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. Вложенные циклы

Вложенные циклы

```
2
3       for(let i = 1; i <=10; i++){
4           for(let j = 1; j <= 10; j++){
5
6               let multiply = i * j;
7
8               console.log(`${i} x ${j} = ${multiply}`);
9
10          }
11      }
12
```

Циклы могут быть **вложены** в тело другого (внешнего) цикла. И на каждом шаге внешнего цикла будет каждый раз полностью проходить все свои шаги вложенный (внутренний) цикл. Любой тип цикла может быть как внешним так и вложенным.

6. 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.**

7. 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.

8. 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

0									
1	2	3	4	5	6	7	8	9	10
11	12	13	14	15	16	17	18	19	20
21	22	23	24	25	26	27	28	29	30
31	32	33	34	35	36	37	38	39	40
41	42	43	44	45	46	47	48	49	50
51	52	53	54	55	56	57	58	59	60
61	62	63	64	65	66	67	68	69	70
71	72	73	74	75	76	77	78	79	80
81	82	83	84	85	86	87	88	89	90
91	92	93	94	95	96	97	98	99	100

Пользователь вводит число, **необходимо определить простое** ли это **число** (т.е. не имеет делителей кроме себя и единицы).

Домашнее задание #B1

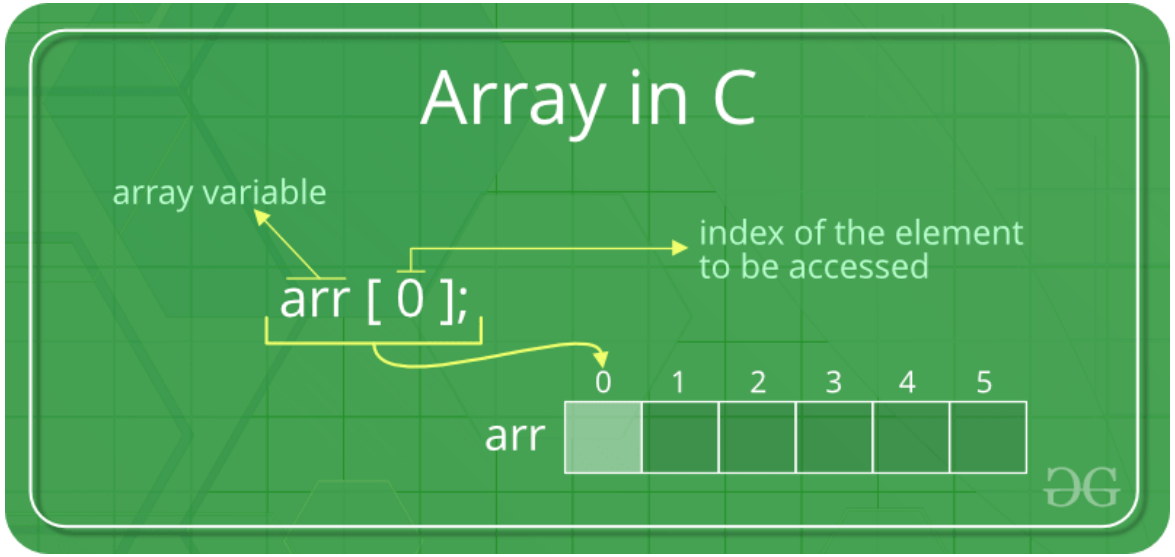
2	3	5	7	11	13	17	19
23	29	31	37	41	43	47	53
59	61	67	71	73	79	83	89
97	101	103	107	109	113	127	131
137	139	149	151	157	163	167	173
179	181	191	193	197	199	211	223
227	229	233	239	241	251	257	263
269	271	277	281	283	293	307	311
313	317	331	337	347	349	353	359
367	373	379	383	389	397	401	409
419	421	431	433	439	443	449	457
461	463	467	479	487	491	499	503
509	521	523	541	547	557	563	569
571	577	587	593	599	601	607	613
617	619	631	641	643	647	653	659
661	673	677	683	691	701	709	719
727	733	739	743	751	757	761	769
773	787	797	809	811	821	823	827
829	839	853	857	859	863	877	881
883	887	907	911	919	929	937	941
947	953	967	971	977	983	991	997

Скрипт должен вывести **список простых чисел от 3 до 1000**.

Здесь пригодятся вложенные циклы.

К следующему занятию…

Следующая тема: циклы и массивы



Предварительные знания – лучший помощник в обучении, поэтому к следующему занятию жду, что **посмотрите небольшой ролик о массивах.**

https://youtu.be/cTeBlVn_lyk