



Занятие №2

Множественные условия и циклы



Множественные условия

Несколько условий можно объединить в одно с помощью ключевых символов:

-&& (и)

-|| (или)

Отличие && от ||

|| - проверяет, если ХОТЬ ОДНО из условий верно

&& - проверяет, что КАЖДОЕ из условий верно

Задание

Придумайте число от 0 до 59 –это будут минуты. Запишите в переменную `min`. Определите в какую четверть часа попадает это число (в первую, вторую, третью или четвертую) и вывести четверть на экран.

Циклы

Циклы позволяют в зависимости от определенных условий выполнять некоторое действие множество раз.

В **JAVASCRIPT** есть 5 видов циклов, ознакомившись с ними, вы поймете когда какой использовать.

Цикл while

За ключевым словом «**while**» в круглых скобках указывается условное выражение, после скобок — инструкция для выполнения (тело цикла).

Цикл обеспечивает повторное выполнение тела до тех пор, пока условие остается истинным.

```
while(условие){  
  
    // действия  
  
}
```

С помощью while числа от 1 до 5

```
let i = 1;  
while(i <=5){  
  
    console.log(i);  
    i++;  
}
```

ЦИКЛ WHILE ЗДЕСЬ БУДЕТ ВЫПОЛНЯТЬСЯ, ПОКА ЗНАЧЕНИЕ I НЕ СТАНЕТ РАВНЫМ 6.

Цикл «while»

Цикл «**WHILE**» удобно применять в случаях, когда:

- ЗАРАНЕЕ НЕИЗВЕСТНО КОЛИЧЕСТВО ПОВТОРОВ ТЕЛА ЦИКЛА,
- УСЛОВИЕ ОКОНЧАНИЯ ЦИКЛА ПРЯМО НЕ ЗАВИСИТ ОТ ЦИКЛОВОЙ ПЕРЕМЕННОЙ,
- ЦИКЛ ЗАВИСИТ ОТ ВХОДЯЩИХ ДАННЫХ, НАПРИМЕР, ОТ ДЕЙСТВИЯ ПОЛЬЗОВАТЕЛЯ.

Задание

1. Вывести # столько раз, сколько указал пользователь.
2. Пользователь ввел число, а на экран вывелись все числа от введенного до 0.
3. Запросить число и степень. Возвести число в указанную степень и вывести результат.
4. Запросить 2 числа и найти все общие делители.
5. Посчитать факториал введенного пользователем числа. (что такое факториал посмотрите в интернете)

Цикл DO WHILE

```
do {  
    statement1;  
    statement2;  
    statement3;  
} while (condition)
```

ВТОРАЯ РАЗНОВИДНОСТЬ УСЛОВНЫХ ЦИКЛОВ, — ЦИКЛ С ПОСТУСЛОВИЕМ, — СОЗДАЕТСЯ ПРИ ПОМОЩИ ОПЕРАТОРА «**DO WHILE**»

В ЦИКЛЕ **DO** СНАЧАЛА ВЫПОЛНЯЕТСЯ КОД ЦИКЛА, А ПОТОМ ПРОИСХОДИТ ПРОВЕРКА УСЛОВИЯ В ИНСТРУКЦИИ **WHILE**. И ПОКА ЭТО УСЛОВИЕ ИСТИННО, ЦИКЛ ПОВТОРЯЕТСЯ.

ТО ЕСТЬ НАШ КОД **ОДИН** РАЗ ВЫПОЛНИТЬСЯ В ЛЮБОМ СЛУЧАЕ, НЕЗАВИСИМО ОТ УСЛОВИЯ

DO...WHILE

Здесь код цикла сработает 5 раз, пока `i` не станет равным 5. При этом цикл **do** гарантирует хотя бы однократное выполнение действий, даже если условие в инструкции **while** не будет истинно.

```
let i = 1;
do{
    console.log(i);
    i++;
}while(i <= 5)
```

У ПОЛЬЗОВАТЕЛЯ НУЖНО ПОЛУЧИТЬ ПОДТВЕРЖДЕНИЕ НЕКОТОРОГО ДЕЙСТВИЯ. МЫ НЕ ХОТИМ ИСПОЛЬЗОВАТЬ ДИАЛОГ «**CONFIRM**», Т.К. В НЕМ МОЖНО СЛУЧАЙНО НАЖАТЬ ПРОБЕЛ, «**ENTER**» ИЛИ «**ESC**». НАМ НУЖНО УВЕРЕННОЕ ПОДТВЕРЖДЕНИЕ, ТО ЕСТЬ ПОЛЬЗОВАТЕЛЬ ДОЛЖЕН ВВЕСТИ ЛИБО «**YES**», ЛИБО «**NO**». В ИНОМ СЛУЧАЕ МЫ БУДЕМ ВЫВОДИТЬ ЗАПРОС ПОВТОРНО, ОЖИДАЯ ОДНОГО ИЗ ДВУХ ОТВЕТОВ. ТАК КАК ЗАПРОС НУЖНО ВЫВОДИТЬ КАК МИНИМУМ ОДИН РАЗ, ЦИКЛ С ПОСТУСЛОВИЕМ БУДЕТ ПРЕДПОЧТИТЕЛЬНЫМ

```
<script>
    var txt;
    do
        txt = prompt("Confirm: yes or no")
    while(txt!="yes" && txt!="no")
</script>
```

Задание

1. ПРЕДЛАГАТЬ ПОЛЬЗОВАТЕЛЮ РЕШИТЬ ПРИМЕР

$$2 + 2 * 2$$

До тех пор, пока он не решит его правильно.

2. Делить число 1000 на 2 до тех пор, пока не получится число меньше 50. Вывести это число и сколько делений произвели

Цикл FOR

ЦИКЛ-СЧЕТЧИК (или цикл со счетчиком) ОРГАНИЗУЕТСЯ ПРИ ПОМОЩИ ОПЕРАТОРА «**FOR**».

СИНТАКСИС ОПЕРАТОРА, СОДЕРЖИТ ТРИ ДЕКЛАРАТИВНЫХ БЛОКА:

```
for ([инициализация счетчика]; [условие]; [изменение счетчика]){  
  
    // действия  
}
```

```
for(let i = 0; i<5; i++){  
  
    console.log(i);  
}  
console.log("Конец работы");
```

В ПЕРВОМ БЛОКЕ ИНИЦИАЛИЗИРУЕТСЯ ЦИКЛОВАЯ ПЕРЕМЕННАЯ «i=0»

Во ВТОРОМ УКАЗЫВАЕТСЯ УСЛОВИЕ НА ЭТУ ПЕРЕМЕННУЮ «i<5»

В ТРЕТЬЕМ — АЛГОРИТМ ЕЕ ИЗМЕНЕНИЯ «i++» (УВЕЛИЧЕНИЕ НА 1).

Задание

1. Вывести все числа от 1 до 100, которые кратные указанному пользователем числу.
2. Вывести каждый 4-й элемент из указанного пользователем диапазона. Пользователь указывает минимальное и максимальное значения диапазона.
3. Запросить число и проверить, простое ли оно. Простое число делится без остатка только на себя и на единицу.

Случайные числа

В жизни любого программиста наступает момент, когда приходится иметь дело со случайными числами.

Для генерации псевдослучайных чисел в JavaScript используется метод `random` объекта `Math`.

`Math.random()` — возвращает псевдослучайное число в диапазоне от 0 до 1;

```
// псевдослучайное число  
var res = Math.random();  
alert(res);
```

```
0.20178068431840956  
0.09212343330337536
```

Каким образом можно сгенерировать псевдослучайное число в другом диапазоне, отличном от 0 и 1. Для решения этой задачи используется связка **random** и **floor**

```
// случайное значение от 0 до 9  
alert(Math.floor(Math.random()*10));  
  
// случайное значение от 0 до 11  
alert(Math.floor(Math.random()*11));  
  
// случайное значение от 1 до 10  
alert(Math.floor(Math.random()*10)+1);
```

Нужно сгенерировать не больше десяти случайных чисел из диапазона 1–7.
В случае если выпадает четверка, генерация прекращается.

У нас есть библиотека «**MATH**» у нее есть метод «**RANDOM()**», который выдает дробные случайные числа в диапазоне от 0 до 1. Также есть метод «**ROUND()**», который округляет числа.

```
rnd = Math.round(Math.random() * 6) + 1
```

Задание

ИГРА УГАДАЙ ЧИСЛО.

В ЦИКЛЕ ПОЛЬЗОВАТЕЛЬ ВВОДИТ ЧИСЛО И В ЦИКЛЕ ГЕНЕРИРУЕТСЯ ЧИСЛО ОТ 1 ДО 10.
ПОВТОРЯТЬ ЦИКЛ ДО ТЕЗ ПОР, ПОКА ПОЛЬЗОВАТЕЛЬ И РАНДОМ НЕ ВВЕДУТ ОДИНАКОВОЕ ЧИСЛО.

СЧИТАТЬ КОЛ-ВО ПОПЫТОК ПОЛЬЗОВАТЕЛЯ И В КОНЦЕ ВЫВОДИТЬ.

Задание

1. Напишите скрипт, который запрашивает у пользователя число N и выводит все четные числа от 2 до N или $N-1$, если N нечетное.

Например: ввод 10, вывод 2 4 6 8 10; ввод 7, вывод 2 4 6.

2. Напишите скрипт, который запрашивает у пользователя число N и выводит все числа, на которые делится N , включая число 1.

Например: ввод $N=10$, вывод 1, 2, 5; ввод 11, вывод 1

3. Напишите скрипт, который выводит ровно 10 случайных чисел из диапазона 1–20, кроме тех, которые делятся на 4.

ВОПРОСЫ

ПРАКТИКА



Давайте подведем итоги.