JavaScript: циклы

WEB
COURSE
ORT DNIPRO

ORTDNIPRO.ORG/WEB

1. Циклы – повторение фрагмента кода

Циклы в JavaScript

Циклы, в языках программирования, - способ **повторения фрагмента** (блока) **кода** многократно. Количество повторений определяется **условием цикла**, в зависимости от условия цикл либо делает еще одну **итерацию** (повторяет фрагмент кода в теле цикла), или же **цикл завершает** свою **работу**.

2. Цикл do/while()

Если какие-либо действия нужно повторять, но заранее неизвестно сколько раз

```
1
2       let pinCode;
3
4       do {
5
6           pinCode = prompt('Введите Код');
7
8           if(pinCode == '4321'){
9               alert('Open Door');
10          }else{
11              alert('Code Error');
12          }
13
14      } while(pinCode != '4321');
15
```

Если **код** не подходит, то нужно повторно запросить его у пользователя, и так повторять до тех пор пока не будет введён правильный **код**. Т.е. нам нужен механизм который будет **повторять набор действий до тех пор пока будет верно условие** *(например: пароль не равен «4321»).*

Цикл **do/while()** позволяет повторят код, по условию. Условие проверяется при завершении каждой **итерации** (шага) цикла, и определяет будет ли цикл заходить на еще один «круг».

3. Цикл while()

Циклы – способ многократно повторить фрагмент кода

```
1
2       let currentTemperature  = 25;
3       let temperatureLimit    = 100;
4
5
6       while(currentTemperature < temperatureLimit){
7           console.log(`Temperature ${currentTemperature} °C`);
8
9           currentTemperature++;
10      }
11
12      console.log(`Heating ended at ${currentTemperature} °C`);
13
14
```

Цикл **while()** позволяет повторят код, по условию. Условие проверяется перед началом каждой **итерации** (шага) цикла, и определяет будет ли цикл заходить на этот «круг». Цикл **while** (как и **do/while**) выполняет фрагмент кода пока условие заданное в нём верно (истинно, **true**).

Основное отличие **while** и **do/while**: первый подходит для задач где может не выполниться ни одного «прогона» цикла, а второй – для задач где минимум один **шаг** (**итерация**) цикла должна выполниться.

Ключевой момент применения циклов в JavaScript

В теле цикла должны происходить какие-либо изменения тех переменных которые находятся в условии, иначе цикл будет выполняться вечно

4. Цикл for()

Цикл for – когда известно сколько раз нужно повторить действия

```
1
2       let steps = 10;
3
4       for(let i = 1; i <= steps; i++){
5           console.log(`Step ${i} of ${steps}`);
6       }
7
```

Цикл **for** удобен для тех случаев, когда заранее известно (или можно просчитать на основе уже имеющихся данных), сколько раз нужно будет повторить то или иное действие. Цикл **for** также называют «циклом со счётчиком» – название условное.

5. Операторы continue/break

Операторы break/continue

```
1
2       let steps = 100;
3
4       for(let i = 1; i <= steps; i++){
5
6
7           if( i % 3 == 0 ) {
8               continue;
9           }
10
11          if( i == 17 ) {
12              break;
13          }
14
15          console.log(`Step ${i} of ${steps}`);
16
17      }
18
```

Оператор **break** заставляет цикл завершить свою работу досрочно, прямо в месте вызова. Не имеет смысла без сопутствующего оператора **if/else.**

Оператор **continue** заставляет цикл завершить текущую итерацию, и сразу перейти к следующей. Также не имеет смысла без сопутствующего оператора **if/else.**

6. Немного практики #1

Игра «Угадай число»

Скрипт **загадывает число** (от 1 до 100 включительно) и даёт игроку **10 попыток** его угадать, если пользователь не угадал - скрипт сообщает, что он проиграл и игра сообщает ему какой ответ был правильный. Если игрок угадал игра сообщает, что он выиграл. Для генерации чисел используйте функцию **Math.random**(); Игра при неправильных ответах должна давать подсказки о том число которое ввёл пользователь больше или меньше загаданного.

7. Немного практики #2

Депозит с капитализацией

Есть данные (вводятся пользователем) о **сумме вклада**, **процентной ставке** (годовых) и **сроке вклада в месяцах**. Необходимо рассчитать какой **доход** принесёт депозит с ежемесячной капитализацией процентов в конце срока.

Считать, что проценты начисляются ежемесячно, количество дней в месяце и в году не влияет на результат, налоги также не учитываем.

Домашнее задание
для тренировки

Домашнее задание #B1

Месяц	Задолженность по кредиту	Погашение кредита	Проценты по кредиту	Комиссии	Выплаты в месяц
1	1000.00	83.40	20.90	0.00	104.30
2	916.60	83.40	19.10	0.00	102.50
3	833.20	83.40	17.40	0.00	100.80
4	749.80	83.40	15.70	0.00	99.10
5	666.40	83.40	13.90	0.00	97.30
6	583.00	83.40	12.20	0.00	95.60
7	499.60	83.40	10.50	0.00	93.90
8	416.20	83.40	8.70	0.00	92.10
9	332.80	83.40	7.00	0.00	90.40
10	249.40	83.40	5.20	0.00	88.60
11	166.00	83.40	3.50	0.00	86.90
12	82.60	82.60	1.80	0.00	84.40
Итого		1000.00	135.90	0.00	1135.90

Необходимо реализовать расчёт схемы выплаты кредита по классической схеме. Параметры которые определяют кредит (данные на входе): **сумма**, **процентная ставка** (% годовых) и **срок кредитования** (в месяцах). Никакие комиссии и другие платежи не учитываем.

В схеме должна быть информация за каждый месяц: сколько нужно погасить **тела кредита**, сколько заплатить **процентов** (и **сумму** тело + проценты) и сколько остаётся **долга** по телу кредите после этого платежа.

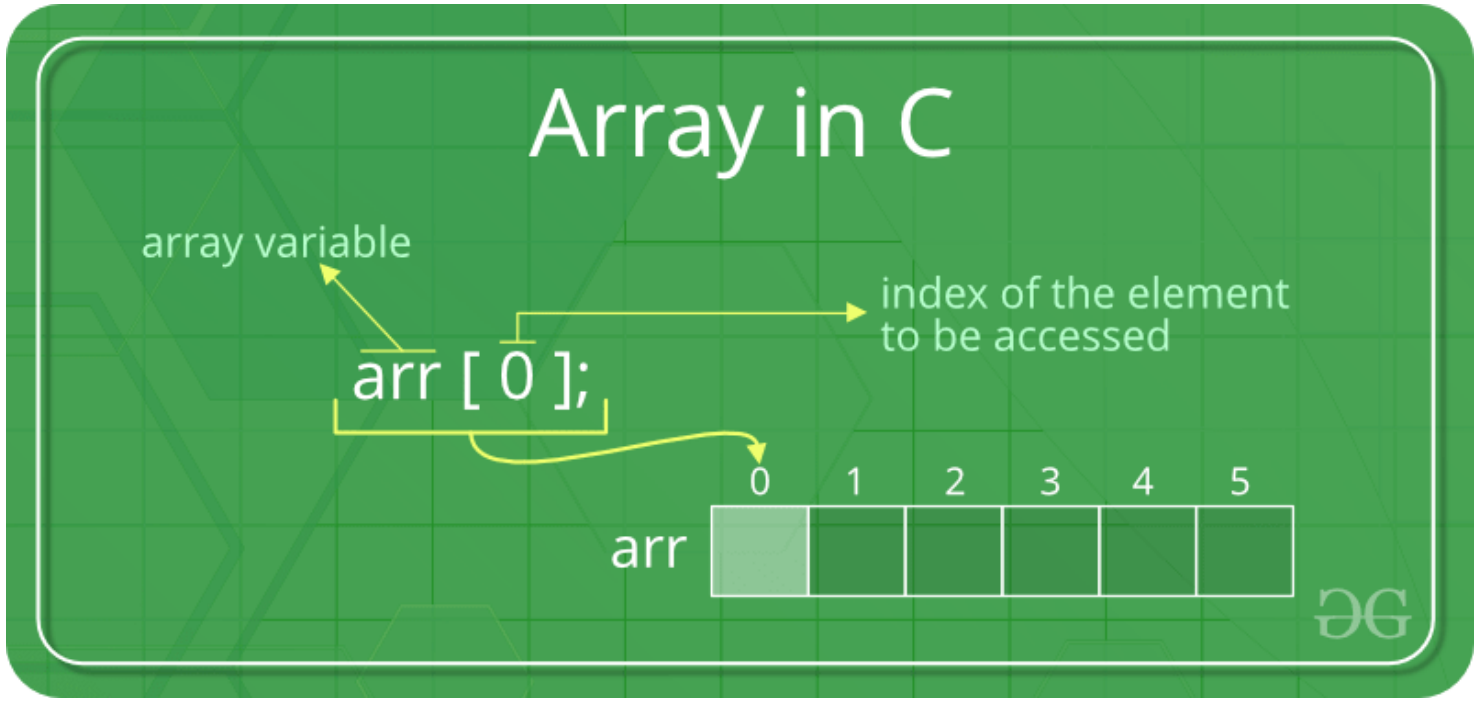
В конце необходимо вывести какая будет **переплата** по кредиту.

Все денежные суммы должны быть округлены до 2х знаков после запятой.

Пример функционала:
https://fin-calc.org.ua/ru/credit/calculate/

На следующем занятии

JS: циклы и массивы



Хранения и обработка наборов данных