` - то третьего.	

JavaScript

```
document.write("hello world"); // выводит на экран строку hello world

i = 0, j = 1; // сохраняет значения в переменных

function1(10, 5); // вызов функции function1 с параметрами 10 и 5

var year = [2014, 2015]; // создает массив с элементами

typeof {a:1}; // вернет "object"

var d = new Date(); // создаем новый объект с помощью конструктора Date()
d instanceof Date; // вернет true

var mycar = {make: "Honda", model: "Accord", year: 2005};
"make" in mycar; // вернет true

var obj = new Object(); // создает пустой объект

var food = ["milk", "bread", "meat", "olive oil", "cheese"];
delete food[3]; // удаляет четвертый элемент из массива food

Нажмите здесь, ничего не произойдет

x > 10 ? x * 2 : x / 2; // возвращает значение x * 2, если x > 10, в противном случае x / 2
```

9. Комментарии в JavaScript

Однострочный комментарий: перед текстом комментария нужно поставить символы `//`.

```JavaScript
// Это однострочный комментарий
```

Многострочный комментарий помещается между символами `/*` и `*/`.

```JavaScript
/* Это многострочный
    комментарий */
```