

JavaScript: переменные и операции

**WEB
COURSE
ORT DNIPRO**

ORTDNIPRO.ORG/WEB

1. Наши планы на JavaScript

Наши планы на JavaScript

Переменные и операции

Ветвления (условные операторы)

Циклы / Массивы (структуры данных)

Функции

Объекты

Основы программирования



Управление документом (DOM)

Событийная модель в JavaScript

Разработка интерактивных виджетов.

Прикладное применение

2. Переменные

Задача любой программы – обработка данных

```
1
2
3   let price = 799;
4
5   let quantity = 10;
6
7   let totalCost = price * quantity;
8
9   console.log('Total Cost:', totalCost, 'UAH.');
```

Для хранения **данных** (*информации*), в JavaScript используются **переменные**. **Переменные** можно представить как «коробочку» у которой есть название и в которой хранится какое-нибудь значение. Значением может быть число, строка или другие **типы данных** поддерживаемые JavaScript.

Перед тем как использовать, **переменную** нужно **объявить**. Сказать браузеру, что мы хотим создать еще одну «коробочку» для значений и дать ей имя. Объявляются переменные при помощи ключевого слова **let**.

$x = 1$

Variable = Value

3. Ввод/вывод данных

Ввод/вывод данных

```
1
2 //Варианты ввода данных (без разметки)
3 let userName = prompt('Please enter user name');
4
5 let canDrive = confirm('are you can drive?');
6
7 //Варианты вывода данных (без разметки)
8 alert('Hello');
9
10 console.log('User name:', userName);
11
12 document.write(`<h1>Hello ${userName}</h1>`);
13
14
```

Основной способ ввода данных в **JavaScript** приложения – формы и элементы ввода, а вывод – в разметку документа. В тоже время в **учебных** (и **отладочных**) целях нам будут полезны следующие способы **ввода/вывода** данных (информации).

4. Операции, операторы, операнды и выражения

Оператор присваивания

10

11

12

13

14

```
let a = 2;
```

```
let b = a + 3 * 4;
```



Чтобы указать, что именно нужно записать в переменную используется **оператор присваивания**.

Оператор присвоения берёт то, что справа от него и записывает в переменную имя которой расположено слева от него. Если справа расположено **выражение**, то первым делом оно будет **рассчитано**, и в переменную попадёт уже **результат** расчёта выражения.

Выражения

```
1
2   let a = 3;
3
4   let b = a + 6;
5
6   let c = b + 1;
7
8   c = a + b * c + 7;
9
10  console.log('In Variable C:', c);
11
12
```

По правую сторону от оператора присвоения может быть как конкретное значение, а также может быть **выражение** – формула расчета которую компьютер получит результат который будет записан в переменную имя которой стоит слева от знака присвоения. В **выражении** могут участвовать как и **конкретные значения (константы)** так и другие **переменные**.

Операторы, операнды и операции...

Для выполнения действий (**операций**) над переменными (или значениями) используются **операторы** (которых) существует довольно много).
С некоторыми из них все знакомы, например с арифметическими операторами.

```
12  
13   let a = (2 + 2) * 2;  
14
```

У **операторов** есть **приоритеты**, какой приоритет выше, какой ниже запомнить непросто. Поэтому в случае сомнений какая операция будет первой а какая второй – смело используйте скобки. Принцип их применения такой же как и в математике – скобки повышают приоритет операции в них записанной.

«Скобками программу не испортишь» (с)

Операторы и их приоритеты

Level	Operators
1	() [] .
2	! ~ - ++ --
3	* / %
4	+ -
5	<< >> >>>
6	< <= > >=
7	== !=
8	&
9	^
10	
11	&&
12	
13	?:
14	= += -= *= /= %= <<= >>= >>>= &= ^= =

У операторов есть **приоритеты**, какой приоритет выше, какой ниже запомнить непросто. Поэтому в случае сомнений какая операция будет первой а какая второй – смело используйте скобки. Принцип их применения такой же как и в математике – скобки повышают приоритет операции в них записанной.

5. Типы данных (string & number)

Типы данных (переменных)

```
1
2 let a = 7;
3
4 let b = 8;
5
6 let c = a + b;
7
8 let d = a * b;
9
10 console.log('Value in C:', c); //15
11
12 console.log('Value in D:', d); //56
13
```

```
1
2 let a = '7';
3
4 let b = '8';
5
6 let c = a + b;
7
8 let d = a * b;
9
10 console.log('Value in C:', c); //'78'
11
12 console.log('Value in D:', d); //56
13
```

```
1
2 let a = '7';
3
4 let b = 8;
5
6 let c = prompt('Enter Some Number');
7
8 console.log(typeof(a), typeof(b), typeof(c)); //string, number, string
9
```

В **JavaScript** отсутствует жёсткая **типизация** данных, при которой **тип переменной** определяется при её объявлении. В **JavaScript** **тип переменной** определяется при присваивании ей значения. И может **меняться** при каждом новом присвоении. Мы можем узнать **тип переменной** воспользовавшись функцией **typeof(...)**.

Типы данных (переменных)

Тип данных – пометка для компьютера как относиться к тем или иным данным и **какие операции с ними возможно проводить.**

Тип определяет **возможные значения** и их «смысл», а также **операции** которое возможно выполнять над этими значениями.

undefined, **number**, **string**,
boolean, **function**, **object**, **symbol**, **bigint**

6. Преобразование ТИПОВ

Преобразование типов

Функции **parseInt()/parseFloat()** позволяют преобразовать тип переменной со строкового на числовой (*это возможно если в строке действительно содержатся хоть какие-нибудь цифровые символы, иначе результатом будет значение NaN*). **parseInt()** – работает с целыми числами, **parseFloat()** поддерживает дробные числа.

Также для преобразования типов может быть использован оператор **+** в унарном виде, но в отличии от **parse*-функций** любые нецифровые символы в строке приведут к получению значения **NaN**.

```
3
4 let a = '7.99 $';
5
6 console.log(a, typeof(a)); // '7.99 $' string;
7
8 a = parseFloat(a); // parseInt(a) – для целых чисел;
9
10 console.log(a, typeof(a)); // 7.99 number;
11
```

```
3
4 let a = '7.99';
5
6 console.log(a, typeof(a)); // '7.99' string;
7
8 a = +a; // оператор + в унарном виде;
9
10 console.log(a, typeof(a)); // 7.99 number;
11
```

```
3
4 let a = '7.99 $';
5
6 console.log(a, typeof(a)); // '7.99 $' string;
7
8 a = +a; // оператор + в унарном виде;
9
10 console.log(a, typeof(a)); // NaN number;
11
```

7. NaN

(**N**ot **a** **N**umber)

NaN – Not a Number

```
4   let a = 'hello';
5
6   let b = 7;
7
8   let c = a * b; //NaN - в результате
    выполнения арифметической операции среди
    операндов оказалось значение, которое
    невозможно преобразовать к числу;
9
10  let d = parseFloat(a); //NaN - невозможно
    даже часть строки преобразовать к числу;
11
```

NaN (Not a Number) –

специальное значение типа number которое показывает, что в результате выполнения арифметической операции (или явного преобразования к числу) один из операндов **не удалось успешно преобразовать к числу**. Поскольку **JavaScript** не типизированный язык то ошибок преобразования типов в нём быть не может, потому и существует такая конструкция как **NaN**.

8. Немного практики #1

Задача: Разработать скрипт который на основании дохода физ. лица рассчитает суммы налогов которые ему необходимо заплатить и сколько у него останется после уплаты налогов.

9. Немного практики #2

Задача: Разработать скрипт который рассчитывает **индекс массы тела** пользователя.

Будет полезным

ВЫРАЗИТЕЛЬНЫЙ JAVASCRIPT

3-Е ИЗДАНИЕ

Современное
веб-программирование

Марейн
Хавербеке



Выразительный JavaScript.

Современное веб-программирование

Марейн Хавербеке

*Замечательная книга,
как для введения в
программирование, так и
освоения JavaScript.*

Домашнее задание
Тренируемся!

Программирование ремесло и требует тренировки...

- 1) *Задаётся сторона квадрата. Найти его периметр;*
- 2) *Задаётся длина ребра куба. Найти объем куба и площадь его боковой поверхности;*
- 3) *Задаётся радиус окружности. Найти длину окружности и площадь круга;*
- 4) *Задаются объем и масса вещества. Определить плотность материала этого вещества;*
- 5) *Известны количество жителей в государстве и площадь его территории (в км²). Определить плотность населения в этом государстве.*
- 6). *Даны катеты прямоугольного треугольника. Найти его гипотенузу.*

7) Рассчитать значение y , при любых введенных значениях a :

$$y = \frac{a^2 + 10}{\sqrt{a^2 + 1}}$$



**К следующему
занятию...**

Условные операторы



```
{ if  
  else }
```

Предварительные знания – лучший помощник в обучении, поэтому к следующему занятию жду, что **посмотрите небольшой ролик об условных операторах.**

<https://youtu.be/N6MSUrc8oH4>