

### Основы

Код записывается внутри `<script></script>`

#### Подключение внешнего скрипта:

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

Во внешних скриптах нет тегов `<script>`

#### Комментарии:

```
// - однострочный
```

```
/* ... */ - многострочный
```

Атрибуты **async** и **defer** выполняются асинхронно:

**defer** сохраняет относительную последовательность скриптов

**defer** всегда ждёт, пока весь HTML-документ будет готов

**async** – нет.

### Типы данных

Number	Число
String	Строка
Boolean	Логический тип (true или false)
Null	Значения не существует
Undefined	Значение не присвоено

### ПЕРЕМЕННЫЕ

#### Объявление

```
var x = 10;
```

Можно объявить несколько переменных сразу:

```
var user = 'John', age = 25, message = 'Hello';
```

#### Вывод:

`document.write(x);` - вывод строкой в браузере

`alert(x);` - вывод в диалоговом окне

#### Имена

Имя может состоять из:

- букв
- цифр
- символов \$ и \_

Первый символ не должен быть цифрой

Название переменной должно быть понятно без кода

**Константа** – это переменная, которая никогда не меняется. Их называют большими буквами, через подчёркивание. Например: `var COLOR_RED = "#F00";`

В любом месте программы, где допустимо использовать число или строку, можно также использовать переменную, в которой хранится число или строка.

### ТИП ДАННЫХ - ЧИСЛО

```
var x = 10;
var x = 10.9;
```

Существуют специальные числовые значения Infinity (бесконечность) и NaN (ошибка вычислений)

### ТИП ДАННЫХ - СТРОКА

```
var text = John;
```

По стандарту используются одинарные кавычки

Не использовать вместе либо надо добавлять `\` `"` `'`

`\` - управляющий символ превращает специальные символы в символы строки:

```
var sayHello = 'Hello, world! \' I am a JavaScript programmer.\'';
```

`\` - одинарная кавычка

`\` - двойные кавычки

`\` - обратный слеш

`\n` - новая строка

`\r` - возврат каретки

`\t` - табуляция

`\b` - бекспейс

`\f` - прогон строки

### ОБЪЕКТЫ

Это способ записать в одну переменную много подпеременных.

```
var error = {
  type: 'fatal',
  message: 'Фатальная ошибка'
};
```

Доступ через `.`

```
alert (error.message);
```

### КОМАНДЫ

```
alert(x);
```

вывод переменной в окне

```
var x = prompt ('Введите значение');
```

запрос на ввод

```
var x = confirm ('Подтвердите');
```

выводит ОК или ОТМЕНА

```
console.log (x);
```

вывод информации в консоль

```
document.write(x);
```

вывод переменной в строке браузера

### ОПЕРАТОРЫ И ОПЕРАНДЫ

**Операнд** – то, к чему применяется оператор. Например:  $5 * 2$  – оператор умножения с левым и правым операндами.  
Другое название: «аргумент оператора».

**Унарным** называется оператор, который применяется к одному выражению. Например, оператор унарный минус "-" меняет знак числа на противоположный.

**Бинарным** называется оператор, который применяется к двум операндам. Тот же минус существует и в бинарной форме.

### МАТЕМАТИЧЕСКИЕ ОПЕРАТОРЫ

+	сложение	$25+5=30$
-	вычитание	$25-5=25$
*	умножение	$10*20=200$
/	деление	$20/2 = 10$
%	деление по модулю	$56\%3=2$
var++	постинкремент	var a = 0, b = 10; var a = b++; a=10 и b=11
++var	преинкремент	var a = 0, b = 10; var a = ++b; a=11 и b=11
var--	постдекремент	var a = 0, b = 10; var a = b--; a=10 и b=9
--var	преддекремент	var a = 0, b = 10; var a = --b; a=9 и b=9
x+=y	увеличить x на y	$x=x+y$
x-=y	уменьшить x на y	$x=x-y$
x*=y	умножить x на y	$x=x*y$
x/=y	разделить x на y	$x=x/y$
x%=y		$x=x\%y$

Постфиксная форма `i++` отличается от префиксной `++i` тем, что возвращает старое значение, бывшее до увеличения.

Инкремент/декремент можно применить только к переменной.

### ОПЕРАТОРЫ СРАВНЕНИЯ

==	Равно	$5==10$ ложь
===	Идентично (значение и тип)	$5===10$ ложь
!=	Не равно	$5!=10$ истина
!==	Не идентично	$10!==10$ ложь
>	Больше чем	$10>5$ истина
>=	Больше или равно	$10>=5$ истина
<	Меньше чем	$10<5$ ложь
<=	Меньше или равно	$10<=5$ ложь

Как и другие операторы, сравнение возвращает значение. Это значение имеет логический тип.

Существует всего два логических значения:

- true – имеет смысл «да», «верно», «истина».

- false – означает «нет», «неверно», «ложь».

### СРАВНЕНИЕ ==

Если строка состоит из нескольких букв, то сравнение осуществляется как в телефонной книжке или в словаре. Сначала сравниваются первые буквы, потом вторые, и так далее, пока одна не будет больше другой.

Иными словами, больше – та строка, которая в телефонной книге была бы на большей странице.

При сравнении значений разных типов, используется числовое преобразование. Оно применяется к обоим значениям.

НЕ строгое сравнение `==` приводит операнды к одному типу.

`==` лучше не использовать из-за его неоднозначности.

При сравнении переменных их лучше преобразовать к одному типу.

ПРИМЕРЫ:

```
alert ('2'>1); // true, сравнивается как 2>1
```

```
alert ('01'==1); // true, сравнивается как 1==1
```

```
alert (false==0); // true, false становится числом 0
```

```
alert (true==1); // true, т.к true становится числом 1
```

### СТРОГОЕ СРАВНЕНИЕ ===

В обычном операторе == есть «проблема» – он не может отличить 0 от false:

```
alert (0 == false); // true
```

```
// Та же ситуация с пустой строкой
```

```
alert(" == false); // true
```

Это естественное следствие того, что операнды разных типов преобразовались к числу. Пустая строка, как и false, при преобразовании к числу дают 0.

Для проверки равенства без преобразования типов используются операторы строгого равенства === (тройное равно) и !==.

```
alert (0===false); // false, т.к. типы различны
```

### ОПЕРАТОР typeof

Возвращает тип аргумента.

Нужен чтобы убедиться в типе переменной.

Тсть два синтаксиса - со скобками и без:

- Синтаксис оператора: typeof x

- Синтаксис функции: typeof(x)

Результатом typeof является строка, содержащая тип:

```
typeof undefined // "undefined"
```

```
typeof 0 // "number"
```

```
typeof true // "boolean"
```

```
typeof "foo" // "string"
```

### ЛОГИЧЕСКИЕ ОПЕРАТОРЫ

&& возвращает true, если обе операнды истины

|| возвращает true, если один из операндов истинный

! возвращает true, если операнд ложный, возвращает false, если операнд истинный

Выражения с логическими операторами могут быть совмещены.

```
var isWeekend = false;
var hadShower = true;
var hasApple = false;
var hasOrange = true;
var shouldGoToSchool = !isWeekend && hadShower && (hasApple || hasOrange);
shouldGoToSchool; true
```

### УСЛОВНЫЙ ОПЕРАТОР

```
var isAdult = (age < 18) ?
"Too young" : "Old enough";
```

### СТРОКОВЫЕ ОПЕРАТОРЫ

+ объединяет строки

Если хотя бы один аргумент является строкой, то второй будет также преобразован к строке. Все остальные операторы приводят аргументы к числу.

```
var apples = "2";
```

```
var oranges = "3";
```

```
alert( apples + oranges ); // "23" бинарный плюс складывает строки
```

```
alert( +apples + +oranges ); // 5, число, оба операнда предварительно преобразованы в числа
```

