Telerik Academy

Loops

Execute Blocks of Code Multiple Times

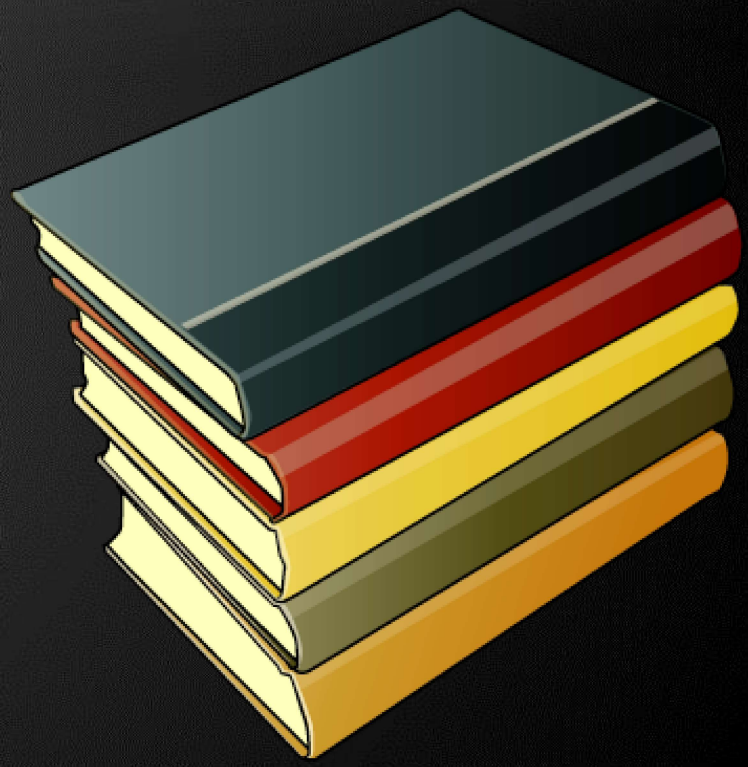
Javascript Fundamentals

Telerik Software Academy

https://telerikacademy.com

Telerik Academy

Table of Contents



Telerik Academy

What is a loop?

- A loop is a control statement that allows repeating the execution of a block of statements
  - May execute a code block fixed number of times
  - May execute a code block while given condition holds
  - May execute a code block for each member of a collection
- Loops that never end are called an infinite loops

Telerik Academy

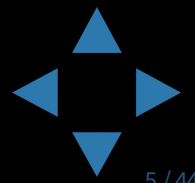
while loop

Telerik Academy

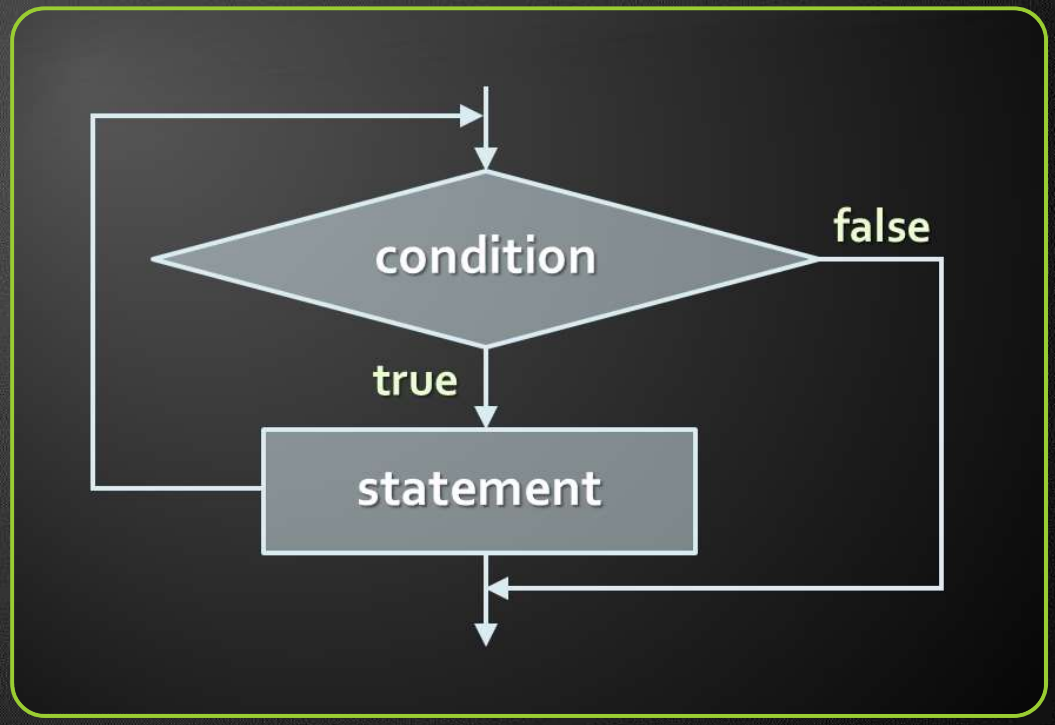
How to use a `while` loop?

- The simplest and most frequently used loop
- Has a repeat conditions
  - Also called loop condition
  - Is not necessary strictly a `Boolean` value
  - Is evaluated to `true` or `false`
    - `5`, `'non-empty'`, `{}`, etc.. are evaluated as true
    - `0`, `''`, `null`, `undefined` are evaluated as false

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

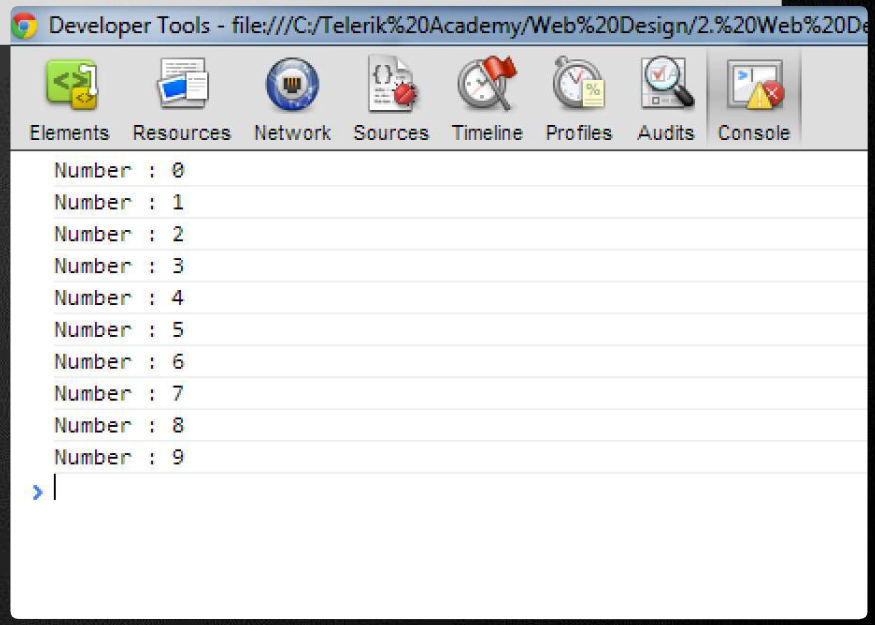
while loop – How It Works?



Telerik Academy

while loop – *Examples*

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

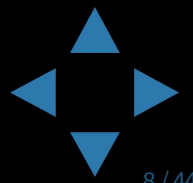


Telerik Academy

Sum 1..N – *Example*

- Calculate and print the sum of the first N natural numbers

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

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

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

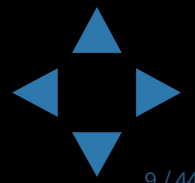


Telerik Academy

Prime Number – *Example*

- Checking whether a number is prime or not

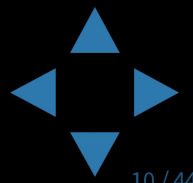
```
const n = 123,
    maxDivider = Math.sqrt(n);

let divider = 2,
    prime = true;

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

console.log(prime);
```



Telerik Academy

While Loop

Demo



Telerik Academy

Using break Operator

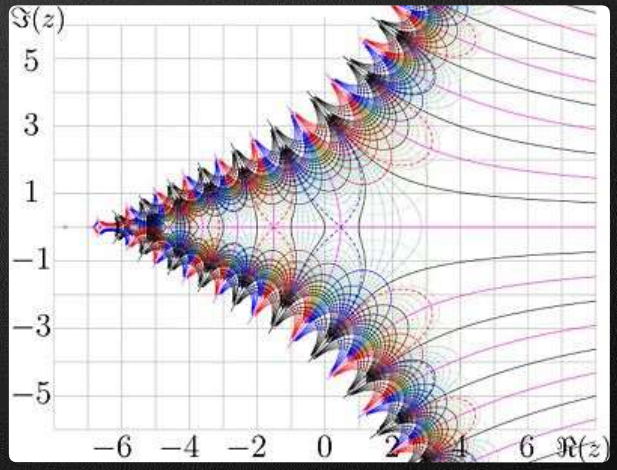
- break operator exits the inner-most loop

```
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);
```

Telerik Academy

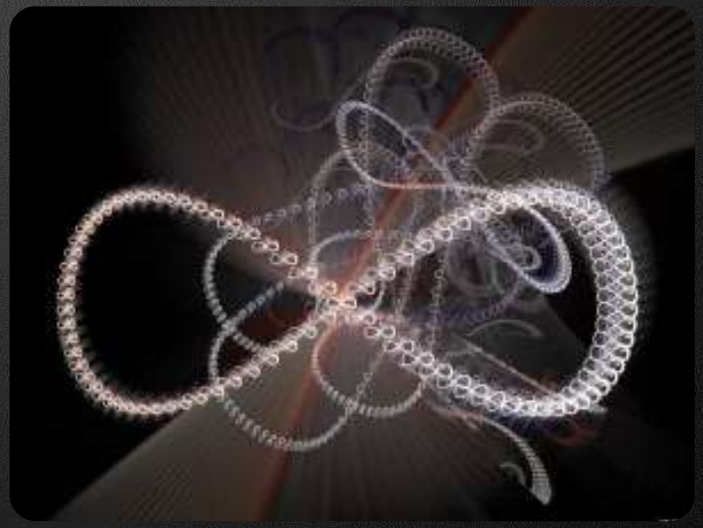
Calculating Factorial

Demo



Telerik Academy

do-while loop



Telerik Academy

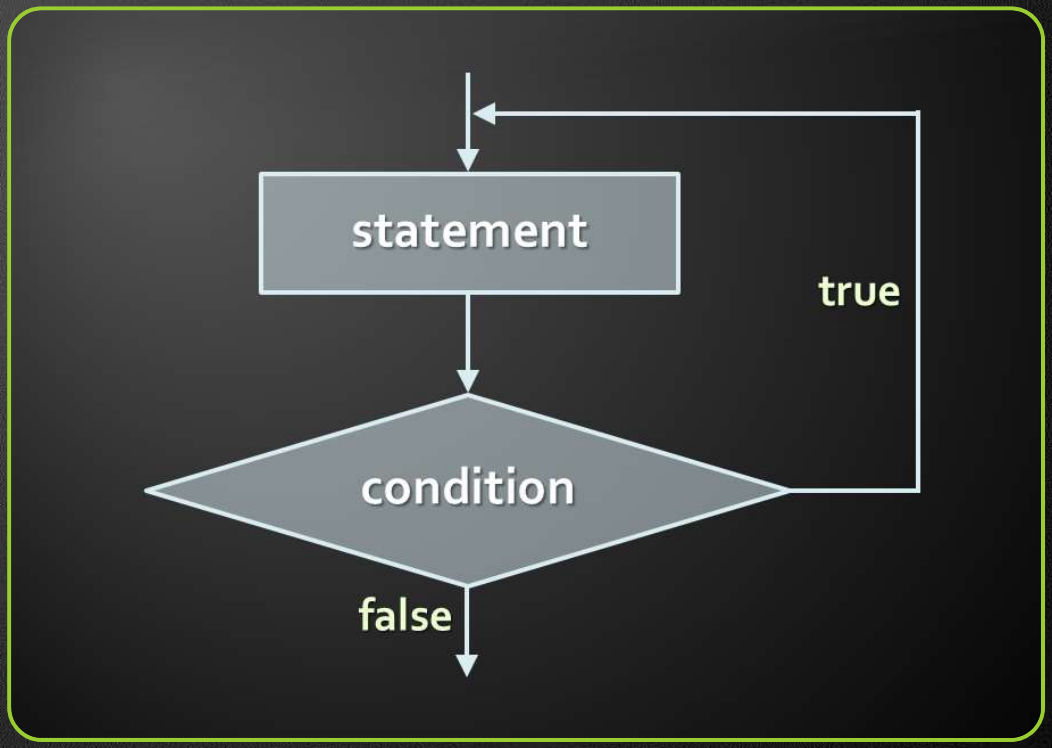
Using do-while loop

- Another loop structure is:
- The block of statements is repeated
  - While the boolean loop condition holds
- The loop is always executed at least once


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

Telerik Academy

do-while statement



Telerik Academy

do-while - *Examples*

- Calculating `N!`
- Calculating the product of the numbers in the range `[N..M]`
- Converting a number from decimal to binary

Telerik Academy

Factorial – *Example*

- Calculating N!

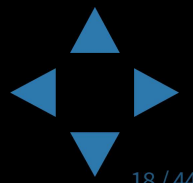
```
let fact = 1,
    factStr = 'n! = ';

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

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


Telerik Academy

Product[N..M] – *Example*

- Calculating the product of all numbers in the range [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);
```

Telerik Academy Decimal to binary - *Example*

- Converting a decimal number to it's binary representation

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

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

console.log(result);
```

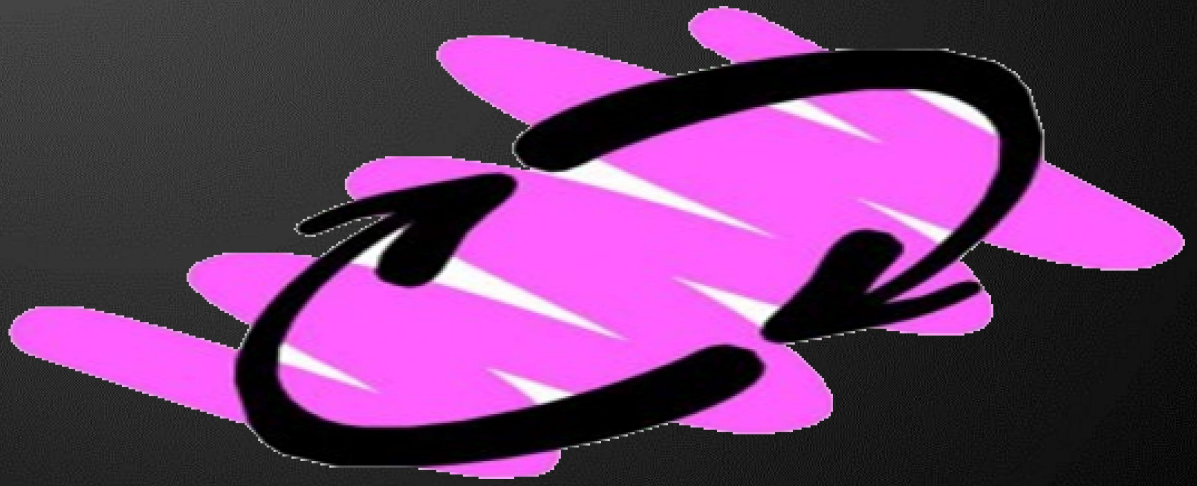
Telerik Academy

do-while loop

Demo

Telerik Academy

for loops

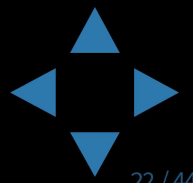


Telerik Academy

for loops

- The typical for loop syntax is:
- Consists of
  1. Initialization statement
  2. Test expression that is evaluated to boolean
  3. Update statement
  4. Loop body block

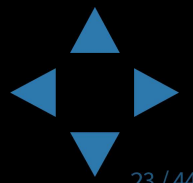
```
for (initialization; test; update) {
    statements;
}
```

The Initialization Expression

```
for (let number = 0; number < 10; number += 1) {
    // Can use number here
}

// Cannot use number here
```

- Executed once, just before the loop is entered
  - Like it is out of the loop, before it
- Usually used to declare a counter variable
  - Multiple variables can be declared in the initialization statement

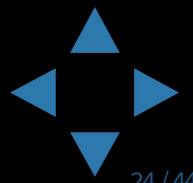


Telerik Academy

The Test Expression

- Evaluated before each iteration of the loop
  - If evaluated true, the loop body is executed
  - If evaluated false, the loop ends
- Used as a loop condition

```
  for (let number = 0; number < 10; number += 1) {
    // Can use number here
}

// Cannot use number here
```

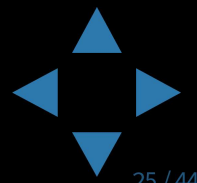



Telerik Academy

The Update Expression

```
for (let number = 0; number < 10; number += 1) {
    // Can use number here
}

// Cannot use number here
```

- Executed at each iteration after the body of the loop is finished
- Usually used to update the counter
  - `for` loops support multiple update statements, separated by the `,` (comma) operator



Telerik Academy

for loop - *Examples*

- Print all natural numbers up to N
- Calculating N!
- Raising N to the power of M

Telerik Academy

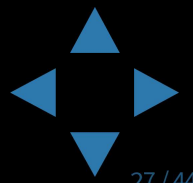
Simple for Loop – *Example*

- A simple for-loop to print the numbers `[0..9]`:

```
const N = 10;

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

- A simple for-loop to calculate `N!`:

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

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

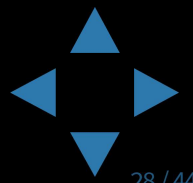
Telerik Academy

Complex for Loop – *Example*

- Complex `for` loops could have several counter variables:

```
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
...
```

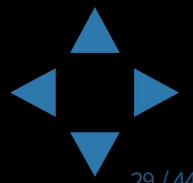



Telerik Academy

N^M – *Example*

- Raising N to power M (denoted as N^M):

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

let result = 1;

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

console.log(result);
```

Telerik Academy

for loop

Demo

Telerik Academy

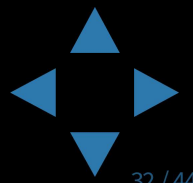
Nested loops

Telerik Academy

What Is Nested Loop?

- A composition of loops is called a nested loop
  - A loop inside another loop
- *Example*:

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

Telerik Academy

Nested Loops - *Examples*

- Triangle of numbers
- Print all prime numbers in the range `[N..M]`
- Happy numbers
- Print all 6/49 combinations

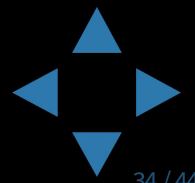
Telerik Academy

Triangle – *Example*

- Print the following triangle:

```
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);
```

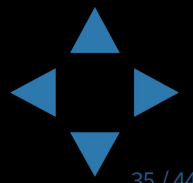



Telerik Academy

Primes[N..M] – *Example*

- Print all prime numbers in the interval [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 + ' ';
    }
}
```



Happy numbers – *Example*

- Print all four digit numbers in format ABCD such that A+B = C+D (known as happy numbers)

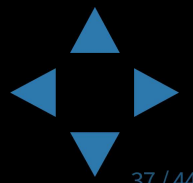
```
for (a =1 ; a <= 9; a += 1) {
  for (b = 0; b <= 9; b += 1)
    for (c = 0; c <= 9; c += 1)
      for (d = 0; d <= 9; d += 1)
        if (a + b == c + d)
            console.log(`${a} ${b} ${c} ${d}`);
  }
```

Telerik Academy

TOTO 6/49 – *Example*

- Print all combinations from TOTO 6/49

```
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}`);
```

Telerik Academy

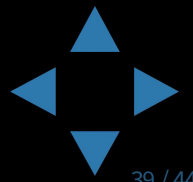
Nested loops

Demo

Telerik Academy

for-in loop

- for-in loop iterates over the properties of an object
  - When the object is array, nodeList or liveNodeList, for-in iterates over their elements
  - When the object is not a collection, for-in iterates over its properties



Telerik Academy

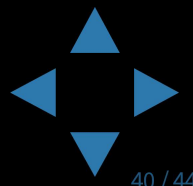
for-in - *Example*

- Iterating over the properties of document

```
// propName is a string - the name of the property
for (const propName in document) {
    console.log(document[propName]);
}
```

- Iterating over the elements of an array

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

for (const index in arr) {
    console.log(arr[index]);
}
```



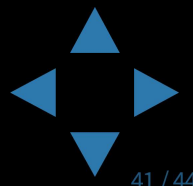
Telerik Academy

for-of loop

- for-of loop iterates over the elements in an array
  - Can be used only on arrays, or array-like objects
    - i.e. the arguments object

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

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

- The for-of loop is part of the ECMAScript 6 standard
  - Supported in all modern browsers



Telerik Academy

for-in and for-of

Demo