JavaScript Syntax and Operators

JS Syntax and JS Operators

SoftUni Team
Technical Trainers

Software University
SoftUni Foundation

Software University
http://softuni.bg

Table of Content

SoftUni Foundation

Have a Question?

SoftUni Foundation

sli.do

#js-core

JavaScript Syntax

Values, Literals, Variables, Operators, Expressions

JavaScript Syntax

- JavaScript syntax refers to a set of **rules** that determine:
  - How the language will be **written** (by the programmer)
  - How the language will be **interpreted** (by the browser).

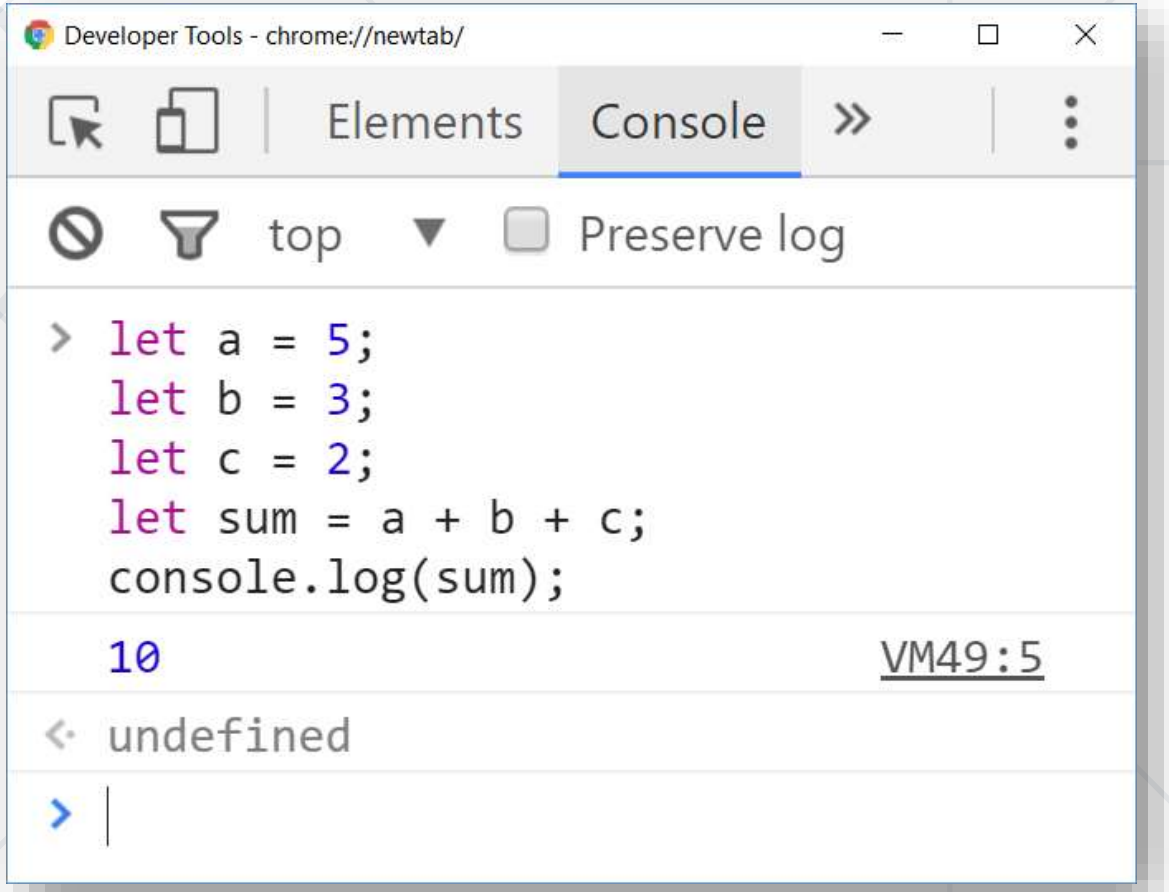
```
let x, y;                    // Declare variables
x=5; y=6;                    // Assign values
let z = x + y;               // Calculate values
console.log(z);              // Print values
```

JavaScript Code: Example

```
let a = 5;
let b = 3;
let c = 2;

let sum = a + b + c;

console.log(sum); // 10
```



JavaScript Fixed Values

- Fixed values – literals
  - **Array Literals**: list of zero or more **array element**, enclosed in square brackets ([ ])

Square brackets

Array element

```
let cars = ["Ford", "BMW", "Peugeot"]
let arrayLength = cars.length;        // 3
let secondCar = cars[1];              // "BMW"
```

JavaScript Fixed Values (2)

SoftUni Foundation

- **Boolean Literals**: two literal values – **true** and **false**

```
console.log("0" == true);                          // false
console.log("0" == false);                         // true
if ("0") { console.log(true) };                    // true
console.log([] == true);                           // false
console.log([] == false);                          // true
if ([]) { console.log(true) };                     // true
console.log(null == false || null == true);        // false
if (!null) { console.log(true) };                  // true
```

JavaScript Fixed Values (2)

SoftUni Foundation

- Integers:

```
0, 16, -528                    // Decimal, base10
015, 0001, -0o77               // Octal, base8
0x1123, 0x00111, -0xF1A7       // Hexadecimal, "hex" or base16
0b11, 0b0011, -0b11            // Binary, base2
```

- Floating-points Literal can have the following parts:
  - Preceded by "+" or "-"
  - A decimal point (".")
  - A fraction (another decimal number)
  - An exponent

```
3.141854
-0.54875
-3.1E+12
.1e-23
```

JavaScript Fixed Values (3)

SoftUni Foundation

- Object Literals:
  - List of zero or more **pairs** of property names
  - Associated values of an object, enclosed in curly braces **{ }**

```
let car = {type: "Infinity", model: "QX80", color: "blue"};
let carType = car.type;
let carType = car["type"];  // Access property
car.year = 2018;
car["year"] = 2018; // Add new property
car.color = "black";
car["color"] = "black"; // Correct existing property
```

JavaScript Fixed Values (4)

- RegExp Literals: **pattern** enclosed between slashes (/ /)

```
let pattern = /[A-Za-z]+/;
```

- String Literals: **immutable** sequences of **Unicode** characters

```
"hello", "apple", '123', 'I like my car'
let str = "Infinity QX80";
console.log(str.length); // 13
console.log(str[0]);  // I
str[0] = 's'; // Beware: no error, but str stays unchanged!
console.log(str); // "Infinity QX80"
console.log(str[20]); // undefined
```

JavaScript special characters: **\b**, **\n**, **\t**, **\v**, **\'**, **\"**, ****

Problem: String Length

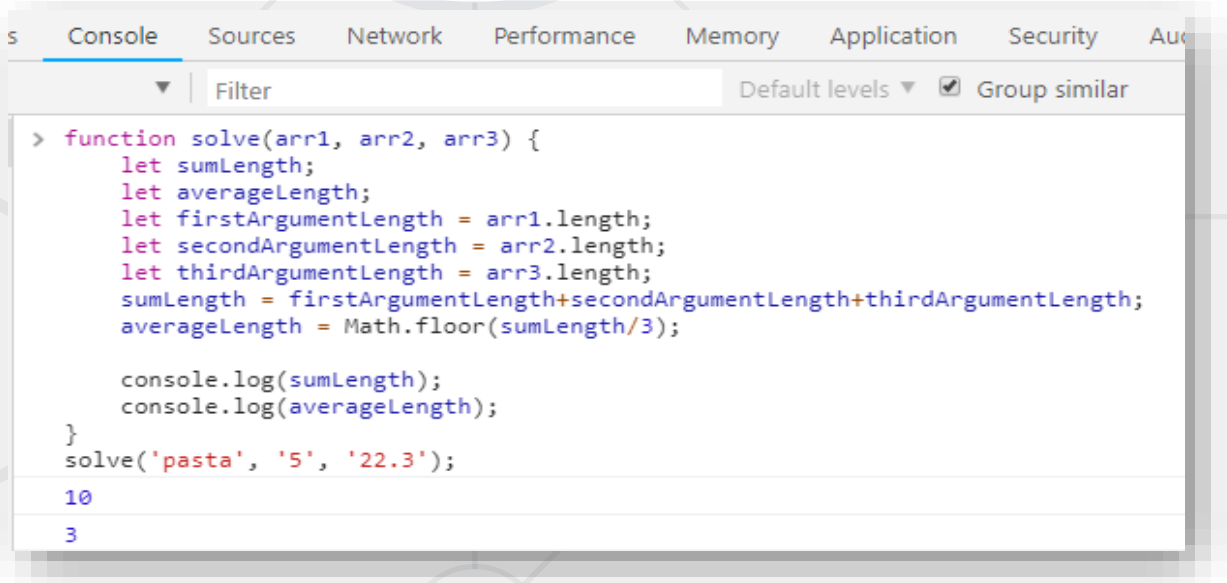
- You are given three strings argument
  - Print the sum of strings length and average length round down

```
function solve(arr1, arr2, arr3) {
   let sumLength;
   let averageLength;
   let firstArgLength = arr1.length;
   let secondArgLength = arr2.length;
   let thirdArgLength = arr3.length;
   sumLength = firstArgLength + secondArgLength + thirdArgLength;
   averageLength = Math.floor(sumLength/3);
   ...
}
```

Problem: String length (2)

```
function solve(arr1, arr2, arr3) {
  ...
  console.log(sumLength);
  console.log(averageLength);
}
```

```
solve('pasta', '5', '22.3')
```



Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1421#0

JavaScript Variable Values

- **Variable values** – variables are used to **store** data values
- JS uses **let**, **const** and **var** keywords to declare variables
  - **let** – for **reassign** a variable:

```
let name = "George";
name = "Maria";
```

  - **const** - once assigned, constants **cannot** be modified

```
const name = "George";
name = "Maria";
console.log(name) // TypeError: Assignment to constant variable.
```

  - **var** - a keyword which defines a variable globally
    - Regardless of block scope
    - **Do not use var in your code**

JavaScript Operators

- **Arithmetic operators**:
  - Take numerical values (either literals or variables) as their operands
  - Returns a single numerical value
    - Addition (+)
    - Subtraction (-)
    - Multiplication (*)
    - Division (/)
    - Remainder (%)
    - Exponentiation (**)

```
let a = 15;
let b = 5;
let c;
c = a + b; // 20
c = a - b; // 10
c = a * b; // 75
c = a / b; // 3
c = a % b; // 0
c = a ** b; // 759375
```

Arithmetic Operators

```
console.log(3 + 4 - 2);                    // 5 (add / subtract numbers)
console.log(5 / 0);                        // Infinity (divide by zero)
console.log(Infinity / Infinity);          // NaN (wrong division)
console.log(Math.round(7 / 3));            // 2 (integral division)
console.log(Math.ceil(7 / 3));             // 3 (integral division)
console.log(Math.floor(7 / 3));            // 2 (integral division)
console.log(7 % 3);                        // 1 (remainder of division)
console.log(5.3 % 3);                      // 2.3 (remainder of division)
let a = 5; console.log(++a);               // 6 (prefixed ++)
console.log(a++);                          // 6 (postfix ++)
```

JavaScript Operators

- **Assignment operators**: they **assign** a value to its left operand based in the value of its right operand.

```
let a = 15;
let b = 5;
a = b;                    // Assign the value of b to a
console.log(a);           // 5
console.log(b);           // 5
```

```
let a = 15;
let b = 5;
b = a;                    // Assign the value of a to b
console.log(a);           // 15
console.log(b);           // 15
```

Compound assignment operators

Name	Shorthand operator	Meaning
Assignment	x = y	x = y
Addition assignment	x += y	x = x + y
Subtraction assignment	x -= y	x = x - y
Multiplication assignment	x *= y	x = x * y
Division assignment	x /= y	x = x / y
Remainder assignment	x %= y	x = x % y
Exponentiation assignment	x **= y	x = x ** y

JavaScript Syntactic Categories

- In JavaScript there are two major syntactic categories:
- Statements are "**commands**" to be executed
  - if

```
let number = 5;

if (number % 2 === 0) {
  console.log("Even number");
}
```

  - else if

```
let number = 5;

if (number % 2 === 0) {
  console.log("Even number");
} else {
  console.log("Odd number");
}
```

Statements

- for

```
for (let i = 0; i <= 5; i++){
  console.log(i);          // 0 1 2 3 4 5
};
```

- switch

```
let day = 3;
switch (day) {
  case 1: console.log('Monday'); break;
  case 2: console.log('Tuesday'); break;
  case 3: console.log('Wednesday'); break;
   …
  case 7: console.log('Sunday'); break;
  default: console.log('Error!'); break;
};
```

Problem: Math Operators

- Make the required arithmetic operation between two numbers and an arithmetic operator you take from the input

```
function solve(num1, num2, operator) {
  let result;
  switch (operator) {
    case '+': result = num1 + num2; break;
    case '-': result = num1 - num2; break;
    case '*': result = num1 * num2; break;
    case '/': result = num1 / num2; break;
    case '%': result = num1 % num2; break;
    case '**': result = num1 ** num2; break;
   }
  console.log(result);
}
```

```
solve(5, 6, '+');
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1421#1

Problem: Sum of Numbers N...M

- Calculate the sum of all numbers from **n** to **m**

```
function solve(n, m) {
  let result = 0;
  let num1 = Number(n);
  let num2 = Number(m);
  for (let i = num1; i <= num2; i++) {
    result+=i;
   }
  return result;
}
```

```
solve(1, 5);
```

```
> function solve(n, m) {
      let num1 = Number(n);
      let num2 = Number(m);
      let result = 0;
      for (let i = num1; i <= num2; i++) {
          result+=i;
      }
      return result;
  }
  console.log(solve('1', '5'));
  15
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1421#2

Statements (1)

- **while**

```
let count = 1;
while (count < 1024) {
  console.log(count *= 2); // 2 4 8 16 32 64 128 256 512 1024
};
```

- **do-while**

```
let s = "ho";
do {
  console.log(s);    // ho hoho hohohoho hohohohohohohoho
  s = s + s;
} while (s.length < 20);
```

Statements (2)

- for … in loop

```
let nums = [5, 10, 15, 20, 'maria', true];
for (let index in nums) {
  console.log(index);
}
// 0 1 2 3 4 5 → loops through the indices (keys), not values
```

- for … of loop

```
let nums = [5, 10, 15, 20, 'maria', true];
for (let value of nums) {
  console.log(value);
}
// 5 10 15 20 maria true → loops through the values
```

Statements (3)

- **debugger**

```
let x = 15 * 5;          // With the debugger turned on,
debugger;                this code should stop executing
console.log(x);          before it executes the third line
```

- **variable declaration**

```
let x = 5;        // x stores the value 5
let y = 14.5;     // y stored the value 14.5
```

Expressions

- Expression is any valid unit of code that resolves to a value:
  - with **side** effects:

```
let assignedVariable = 2;          // This is a statement
assignedVariable = 5;              // expression
console.log(assignedVariable)      // 5
```

  - with **resolve** effects:

```
let assignedVariable = 2;          // This is a statement
assignedVariable + 4;              // expression
assignedVariable * 10;             // expression
assignedVariable - 10;             // expression
console.log(assignedVariable);     // 2
```

JavaScript Operators

Comparison, Logical Operators

Comparison Operators

- **Comparison operators** - compare values
  - The `==` means "equal after type conversion"
  - The `===` means "equal and of the same type"
  - The `!=` means "not equal after type conversion"
  - The `!==` means "not equal and of the same type"
  - The `>` means "greater than"
  - The `<` means "less than"
  - The `>=` means "greater than or equal to"
  - The `<=` means "less than or equal to"

Comparison Operators (2)

- The ? is ternary operator

```
let a = 5, b = 4;
console.log(a == b);          // false
console.log(0 === "");        // false
console.log(a != b);          // true
console.log(3 !== "3");       // true
console.log(a < "5.5");       // true
console.log(a >= b);          // true
console.log(0 == []);         // true
console.log(a ? b : 10);      // 4
```

Logical Operators

SoftUni Foundation

- **Logical operators** are used to determine the **logic** between **variables** or **values**
  - && (**logical and**) - returns the leftmost "**false**" value:

```
let val = true && 'yes' && 5 && null && false;
console.log(val); // null
let val = true && 'no' && 5 && 25 && 'yes';
console.log(val); // 'yes'
```

If all values are **true**, return the **last** value

  - || (**logical or**) - operators returns the leftmost "**true**" value:

```
let val = false || 0 || '' || 5 || 'hi' || true;
console.log(val); // 5
let val = false || '' || null || NaN || undefined;
console.log(val); // undefined
```

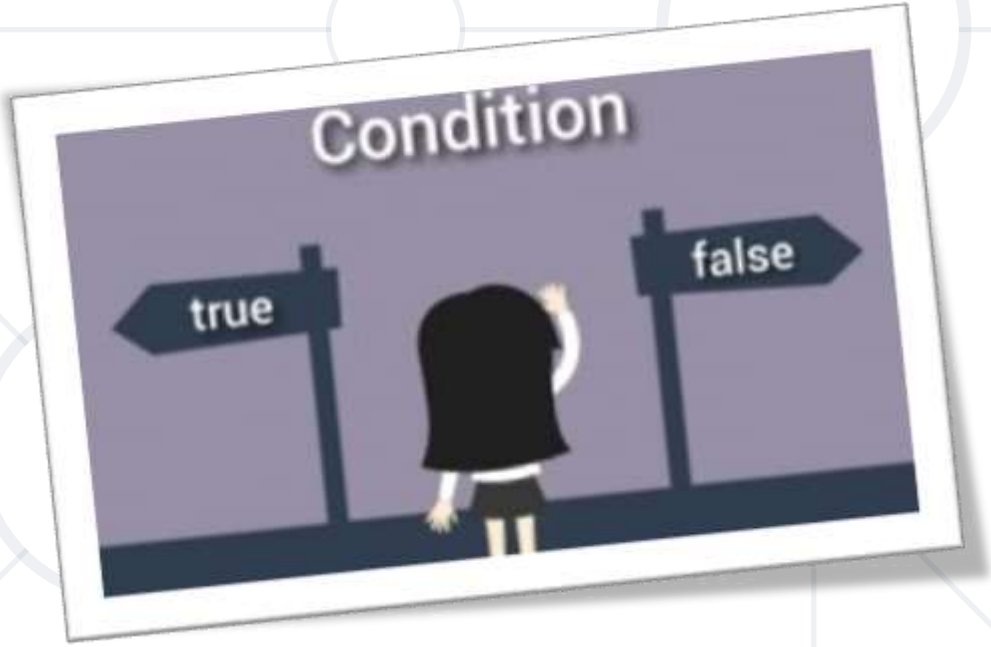
If all values are **false**, return the **last** value

Logical Operators (2)

- ! (**logical not**) – convert the operand to Boolean type: **true/false**

```
let val = !true
console.log(val); // false
let val = !false;
console.log(val); // true
```

A	B	A AND B	A OR B	NOT A
False	False	False	False	True
False	True	False	True	True
True	False	False	True	False
True	True	True	True	False



Problem: Larger Number

- Write a JS function that takes three number arguments as input and find the largest of them.

```
function solve(num1, num2, num3) {
  let result;
  if(num1>num2 && num1>num3) {
    result = num1;
  } else if (num2>num1 && num2>num3) {
    result = num2;
   } else if(num3>num1 && num3>num2) {
    result = num3;
   }
   console.log(`The largest number is ${result}.`)
}
```

```
solve(5, -3, 16)
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1421#3

Typeof Operator

- The **typeof** operator returns a string indicating the type of operand

```
let val = 5; console.log(typeof(val));              // number
let str = 'hello'; console.log(typeof(str));        // string
let obj = {name: 'Maria', age:18};
console.log(typeof(obj));                           // object
let arr = [1, 2, 3]; console.log(typeof(arr));      // object
let bool = true; console.log(typeof(bool));         // Boolean
let func = function(){};
console.log(typeof(func));                          // function
let date = new Date();
console.log(typeof(date));                          // object
console.log(typeof(notDeclaredVariable));           //undefined
```

Problem: Circle Area

- Calculate the circle area if the input is a number.
  You need to check the type of the input

```
function solve(input) {
  let result = 0;
  let inputType = typeof(input);
  if (inputType === 'number') {
    result = Math.pow(input, 2) * Math.PI;
    console.log(result.toFixed(2));
  } else {
    console.log(`We can not calculate the circle area,
    because we receive a ${inputType}.`)
   }
}
```

```
solve(5);
```

Check your solution here: https://judge.softuni.bg/Contests/Practice/Index/1421#4

Instanceof Operator

- The **instanceof** operator returns **true** if the specified object in an instance of the specified object:

```
let cars = ["Saab", "Volvo", "BMW"];
console.log(cars instanceof Array);          // Returns true
console.log(cars instanceof Object);         // Returns true
console.log(cars instanceof String);         // Returns false
console.log(cars instanceof Number);         // Returns false
```

Live Exercises in Class (Lab)

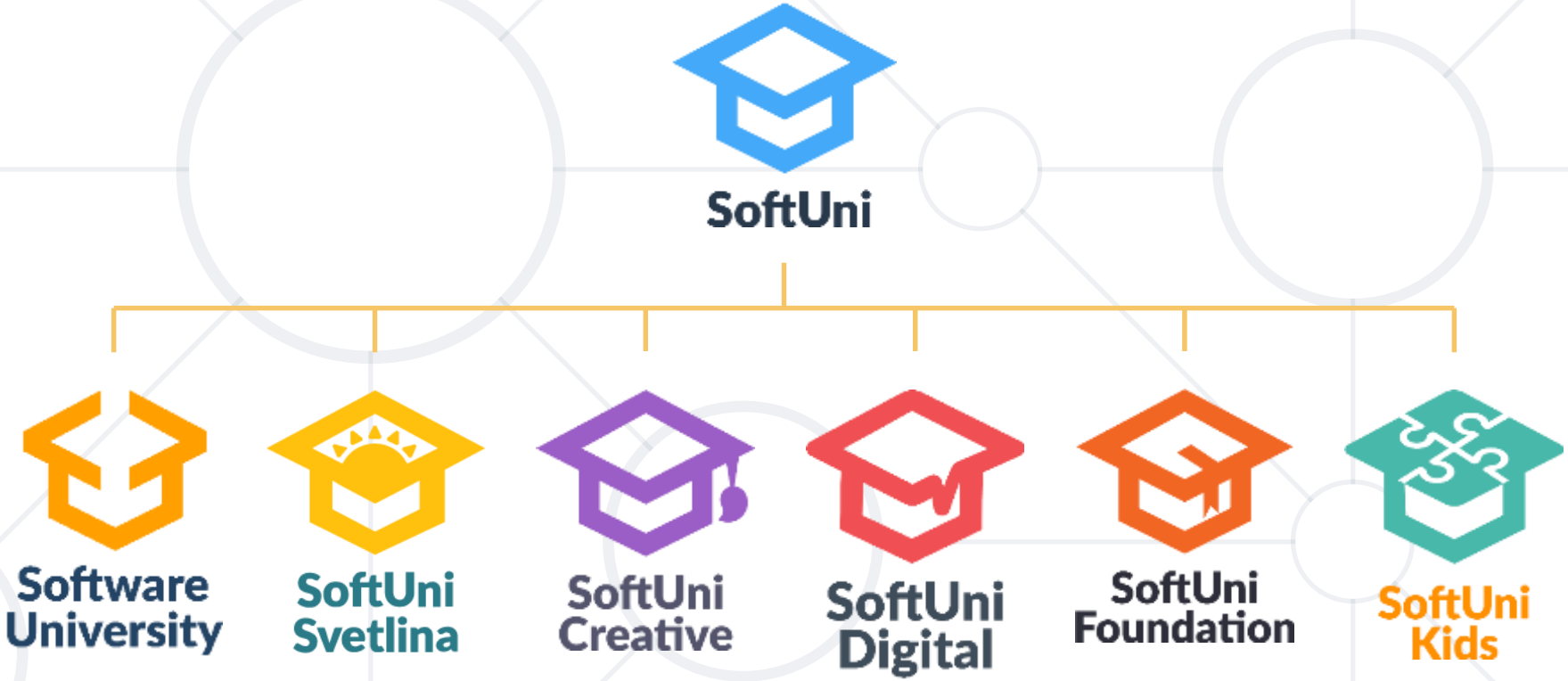
Practice: JavaScript Syntax and Operators

Summary

SoftUni Foundation

- Basic JavaScript syntax
- Conditional statements in JS are like in all modern programming languages:
  - Classical conditionals: if-else, switch-case
- Loops in JavaScript
  - Classical loops: while, do-while, for loops
  - Iterate over collection: for … in and for … of
- Comparison operators - ==, ===, >, <, >=, <=, !=, !==, ?
- Logical operators - &&, ||, !
- Typeof, instanceof

Questions?


SoftUni
Software University
SoftUni Svetlina
SoftUni Creative
SoftUni Digital
SoftUni Foundation
SoftUni Kids

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SBTech
we know sports

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

INFRAGISTICS

LIEBHERR

æternity

codexio

SoftUni Organizational Partners

SoftUni Foundation

ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

One SOFTW

WORLD OF MYTHS

Lukanet.com

codeXio

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