

ОСНОВЫ JS


Работа с таким языком, в рамках этой статьи, будет располагаться на [бесплатном сайте из сети интернет](#) этот сайт располагает всей нужной документацией для работы я почти любым ЯП (ЯП - Язык Программирования), а именно эта страница позволяет удобно отобразить полезную информацию которую в последствии я буду разбирать.

Прежде чем начать говорить о таком языке как Java Script, хочется углубиться в основы и понять что этот язык из себя представляет, как делаются функции, создаются переменные, и работа с циклами `for` и `while`.

Создание переменных

Все переменные в таком языке объявляются с помощью слов `var` и `let`, почему их 2? Потому что как правило `var` используется для объявления глобальных переменных, которые будут использоваться только в этом классе и его подклассах, а `let` это локальные переменные, которые живут подустим в рамках одной функции или итератор в цикле (итератор это та переменная которая отвечает за то чтобы цикл продолжался, но об этом потом)

Создадим пару переменных и попробуем с ними поработать)



The screenshot shows a web-based JavaScript demo interface. The title bar reads "JavaScript Demo: Statement - For". The main text area contains four lines of code: `1 let a = 5;`, `2 let b = 6;`, `3 let c = a + b;`, and `4 console.log(c);`. Below the code area, there are two buttons: "Run" and "Reset". To the right of the "Run" button, the console output is displayed as `> 11`.

В этом примере мы создали переменные `a` и `b` и `c`, в первых двух у нас находятся числа, в третьей переменной сумма первых и вывод в консоль (так как это браузерный ЯП, то и вывод идет в консоль через команду `console.log()`)

```
JavaScript Demo: Statement - For
1 let a = "5";
2 let b = "6";
3 let c = a + b;
4 console.log(c);
```

Run ›

Reset

> "56"

Сейчас я поменял числа на строки и получил их конкатенацию (**Конкатенация** - сложение)

Функции

Функции создаются через слово `function ... (){}`

```
JavaScript Demo: Statement - For
1 function summ(a,b){
2   console.log(a + b)
3 }
4 let a = "5";
5 let b = "6";
6 summ(a,b)
```

Run ›

Reset

> "56"

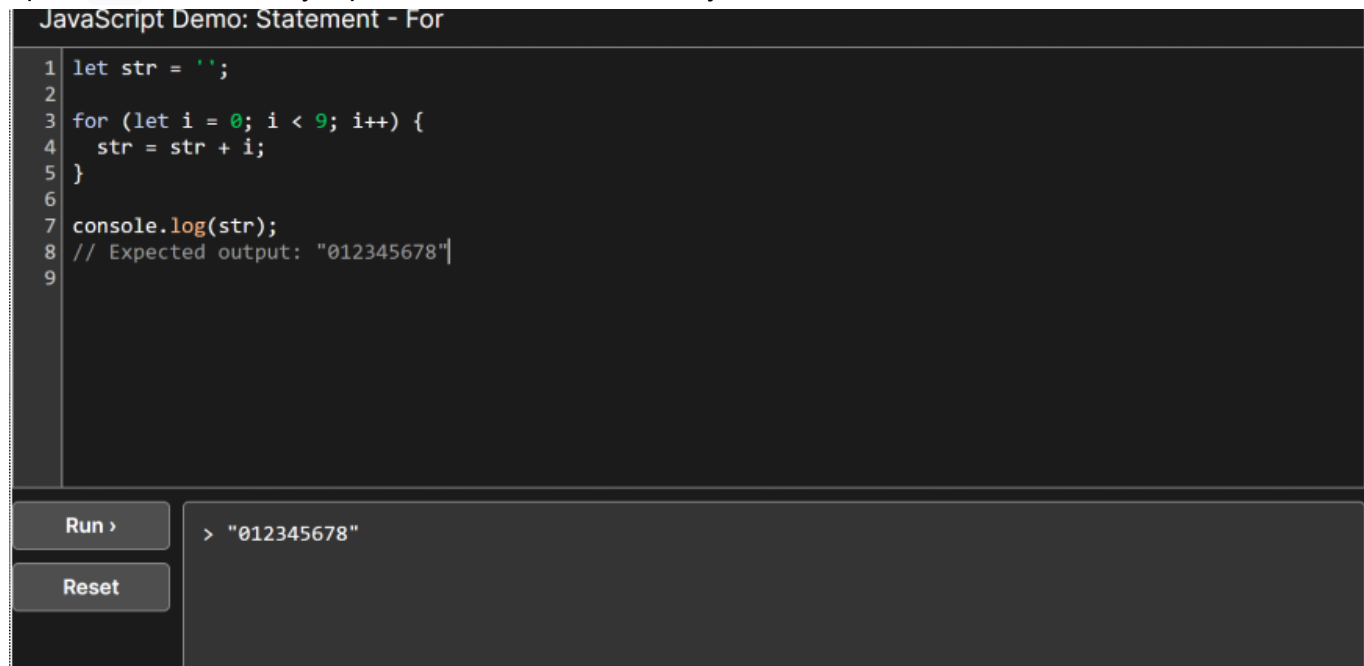
Кстати, вы могли заметить что не везде находится `;` ; это потому что JavaScript особо плевать на это, но плевать если этой строки не будет еще команд, иначе оно сломается с ошибкой. Тогда у вас наверно сразу возник вопрос, если ты говоришь что ему плевать на то что является последним, то тогда почему оно работает даже в функции которая написана в начале, ведь после нее идет основной код, но оно работает, так почему? А я поясню, наша функция она однострочна и все что она делает это конкатенирует 2 полученных значения друг с другом, и если бы я сделал функцию сложнее то мне бы пришлось ставить `;` ; а в данном примере это не требуется, но не стоит о них забывать, так как если ее пропустить, то может сломаться код в

котором тысячи строк.

Теперь вернемся к коду, словом `function` мы сказали JS что сейчас мы будем делать функцию далее идет имя, я ей дал имя `summ` и передал в нее 2 параметра (`a` и `b`) они локальные, почему без `let` или `var` ? Я сам не знаю, но так надо, просто пишем как в `Python` и будет нам счастье. И после уже известных действий я вызвал нашу функцию `summ()` где она выводит в консоль конкатенацию того что мы в нее передали.

Циклы

Цикл `for` самый популярный и много где используемый.



The screenshot shows a code editor titled "JavaScript Demo: Statement - For". The code is as follows:

```
1 let str = '';
2
3 for (let i = 0; i < 9; i++) {
4   str = str + i;
5 }
6
7 console.log(str);
8 // Expected output: "012345678"
9
```

Below the code editor, there are two buttons: "Run" and "Reset". To the right of the "Run" button, the output is displayed: `> "012345678"`.

Данный цикл записывает в переменную `str` каждое значение итератора `i`, строение цикла `for` в JS идентично тому как создается цикл в Java. Но нужно пояснить что вообще тут происходит `for` (объявление итератора; условие прохода; что делаем с итератором после прохождения круга){исполняемый код}. Можешь поиграться с этим и посмотреть что будет выводиться в том или ином случае.

```
JavaScript Demo: Statement - While

1 let n = 0;
2
3 while (n < 3) {
4   n++;
5 }
6
7 console.log(n);
8 // Expected output: 3
9
```

Run › > 3

Reset

Что за `n++`? Это так называемые `дженерики` с их помощью можно сокращать код в разы, позволяя делать приятные глазу счетчики, у них есть вид `++n` и `n++`. Обратимся к нашим собратьям нейросетям чтобы они нам объяснили что это, ибо я сам запутался)

В языке программирования Java, `n++` и `++n` являются операторами инкремента, но они работают немного по-разному.

`n++` - это постфиксный инкремент. Это означает, что сначала будет возвращено текущее значение переменной `n`, а потом произведено увеличение на единицу. Например:

```
int n = 5;
int result = n++;
// result теперь равно 5
// n теперь равно 6
```

`++n` - это префиксный инкремент. Это означает, что сначала переменная `n` увеличивается на единицу, и уже потом возвращается её новое значение. Например:

```
int n = 5;
int result = ++n;
// n теперь равно 6
// result теперь равно 6
```

Разница между этими двумя операторами становится очевидной, когда они используются в выражениях, где порядок выполнения операций важен.

Ну как то так, и в целом, если все было понятно, то можешь ознакомиться с другими моими статейками, удачи тебе в программировании)