Циклы

Вызыает исполнение кода в области несколько раз

Что такое цикл?

- Цикл это **управляющая конструкция** которая позволяет **повторять утверждение несколько раз**
  - Позволяетт вызывать код в блоке определенное количество раз
  - Позволяет вызывать код в блоке до тех пор пока выполняется условие
  - Позволяет перебирать каждый из элементов выбранной коллекции
- Циклы могут быть бесконечными.

цикл `while`

- Самый просто и часто используемый тип циклов
- Содержит **условие повторения**
  - Так же называется **условным циклом**
  - Типизируется не только `Булквым` значением
  - Вызывает `true` или `false`
    - `5`, `'non-empty'`, `{}`, интерпретируется как 'true'
    - `0`, `''`, `null`, `undefined` будет интерпретировано как false

```
while (condition) {
    statements;
}
```

`while` цикл. Как это работает?

- Базовый пример

```
let counter = 0;
while (counter < 10) {
    console.log('Number : ' + counter);
    counter += 1;
}
```

- Вычисляет и печатает сумму первых N натуральных чисел

```
let n = 123,
    sum = 0,
    operands = 'Сумма 123';

while(n > 0) {
    sum += n;
    n -= 1;
    operands += '+' + n;
}

console.log(operands + ' = ' + sum);
```

- Проверяет является ли число простым или нет.

```
const n = 123,
    maxDivider = Math.sqrt(n);

let divider = 2,
    prime = true;

while (prime && (divider <= maxDivider)) {
   if (!(n % divider)) {
       prime = false;
   }
   divider += 1;
}
```

```
console.log(prime);
```

Использование break оператора

- `break` прерывает выполнение цикла. И вызывает код следующий далее за циклом.

```
let n = 10,
    fact = 1,
    factStr = 'n! = ';

while (1) { //infinite loop
  if (n === 1) {
      break;
  }

  factStr += n + '*'
  fact *= n;
  n -= 1;
}

factStr += '1 = ' + fact;
console.log(factStr);
```

do-while` цикл

- Имеет другую структуру:
- Утверждение в блоке выполняется
  - Пока условие выозвращает `true`
- Цикл всегда **выполняется хотя бы один раз**

```
do {
    statements;
} while (condition);
```

- Вычислить `N!`

```
let fact = 1,
    factStr = 'n! = ';

do {
    fact *= n;
    factStr += n + '*'
    n -= 1;
} while (n);

factStr += ' = ' + fact;
console.log(factStr)
```

- Вычисление производить в диапазне `[N..M]`

```
let number = n,
    product = 1,
    productStr = '';

do {
  product *= number;
  productStr += number;

  if (number != m) {
      productStr += '*';
  }

  number += 1;
} while (number <= m);

productStr += ' = ' + product;
console.log(productStr);
```

- Конвертиртировать число с десятичного представления до бинарного

```
let dec = 125,
    result = '';

do {
    result = (dec & 1) + result;
    dec >>= 1;
} while(dec > 0)

console.log(result);
```

Цикл `for`

- Обычнно синтаксис цикла `for` выглядит так:
- Содержит
  - **Проинициализованную переменную - счетчик**
  - **Проверяет выражение на логическое соответствие**
  - **Обновляет счетчик**
  - **Цикл заключается в блоке**

```
for (инициализация; условие; обновление) {
    утверждение;
}
```

Выражение инициализации

```
for (let number = 0; number < 10; number += 1) {
    // Можно использовать номер сдесь
}
// Не использовать номер сдесь
```

- Вызывается единожды, перед входом в цикл
- Обычно приименяется декларируется переменная счетчик
  - В выражении инициализапции может быть задекларировано множество переменных

Условие цикла

- Вызывается перед каждой итерацией
  - Если возвращается **true**, происходит переходт в тело цикла
  - Если возвращается **false**, цикл завершается

```
  for (let number = 0; number < 10; number += 1) {
    // Можно использовать номер сдесь
}

// Не использовать номер сдесь
```

Выражение обновления

```
for (let number = 0; number < 10; number += 1) {
    // Можно использовать номер сдесь
}

// Не использовать номер сдесь
```

- Вызывается в каждой итерации **после** завершения выполнения операций в теле цикла
- Обычно используется для обновления счетчика
  - `for` цик поддерживает список декларированных переменных, разделенным символов `,` (запятая) operator
- Вывести все натуральные числа вплоть до `N`

```
const N = 10;
```

```
for (let number = 0; number < N; number += 1) {
    console.log(number + ' ');
}
```

- Вычислить факториал `N!`

```
let factorial = 1;
const N = 5;

for (let i = 1; i <= N; i += 1) {
    factorial *= i;
}
```

- Полный пример использования цикла `for` с использованием нескольких переменных счетчик:

```
for (let i = 1, sum = 1, N = 128; i <= N; i *= 2, sum += i) {
    console.log('i=' + i + ', sum=' +sum);
}
```

```
// output
i=1, sum=1
i=2, sum=3
i=4, sum=7
i=8, sum=15
...
```

- Возведение `N` в степень `M` (обозначается как `N^M`):

```
const N = 3,
      M = 5;

let result = 1;

for (let i = 0; i < M; i += 1) {
      result *= N;
}

console.log(result);
```

Что насчет вложенных циклов?

- **комопозиция циклов** называется вложенными циклами
    - Цикл находится внутри другого цикла

```
for (let i = 0; i < 10; i += 2) {
    for (let j = 0; j < 20; j += 1) {
        while(i !== j) {
            console.log(i);
            j += 1;
        }
    }
}
```

- Треугольник циклов

```
1
1 2
...
1 2 3 ... N
```

```
const N = 7;
let result = '';

for(let row = 1; row <= N; row += 1) {
   for(let column = 1; column <= row; column += 1) {
      result += column + ' ';
   }
```

```
   result += '\n';
}

console.log(result);
```

- Вывести все натуральные числа в диапазоне `[N..M]`

```
const N = 5, M = 20;
let result = '';

for (let number = N; number <= M; number += 1) {

    const maxDivider = Math.sqrt(number);
    let isPrime = true,
        divider = 2;

    while (divider <= maxDivider) {
        if (!(number % divider)) {
            isPrime = false;
            break;
        }
        divider += 1;
    }
    if (isPrime) {
        result += number + ' ';
    }
}
```

- Вывести все числовые индексы в формате **ABCD** которые **A+B = C+D** (известны как "счастливые номера")

```
for (a =1 ; a <= 9; a += 1) {
  for (b = 0; b <= 9; b += 1)
    for (c = 0; c <= 9; c += 1)
      for (d = 0; d <= 9; d += 1)
        if (a + b == c + d)
            console.log(`${a} ${b} ${c} ${d}`);
  }
```

- Вывести все 6/49 комбинации -

```
  for (let i1 = 1; i1 <= 44; i1 += 1)
  for (let i2 = i1 + 1; i2 <= 45; i2 += 1)
    for (let i3 = i2 + 1; i3 <= 46; i3 += 1)
      for (let i4 = i3 + 1; i4 <= 47; i4 += 1)
        for (let i5 = i4 + 1; i5 <= 48; i5 += 1)
          for (let i6 = i5 + 1; i6 <= 49; i6 += 1)
            console.log(
            `${i1}, ${i2}, ${i3}, ${i4}, ${i5}, ${i6}`);
```

`цикл for-in`

- `for-in` цикл **перебирает все параметры объекта**
  - Когда объект является **массивом**, **nodeList - коллекция узлов** or **liveNodeList - живая коллекция узлов**, `for-in` прохводит по всем **элементам**
  - Если объект не является коллекцией, `for-in` итерация происходит по **его свойствам**
- Итерация по листу параметров **в документе**

```
// propName это строка - имя параметра
for (const propName in document) {
    console.log(document[propName]);
}
```

- Проход по всем элементам массива

```
const arr = [1, 2, 3, 4, 5, 6];

for (const index in arr) {
```

```
    console.log(arr[index]);
}
```

`for-of` loop

- `for-of` цикл **перебирает элементы в массиве**
  - Может использоваться только в массивах, и массиво-подобных обьектах

```
const arr = ['One', 'Two', 'Three', 'Four'];

for(const n of arr) {
  console.log(n);
}
```

- for-of цикл является **частью ECMAScript 6** станданрта
  - Поддерживается всеми современными браузерами