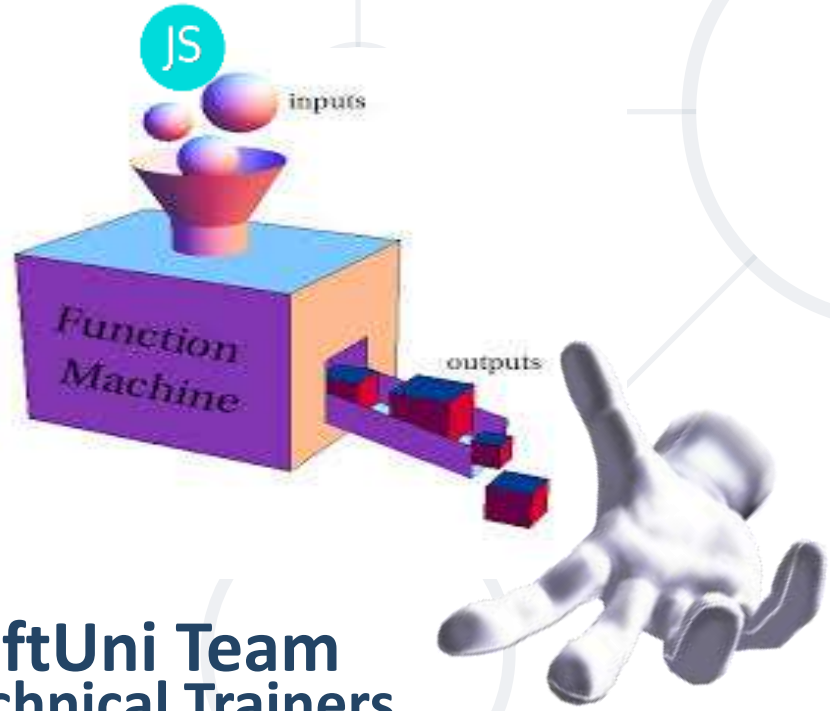
Functions and Logic Flow

Functions, Loops and Scope



SoftUni Team
Technical Trainers

Software University

http://softuni.bg

Table of Contents

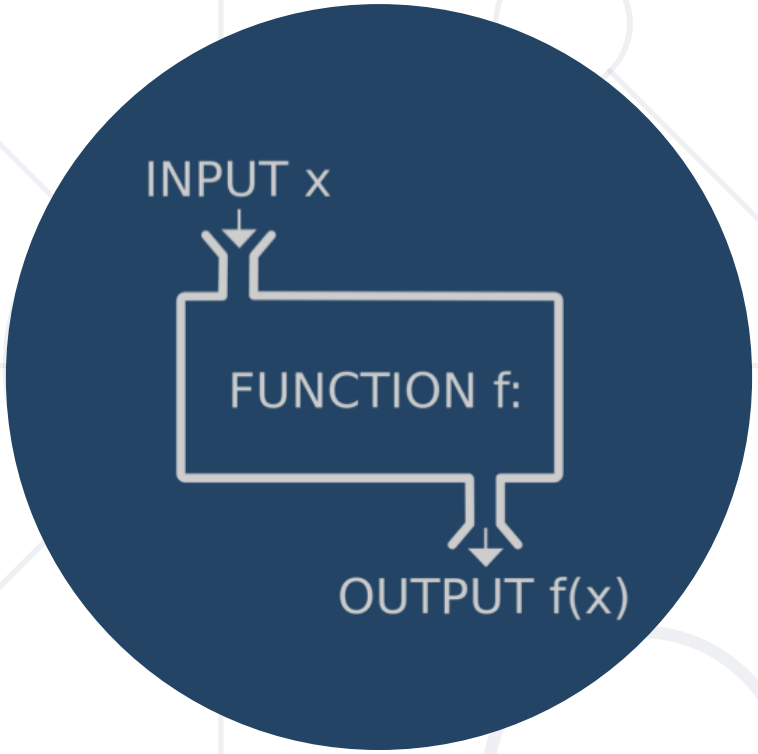
SoftUni Foundation

Have a Question?

SoftUni Foundation

sli.do

#JS-CORE



JavaScript Functions

Syntax, Invocation, Return, Functions as values

Functions in JS

- Why Functions?
  - You can **reuse** code, define **once**, use **many times**.
- **Function** = block of code
  - Can take **parameters** and return **result**

Function name: use **camelCase**

Function **parameters**: use **camelCase**

```
function printStars(count) {
  console.log("*".repeat(count));
}
```

The { stays at the same line

```
printStars(10);
```

Invoke the function

Functions in JS (2)

SoftUni Foundation

```
function sum (a, b) {
  console.log(a + b);
}
sum (5, 6);              // 11
```

Invoke the function with different parameters value

```
function sum (a, b = 3) {
  console.log(a + b);
}
sum (11);                // 14
```

Default function parameters

```
function sum (a, b) {
  return a + b;
}
let c = sum (5.8, 3);
console.log (c);         // 8.8
```

Return ends function execution

Function Declaration, Expression, Arrow

SoftUni Foundation

```
function walk() {
    console.log('walking');
}
walk();          // walking
```

Function **Declaration**

```
let solve = function walk() {
    console.log('walking');
}
solve();         // walking
```

Function **Expression**

```
let solve = () => {
    console.log('walking');
}
solve();         // walking
```

Arrow function

Problem: Multiplication Table

- Write a JS function to create multiplication table based on 2 numbers, that you will receive. If the first number is **greater**, print "*Try with other numbers.*"

Problem: Multiplication Table (2)

```
function multiplicationTable () {
     let numberToBeMultiplied =
parseInt(document.getElementById("num1").value);
     let multiplier = parseInt(document.getElementById("num2").value);
     let divResult = document.getElementById('result');

     function findWrongInput(numberToBeMultiplied, multiplier) {
          if (numberToBeMultiplied > multiplier) {
               document.getElementById("result").innerHTML = "Try with
other numbers.";
     }
}

 // Continues on the next slide
```

Problem: Multiplication Table (2)

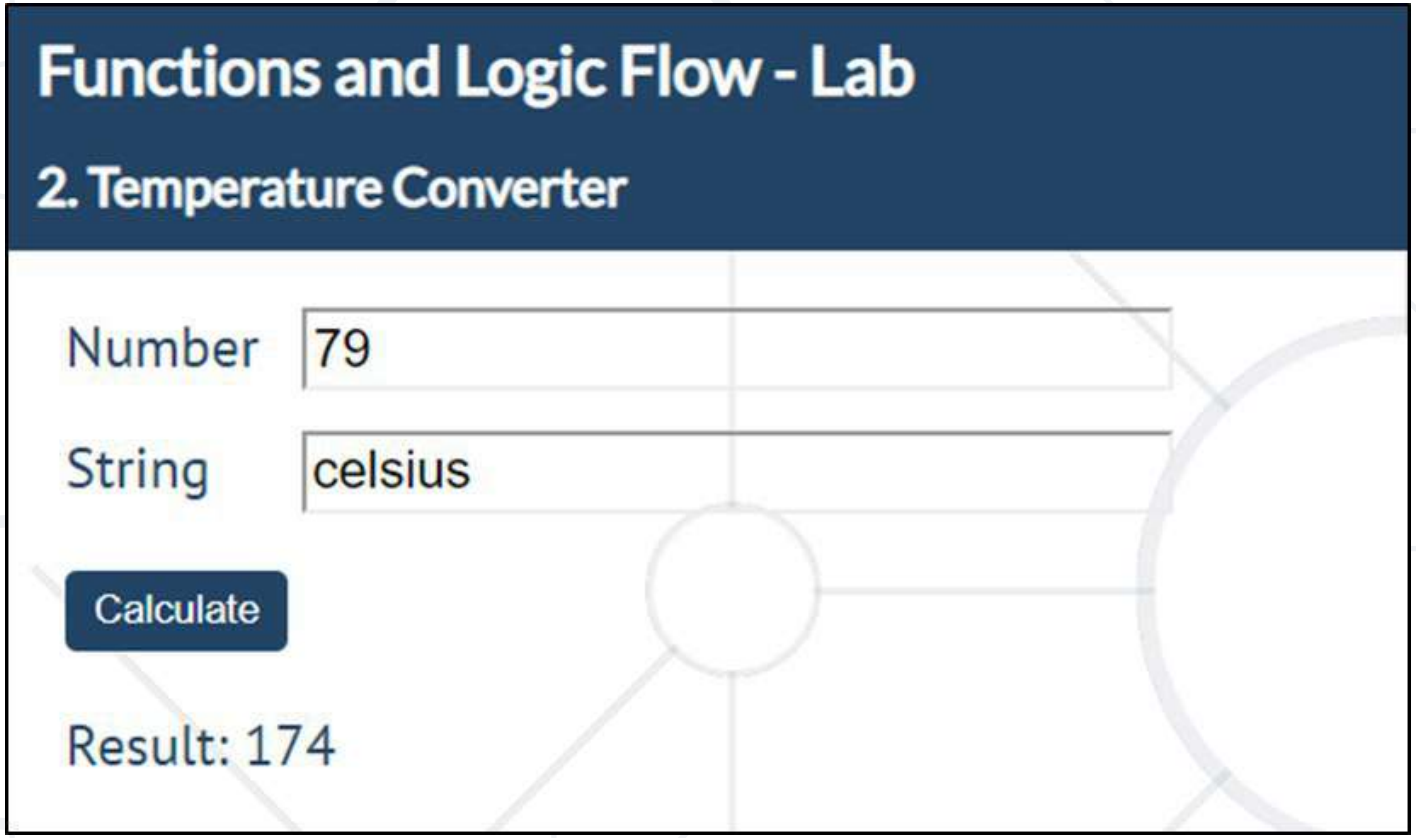
```
function printTable(numberToBeMultiplied, multiplier) {
     for (let i = numberToBeMultiplied; i <= multiplier; i++) {
         let result = multiplier * i;
         let p = document.createElement('p');
         p.textContent = `${i} * ${multiplier} = ${result}`;
         divResult.appendChild(p);
     }
}
divResult.textContent = '';

findWrongInput(numberToBeMultiplied, multiplier);
printTable(numberToBeMultiplied, multiplier);
}
```

`multiplicationTable(8, 3)` → Try with other numbers.

Problem: Temperature Converter

- Write a JS function to convert Fahrenheit to Celsius.



Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1449#1

Problem: Temperature Converter (2)

```
function temperatureConverter() {
     let degrees = parseInt(document.getElementById("num1").value);
     let type = document.getElementById("type").value;
     let result = '';
     let convertedTemperature = 0;
     let correctType = false;

    function convert(degrees, type) {
        if (type.toLowerCase() === "fahrenheit") {
            convertedTemperature = (((degrees - 32) * 5) / 9);
            correctType = true;
        }  else if (type.toLowerCase() === "celsius") {
           convertedTemperature = ((degrees * 9) / 5) + 32;
           correctType = true;
         }
      }                                          // Continues on the next slide
```

Problem: Temperature Converter (3)

```
function print(degrees, type) {
    if (correctType) {
        result = Math.round(convertedTemperature);
    } else {
        result = "Error!";
    }
}
    convert(degrees, type);
    print(degrees, type);
    document.getElementById("result").innerHTML = result;
}
```

```
temperatureConverter(85, "fahrenheit");
```

→

```
29
```

Function Invocation

- The code inside a function is executed when the function is **invoked** with **()** operator

```
function write(name) {
      console.log(`I am ${name}`);
}
write('George');        // I am George
```

Invoke the function **after** declaration

```
write('George');        // I am George
function write(name) {
      console.log(`I am ${name}`);
}
```

Invoke the function **before** declaration

Function Invocation (2)

SoftUni Foundation

```
(function (count) {
  for (let i = 1; i <= count; i++)
    console.log('+'.repeat(i));
})(4);
```

Anonymous **self-invoking** function

```
+
++
+++
++++
```

```
let f = (function () {
  let x = 0;
  return function() { console.log(++x); }
})(); f(); f(); f(); f();
```

This is called "**closure**" (a state is closed inside)

```
1
2
3
4
```

Function Return

- Return statement **ends** function execution

```
function getRectArea(width, height) {
  if (width > 0 && height > 0) {
    return width * height;
  }
  return 0;
}
console.log(getRectArea(3, 4));        // 12
console.log(getRectArea(-3, 4));       // 0
```

```
let result = function (a, b) {
     return a % b;
};
console.log(result(10, 3));            // 1
```

Variables Holding Functions

SoftUni Foundation

- In JS variables can hold functions as their values

```
let f = function(x) { return x * x; }
console.log(f(3)); // 9
console.log(f(5)); // 25

f = function(x) { return 2 * x; }
console.log(f(3)); // 6
console.log(f(5)); // 10

f = undefined;
console.log(f(3)); // TypeError: f is not a function(…)
```

Functions as Parameters

```
function repeatIt(count, func) {
  for (let i = 1; i <= count; i++)
    func(i);
}

let starsFunc = function(i) {
  console.log("**".repeat(i))
};

repeatIt(3, starsFunc);

repeatIt(3, function(x) {
    console.log(2 * x);
});
```

```
**
****
******
```

```
2
4
6
```

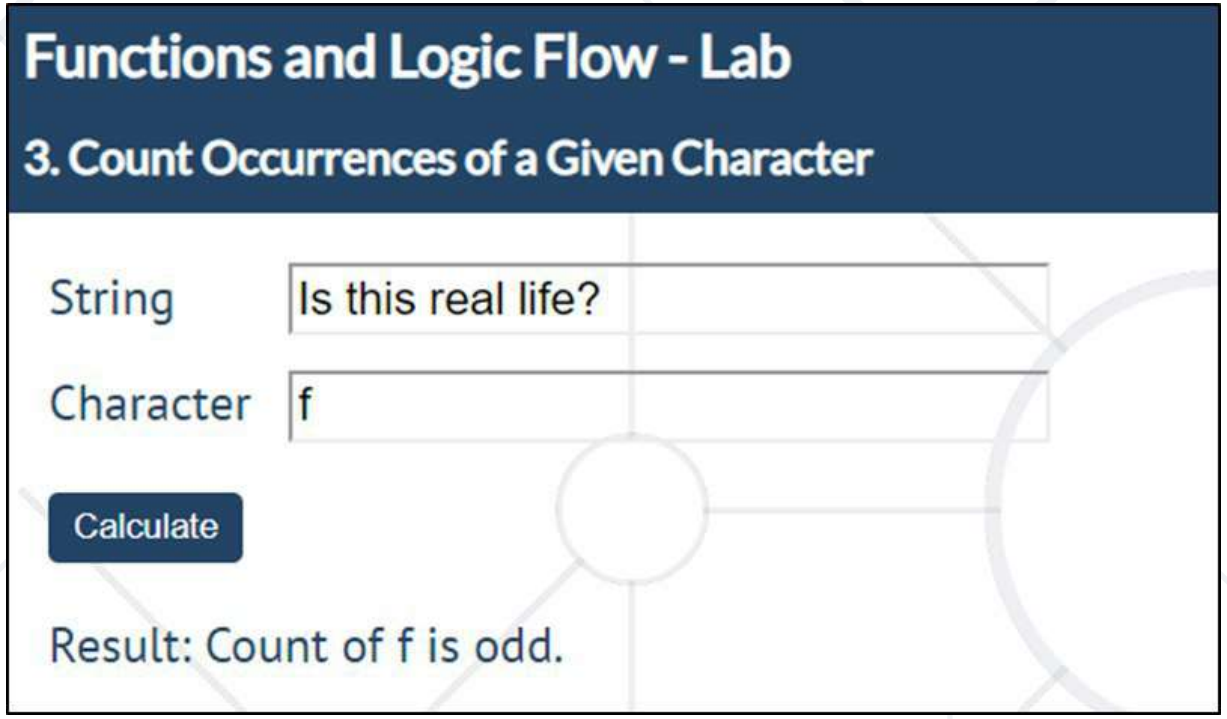
Problem: Count Occurrences of a Given Character

SoftUni Foundation

- Write a JS function that finds occurrences of a character in a string
  - Takes 2 parameters – string and character
  - Finds the count of occurrences of the given char in the given string
  - Checks if the count is even / odd

```
countChars('Is this real life?', 'f');
```

```
Count of f is odd.
```



Problem: Count Occurrences of a Given Character

```
function countChars() {
            let string = document.getElementById("string").value;
            let char = document.getElementById("character").value;
            let count = 0;
            let result = '';

    function findCharacterCount(string, char) {
        for (let i = 0; i < string.length; i++) {
            if(string[i] === char) {
                count++;
            }
        }
    }
```

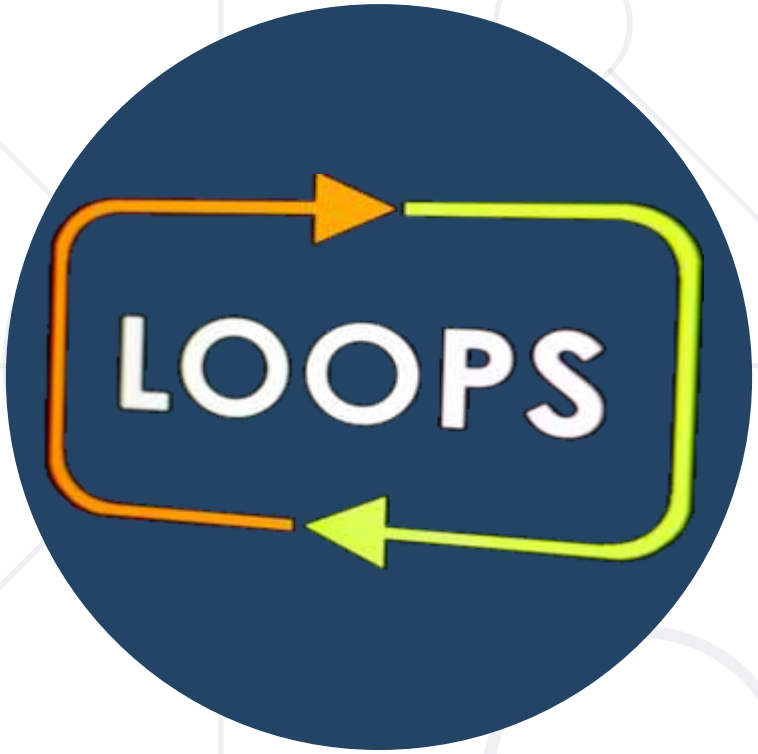
Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1449#2

Problem: Count Occurrences of a Given Character

```
    function evenOrOddCount(string, char) {
        if (count % 2 === 0) {
            result = `Count of ${char} is even.`;
        } else {
            result = `Count of ${char} is odd.`;
        }
    }

    findCharacterCount(string, char);
    evenOrOddCount(string, char);
    document.getElementById("result").innerHTML = result;
}
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1449#2



Loops in JS

For…in, For…of, While

Loops: for…in

- The **for** / **while** loops work as in C++, C# and Java
- For…in loop is used to iterate over the **enumerable** properties of objects

```
function solve(arr) {
    for (const arrElement in arr) {
        console.log(arrElement);
    }
}
solve(['George', 5, null, 54]);
```

for…in loop iterates over the **indexes**

```
0
1
2
3
```

Loops: for...in

```
function solve(str) {
    for (const character in str) {
        console.log(str[character]);
    }
}
solve('Hello');
```

```
H
e
l
l
o
```

```
function solve(obj) {
    for (const objElement in obj) {
        console.log(objElement);
    }
}
solve({name: 'Peter', age: 32, town:
   'Sofia'});
```

```
name
age
town
```

Loops: for…of

- For…of loop is used to iterate over the **iterable** objects

```
function solve(arr) {
    for (const arrElement of arr) {
        console.log(arrElement);
    }
}
solve(['George', 5, null, 54]);
```

```
George
5
null
54
```

Loops: for…in

```
function solve(str) {
    for (const character of str) {
        console.log(character);
    }
}
solve('Hello');
```

```
H
e
l
l
o
```

```
function solve(obj) {
    for (const objElement of obj) {
        console.log(objElement);
    }
}
solve({name: 'Peter', age: 32, town:
   'Sofia'});
```

TypeError: obj is not iterable

Loop: while

- The while loops through a block of code as long as a specified condition is **true**.

```
function count(num) {
    while (num < 200) {
        console.log(num *= 2);
    }
}
count(5);  // 10 20 40 80 160 320
```

```
function count(arr) {
    while (arr.length>0) {
        console.log(arr.length);
        arr.length-=1;
    }
}
count(['Peter', 45, 0, 58]); // 4 3 2 1
```

Problem: Unique Characters

- Write a JS function that takes one string parameter and extract only the unique characters from the string except for whitespaces.

```
function extractUniqueChars() {
            let uniqueChars = "";
            let string = document.getElementById("string").value;

            function findUniqueChars(string) {
                for (let i = 0; i < string.length; i++) {
                    isCharWhiteSpace(i);
                    isCurrentCharUnique(i);
                }
            }
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1449#3

Problem: Unique Characters (2)

```
    function isCharWhiteSpace(i) {
        if (string[i] === " ") {
            uniqueChars += string[i];
        }
    }

    function isCurrentCharUnique(i) {
        if (uniqueChars.indexOf(string[i]) === -1) {
            uniqueChars += string[i];
        }
    }

findUniqueChars(string);
document.getElementById("result").innerHTML = uniqueChars;
}
```

Scope

Local and global scope

Scope

- Scope is the **visibility** of functions and variables in the different **parts** of your code during runtime.
  - Global Scope – **Global** variables can be accessed from anywhere in a JavaScript function

```
var name = 'Maria’;          // code here can use name
function myFunction() {
    console.log(name);       // code here can also use name
}

myFunction();                // Maria
console.log(name);           // Maria
```

Scope (2)

SoftUni Foundation

- Function Scope – **Local** variables can only be accessed from inside the function where they are declared

```
function myFunction() {
    var name = 'Maria';
    // only here code CAN use name
}
console.log(name);
```

ReferenceError: name is not defined

- Block Scope - Variables declared inside a block {} can not be accessed from outside the block.

```
{
    let x = 2;
} // x can NOT be used here
```

Automatically global

- We can assign a value to a variable that has **not** been **declared** and it automatically become a **Global** variable

Invoke the function **before** declaration

```
myFunction();
console.log(name);          // Maria

function myFunction() {
    name = 'Maria';
}
```

Automatically global (2)

- Global variables are **not created** automatically in 'Strict Mode'

```
'use strict'
myFunction();
console.log(name);              // ReferenceError: name is
                                   not defined

function myFunction() {
    name = 'Maria';
}
```

Problem: Special Words

Write a JS function that takes 5 parameters and Iterates from the first number to the second one.

- For numbers which are multiples of both 3 and 5, print all 3 strings
- For multiples only of 3, print the second string
- For multiples only of 5, print the third string

Problem: Special Words (2)

```
function specialWords() {
     let startNum =
parseInt(document.getElementById("firstNumber").value);
     let endNum =
parseInt(document.getElementById("secondNumber").value);
     let firstWord = document.getElementById("firstString").value;
     let secondWord = document.getElementById("secondString").value;
     let thirdWord = document.getElementById("thirdString").value;
     let divResult = document.getElementById("result");

     for (let i = startNum; i <= endNum; i++) {
          checkCurrentNumber(i);
     }

 // Continues on the next slide
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1449#4

Problem: Special Words (3)

```
    function checkCurrentNumber(i) {
        let p = document.createElement('p');
        if (i % 3 === 0 && i % 5 === 0) {
            p.textContent = `${i} ${firstWord}-${secondWord}-
${thirdWord}`;
        } else if (i % 3 === 0) {
            p.textContent = `${i} ${secondWord}`;
        } else if (i % 5 === 0) {
            p.textContent = `${i} ${thirdWord}`;
        } else {
            p.textContent = i;
        }

        divResult.appendChild(p);
    }
}
```

Live Exercises

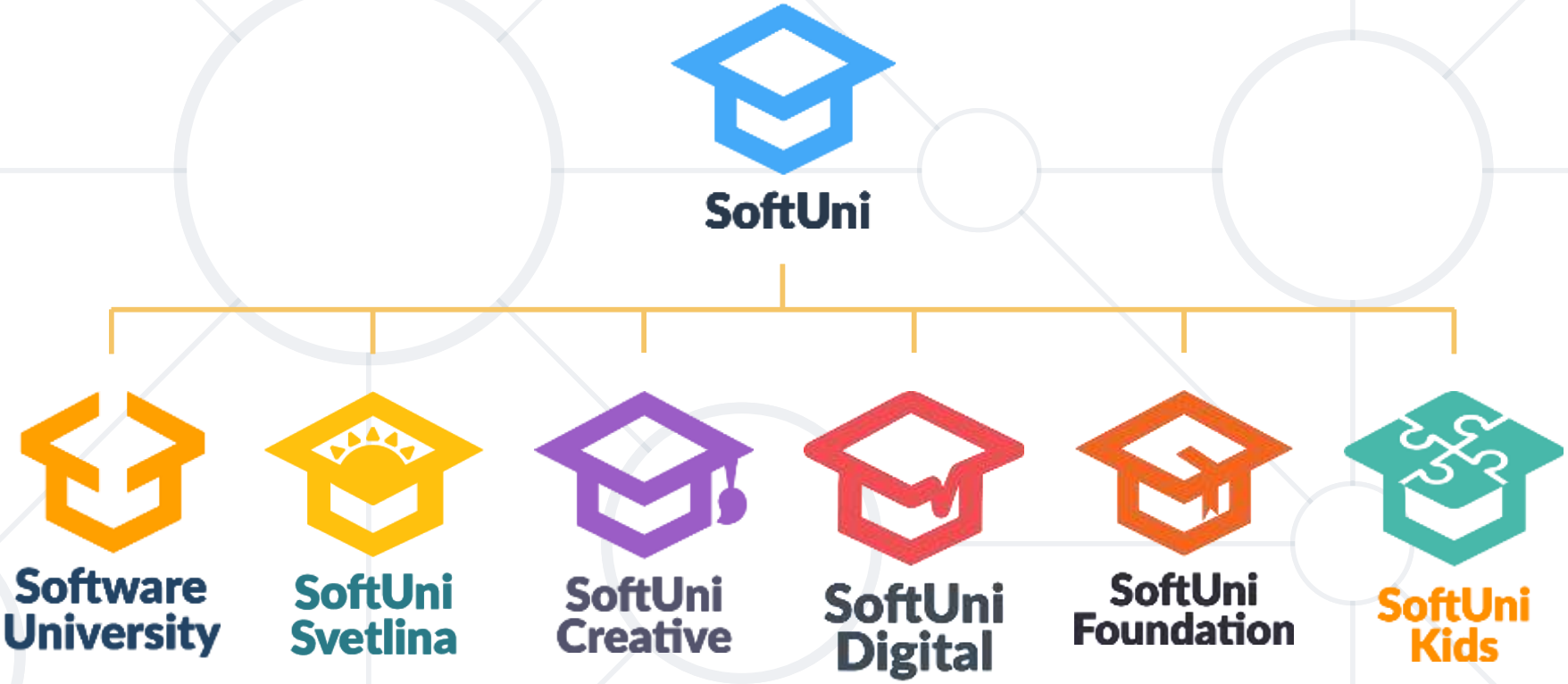
Summary

- Function = named piece of code
  - Syntax, invocation, return

```
function calcSum(a, b) {
  let sum = a + b;
  return sum;
}
```

- Loops – for…in, for…of, while
- Local and global scope

Questions?



https://softuni.bg/courses/javascript-fundamentals

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SBTech
we know sports

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

æternity

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

INFRAGISTICS

LIEBHERR

Postbank
Решения за твоето утре

SoftUni Organizational Partners

SoftUni Foundation

ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

Lukanet.com

WORLD OF MYTHS

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education and Employment Opportunities
  - softuni.bg
- Software University Foundation
  - http://softuni.foundation/
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

License

- This course (slides, examples, demos, videos, homework, etc.) is licensed under the "Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International" license