Циклы

Циклы позволяют выполнять фрагмент кода многократно — до тех пор, пока некое условие дает true. Примеры: до тех пор, пока в тарелке есть пища, следует продолжать есть; до тех пор, пока на лице грязь, следует продолжать умываться.

Самый простой из циклов — цикл while. Этот цикл снова и снова выполняет код своего тела, до тех пор, пока заданное условие не перестанет давать true. Используя цикл while, мы имеем в виду следующее: «Продолжай делать это, пока условие дает true. Но если оно даст false, остановись».

Цикл while начинается с ключевого слова while (пока), после которого в скобках стоит условие, а за ним идет тело, заключенное в фигурные скобки:

```
while (condition) {
  console.log("Делаем что-то");
  i++;
}
```

Код, который будет выполняться снова и снова, до тех пор, пока условие (condition) дает true (что-то в этом коде должно влиять на условие, чтобы в какой-то момент оно дало false). Это условие проверяется при каждом повторе цикла.

Аналогично конструкции if, тело цикла while выполняется, если заданное условие дает true. Но, в отличие от if, после того как тело цикла выполнено, условие будет проверено снова, и, если оно все еще дает true, тело цикла начнет выполняться опять. И так будет продолжаться, пока условие не даст false.

Создадим себе проблему! Предположим, у вас проблемы со сном и вы решили посчитать овец. Но раз уж вы познаете азы программирования, почему бы не написать программу, которая будет считать овец за вас?

Задание

1. Создайте переменную sheepCounted со значением 0
2. Теперь создайте цикл while, в условии которого будет проверяться количество посчитанных овечек. Оно не должно превышать 10 штук
3. В теле цикла будем печатать строку в консоль, состоящую из "Посчитано овец: " + sheepCounted + "!"
4. Теперь все там же, в теле цикла, необходимо инкрементировать переменную sheepCounted

Пора баиньки

Мы создали переменную sheepCounted и задали ей значение 0. Дойдя до цикла while в строке , мы проверяем, правда ли, что sheepCounted меньше 10. Поскольку 0 меньше 10, выполняется код в фигурных скобках (тело цикла, которое начинается со строки) и выражение "Посчитано овец: " + sheepCounted + "!" выводится в консоль как «Посчитано овец: 0!». Далее команда sheepCounted++ увеличивает значение sheepCounted на 1, мы возвращаемся к началу цикла, и все повторяется снова:

Посчитано овец: 0!
Посчитано овец: 1!
Посчитано овец: 2!
Посчитано овец: 3!
Посчитано овец: 4!
Посчитано овец: 5!
Посчитано овец: 6!
Посчитано овец: 7!
Посчитано овец: 8!
Посчитано овец: 9!

Тело цикла повторяется, пока sheepCounted не примет значение 10, после чего условие становится ложным (false), ведь 10 не меньше 10. И тогда программа переходит к строке, идущей после цикла.

Задание

1. Чтобы удостовериться в окончании цикла, в строке после него сымитируем свое засыпание, для этого выведем в консоль строку "Я уснулъ" или имитацию храпа

Бесконечность - не предел!

Имея дело с циклами, помните: если условие никогда не даст false, цикл будет повторяться бесконечно (по крайней мере до тех пор, пока вы не закроете страницу в браузере). Например, не будь в теле цикла строчки sheepCounted++;, в sheepCounted всегда был бы 0 и программа печатала бы:

Посчитано овец: 0!
Посчитано овец: 0!
Посчитано овец: 0!
Посчитано овец: 0!
...

Поскольку повторения цикла ничем не ограничены, программа будет печатать эту строку снова и снова, без конца. Это называется бесконечным циклом. Не допускайте таких моментов

На лекциях мы разбирали разные варианты циклов, поэтому вспомним синтаксис цикла for - он упрощает создание циклов, устроенных следующим образом: сначала создается переменная, а затем тело цикла выполняется снова и снова до тех пор, пока условие дает true, причем в конце каждого повтора значение переменной обновляется. Программируя цикл for, мы создаем переменную, задаем условие, указываем, как должна меняться переменная после каждого повтора, — и лишь затем переходим к написанию тела цикла. Например:

```
for (let i  = 3; i <= 5; i++) {
  console.log("Сегодня я получил по информатике оценку: " + i + "!");
}
console.log("Я могу и лучше!");
```

В составе цикла for есть три выражения, разделенные точками с запятой: это инициализация счетчика, проверка условия и инкре/декрементация.

1. Первая часть (let i = 3) выполняется до запуска цикла. Как правило, здесь создают переменную для отслеживания количества повторов.
2. Условие (i <= 5) проверяется перед каждым повтором тела цикла. Если условие дает true, тело выполняется, иначе цикл заканчивает работу. В нашем случае цикл остановится, когда значение i достигнет 5.
3. Третья часть (i++) выполняется после каждого повтора тела цикла. Как правило, здесь изменяют значение переменной цикла. В этом примере мы после каждого повтора инкрементируем переменную

Задание

1. Перепишите овцесчетчик с использованием цикла for

Элементарно

Циклы for удобны, когда нужно сделать что-то определенное количество раз. Можете попробовать изменить счётчик и убедиться в этом. Также цикл for часто используют для перебора всех элементов массива или всех символов строки. Например, вот цикл, который печатает названия всех животных, которые есть в зоопарке:

```
let animals = ["лев", "фламинго", "белый медведь", "удав"];
for (let i = 0; i < animals.length; i++) {
  console.log("В этом зоопарке есть " + animals[i] + ".");
}
```

*.length - метод измерения длины, в случае массива - количества его элементов, в случае строки - количества символов в ней.

В самом цикле i сначала равняется 0, а затем возрастает до значения animals.length - 1, то есть 3. Числа 0, 1, 2 и 3 — индексы элементов в массиве animals. Это значит, что при каждом повторе цикла i принимает значение очередного индекса, а animals[i] соответствует очередному животному из массива animals. Когда в i число 0, animals[i] даст нам строку "лев". Когда в i число 1, animals[i] даст "фламинго" и т. д.

Запустив эту программу, мы увидим:

В этом зоопарке есть лев.
В этом зоопарке есть фламинго.
В этом зоопарке есть белый медведь.
В этом зоопарке есть удав.

Задание

1. Создайте переменную myName и присвойте ей значение вашего имени
2. Создайте цикл for. с итератором i = 0, инкрементируйте его каждый раз, пока i не вырастет до “длины” вашего имени
3. В теле цикла возвращайте в консоль запись формата "В моем имени есть буква " склеенную с порядковой буквой имени - name[i]

Другие варианты применения for

Как вы, может быть, догадываетесь, не обязательно сначала задавать переменной цикла значение 0, а затем каждый раз увеличивать ее на 1. Например, вот как можно напечатать все степени двойки, не превышающие числа 10 000:

```
for (var x = 2; x < 10000; x = x * 2) {
console.log(x);
}
```

Здесь мы присваиваем x значение 2 и увеличиваем его командой x = x * 2, то есть, удваиваем значение x при каждом повторе цикла. В результате x очень быстро возрастает, попробуйте выполнить это в консоли.

Задание

1. Напишите цикл for, который печатает степени тройки, не превышающие 10 000 (программа должна выводить 3, 9, 27 и т. д.). Перепишите это задание, вместо for использовав цикл while. (Подсказка: установите начальное значение перед входом в цикл.)

Конспект «Циклы». Раздел 1

Цикл while. Синтаксис

```
let i = 0;
while (i < 10){
  // Повторяющиеся команды
  i++;
}
```

Цикл for. Синтаксис

```
for (let i = 0; i < 10; i++) {
  // Повторяющиеся команды
}
```

В круглых скобках записывается код управления циклом. Он состоит из трех частей, разделенных ;.

1. Первая часть — подготовительная. Команды отсюда запускаются один раз перед началом работы цикла. Обычно здесь задается исходное значение для переменной-счётчика. Обратите внимание, что в цикле мы создаём переменную-счетчик с помощью let, как в случае с любой другой переменной.

for (let i = 0; i < 5; i = i + 1) { }

2. Вторая часть — проверочная. Она содержит условие и запускается перед каждым новым витком цикла. Если условие возвращает true, цикл делает ещё один виток, иначе цикл завершает свою работу.

for (let i = 0; i < 5; i = i + 1) { }

3. Третья часть — дополняющая, или «закон изменения». Код третьей части запускается после каждого витка цикла. Обычно там изменяется переменная-счётчик.

for (let i = 0; i < 5; i = i + 1) { }

Накопление значений в цикле:

Внутри циклов можно использовать обычные математические операции. Например, сложение:

```
let sum = 0;

for (let i = 1; i <= 5; i++) {
  sum += 2;
  console.log(sum);
}
```

Проверки в теле цикла

Если добавить условие внутрь цикла, то оно будет проверяться на каждой итерации.

```
let sum = 0;
for (let i = 1; i <= 5; i++) {
  if (i > 2) {
    sum += 1;
  }
}
```

Поиск четного числа

Оператор %, или «остаток от деления», возвращает остаток от деления.
10 % 5; // Вернёт 0
12 % 5; // Вернёт 2
7 % 3;   // Вернёт 1
5.5 % 2; // Вернёт 1.5

Если остаток от деления числа на 2 равен 0 — число четное, иначе нечётное.

Сокращенные операторы

В JavaScript есть несколько удобных операторов, которые позволяют сократить код:

звание	ример	налог
кремент (увеличение на единицу)	-+	= i + 1
кремент (уменьшение на единицу)	-	= i - 1
-комбо!	+= 2	= i + 2

Комбинировать можно не только сложение, но и остальные математические операции: вычитание -=, умножение *=, деление /= и нахождение остатка %=.

Задача 1. Прекрасные животные

Напишите цикл for, который изменяет массив животных, делая их прекрасными! Для этого каждому элементу массива вам понадобится переприсвоить значения для каждого индекса, то есть присвоить новые значения уже существующим элементам. Например, сделать первое животное прекрасным можно так:

```
animals[0] = animals[0] + " - прекрасное животное";
```

В таком случае - какое бы животное ни шло под 0 индексом - оно тут же станет прекрасным!

Стартовый код

```
let animals = ["Кот", "Рыба", "Лемур", "Комодский варан"];
```

Задача 2. Генератор случайных паролей

Пишем полезную утилиту. Для этого вам понадобится строка со всеми символами, которые мы хотим видеть в своем пароле. Чтобы выбирать из этой строки случайный символ, можно использовать следующую конструкцию:

```
Math.floor(Math.random() * passAlphabet.length)
```

Так вы получите случайный индекс в строке. Затем, воспользовавшись квадратными скобками, можно получить символ по этому индексу.

Вам нужно добавить цикл while и при каждом его повторе добавлять в строку randomString новый случайный символ — до тех пор, пока длина строки randomString не превысит шесть символов (или любой другой длины на ваш выбор).

Добавлять символ в конец строки можно с помощью оператора +=. После того как цикл закончит работу, выведите получившуюся строку в консоль, чтобы полюбоваться на свое творение!

Стартовый код

```
let passAlphabet= "abcdefghijklmnopqrstuvwxyz0123456789";
let randomString = "";
```

Задача 3. h4ck3r sp34k

Теперь попробуем довольно древнее интернет-развлечение - заменять некоторые буквы похожими на них числами. Например, число «4» похоже на букву «A», «3» похоже на «E», «1» — на «l», а «0» — на «O». Хотя цифры напоминают скорее заглавные буквы, мы будем заменять ими буквы строчные. Чтобы перевести обычный текст на такой «хацкерский язык», понадобится строка с исходным текстом и новая пустая строка для результата. Теперь воспользуйтесь циклом for, чтобы перебрать все символы исходной строки. Встретив букву «a», добавьте к результирующей строке «4». Встретив «e», добавьте «3», встретив «i», добавьте «1», а встретив «o» — «0». В противном случае просто добавляйте к результату исходный символ. И снова оператор += отлично подойдет для добавления символа в конец строки. После завершения цикла выведите результирующую строку в консоль. Если программа работает верно, вы должны увидеть следующее: "j4v4scr1pt 1s 4w3s0m3". После обкатки скрипта, вместо указанной строки можете вставить свои ФИО в транслите и выполнить

Стартовый код

```
var input = "javascript is awesome";
var output = "";
```

Задача 4. Протеины

Представим, что мы снова ударились в ЗОЖ, попробуем правильно запастись протеином на любой период, например, на 15 или 25 дней.

Особенность диеты в том, что количество протеина в чётные и нечётные дни разное. Программа должна считать сколько протеина ты должен выпить за весь тренировочный период.

В чётные дни - 200 грамм. В нечётные - 100 грамм.

Количество дней хранится в переменной days, количество протеина для приёма в чётный день — в переменной evenDayAmount, протеин в нечётный день — в переменной oddDayAmount, а результат необходимо записать в переменную total, которая уже задана.

В этой программе удобно использовать цикл, в котором будет накапливаться количество протеина. Обратите внимание, что в задаче вы работаете с днями, поэтому дни надо считать с 1, а не с 0.

Чтобы задача решалась верно, нужно определять чётные и нечётные дни в периоде. Используйте для этого оператор % (остаток от деления).

Стартовый код

```
let days = 9;
let evenDayAmount = 200;
let oddDayAmount = 100;
let total = 0;
```

Тесты

1. Запускаем тест: рассчитать потреблённый протеин за 5 дней. Количество протеина: 100 гр. в нечётные дни, 200 гр. в чётные дни. ответ 700.
2. Запускаем тест: рассчитать потреблённый протеин за 10 дней. Количество протеина: 100 гр. в нечётные дни, 80 гр. в чётные дни. ответ 900.
3. Запускаем тест: рассчитать потреблённый протеин за 10 дней. Количество протеина: 175 гр. в нечётные дни, 150 гр. в чётные дни. ответ 1625.
4. Запускаем тест: рассчитать потреблённый протеин за 15 дней. Количество протеина: 170 гр. в нечётные дни, 100 гр. в чётные дни. ответ 2060.
5. Запускаем тест: рассчитать потреблённый протеин за 0 дней. Количество протеина: 100 гр. в нечётные дни, 125 гр. в чётные дни. ответ 0.

Задача 5: Сушимся

Продолжаем тему ЗОЖ. Диета и тренировки – вещь утомительная, поэтому знать, сколько ещё дней осталось заниматься, довольно необходимая штука.

Входные данные: вес в начале диеты и вес в конце.

Логика работы: пока я тренируюсь и сижу на диете, то теряю 5% своей массы в день. А тренируюсь я до тех пор, пока мой вес больше желаемого.

Просто посчитай, сколько дней понадобится провести в таком режиме и запиши количество в переменную days. В этой программе удобнее использовать while, так как количество итераций заранее неизвестно.

Обратите внимание, что каждый день теряется 5% от своего веса. При этом вес обновляется каждый день. Например, если сначала вы весите 50кг (то есть 50000 грамм), то на следующий день его вес станет 47 500 грамм. И во второй день уже теряешь 5% от нового веса, то есть от 47500. Учтите это при расчётах.

Стартовый код

```
let initialWeight = 60000; // Исходный вес в граммах
let targetWeight = 50000; // Желанный вес в граммах
```

Тесты

1. Первый тест. Начальный вес 60000 гр., хотим похудеть до 50000 гр. ответ 4 дня.
2. Второй тест. Начальный вес 50000 гр., хотим похудеть до 50000 гр. ответ 0 дней.
3. Третий тест. Начальный вес 70000 гр., хотим похудеть до 50000 гр. ответ 7 дней.
4. Четвёртый тест. Начальный вес 90000 гр., хотим похудеть до 55000 гр. ответ 10 дней.
5. Пятый тест. Начальный вес 100000 гр., хотим похудеть до 57000 гр. ответ 11 дней.