JavaScript Basics

Syntax, Data Type and Variables, Operators, Conditional Statements, Loops, Debugging

SoftUni Team

Technical Trainers

Software University

https://softuni.bg

You Have Questions?

sli.do

#QA-Auto-BackEnd

Table of Contents

JavaScript Overview

Definition, Execution, IDE Setup

What is JavaScript?

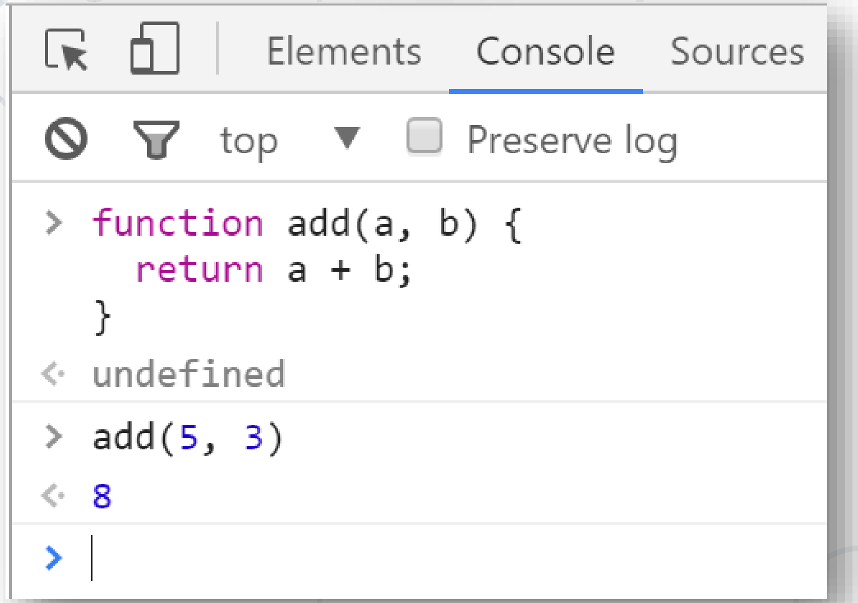
- JavaScript (**JS**) is a **high-level** programming language
  - One of the **core technologies** of the World Wide Web
  - Enables **interactive** web pages and applications
  - Can be **executed** on the **server** and on the **client**
- Features
  - C-like **syntax** (curly-brackets, identifiers, operator)
  - **Multi-paradigm** (imperative, functional, OOP)
  - Dynamic **typing**

Dynamic Programming Language

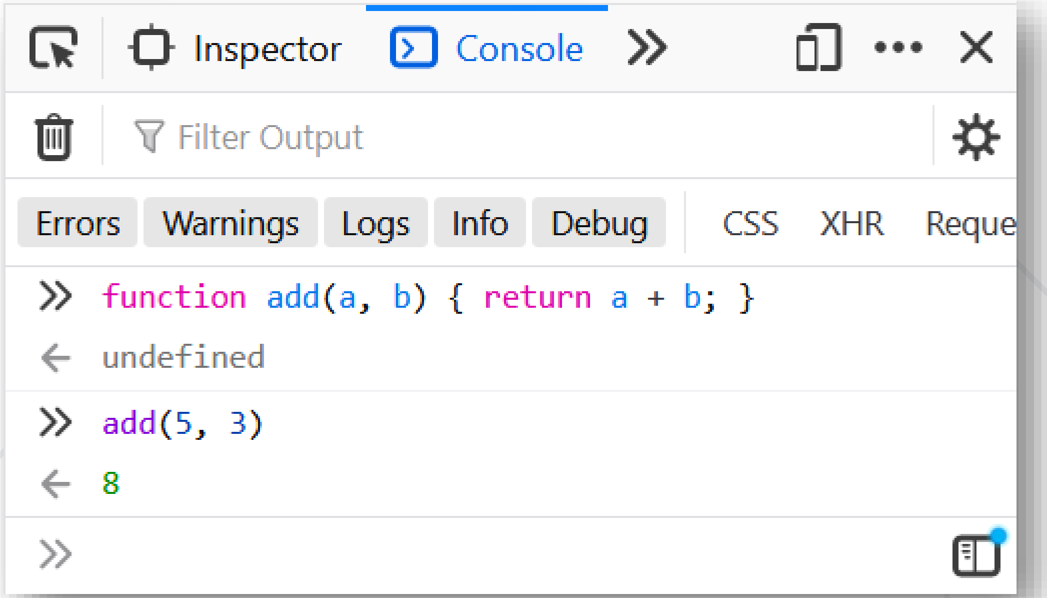
- JavaScript is a **dynamic programming language**
  - Operations otherwise done at **compile-time** can be done at **run-time**
- It is **possible** to change the **type** of a variable or add new properties or methods to an object **while** the program is **running**
- In **static programming languages**, such changes are normally **not possible**

Web Browser Dev Console

- Developer Console: **[F12]**



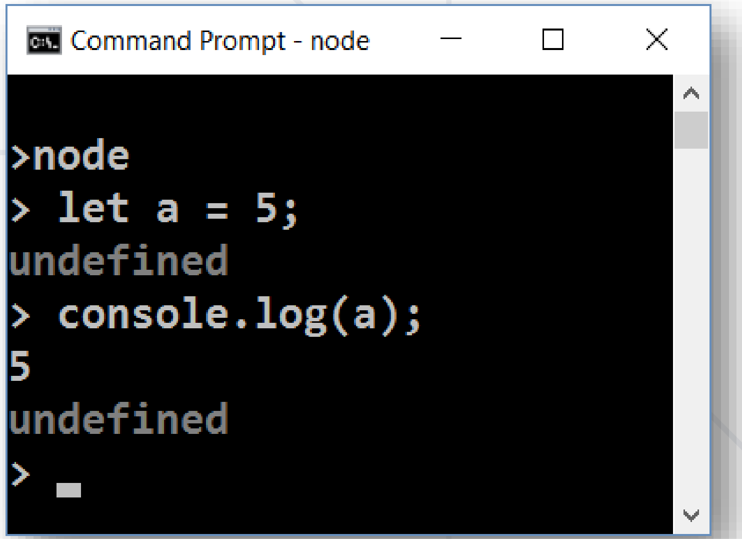
```
> function add(a, b) {
    return a + b;
  }
<· undefined
> add(5, 3)
<· 8
>
```



```
>> function add(a, b) { return a + b; }
<- undefined
>> add(5, 3)
<- 8
>>
```

Node.js

- What is **Node.js**?
  - **Server-side** JavaScript runtime
  - Chrome V8 JavaScript engine
  - NPM **package manager**
  - Install node packages



```
>node
> let a = 5;
undefined
> console.log(a);
5
undefined
> _
```

JavaScript Syntax

Functions, Operators, Input and Output

JavaScript Syntax

- Defining and initializing variables

```
let a = 5;
let b = 10;
```

Declare a variable with let

Variable name

Variable value

- Conditional statement

```
if (b > a) {
  console.log(b);
}
```

Body of the conditional statement

Functions and Input Parameters

- In order to solve different problems, we are going to use **functions** and the **input** will come as **parameters**
- A function is similar to a **procedure**, which executes when called

declaration

parameters

```
function solve (num1, num2) {
    // some logic
}

solve(2, 3);
```

calling the function

Printing to the Console

- We use the `console.log()` method to print to console

```
function solve (name, grade) {
  console.log('The name is: ' + name + ', grade: ' + grade);
}

solve('Peter', 3.555);
// The name is: Peter, grade: 3.555
```

- Text can be composed easier using interpolated strings
  - Works only with the ` brackets

```
console.log(`The name is: ${name}, grade: ${grade}`);
```

Printing to the Console

- To format a number, use the **toFixed()** method
  - Converts a number to **string**
    - Rounds the string to a specified number of decimals
      - Default value is 0 (no decimals)

Number of digits after the decimal sign

```
grade.toFixed(2);
// The name is: Peter, grade: 3.56
```

    - If the number of decimals is higher than in the number, zeros are added

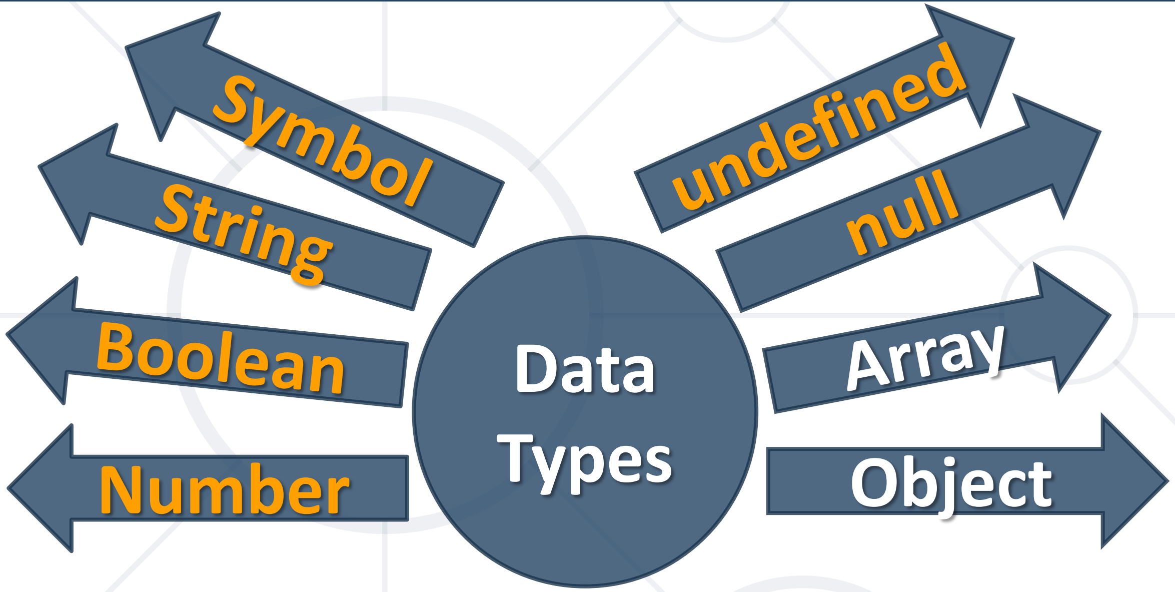
Data Types and Variables

Definitions and Examples

JavaScript Data Types

- Seven **primitive types**
  - Boolean
  - null
  - undefined
  - Number
  - String
  - Symbol
  - BigInt
- and **Objects** (including Functions and Arrays)

Data Types Examples



```
let number = 10;                              // Number
let person = {name: 'George', age: 25};       // Object
let array = [1, 2, 3];                        // Array
let isTrue = true;                            // Boolean
let name = 'George';                          // String
let empty = null;                             // null
let unknown = undefined;                      // undefined
```

Variable Scope

- **var**
  - Use **function scope**
  - Can be accessed anywhere in the function, including outside the initial block

```
{
  var x = 2;
}
console.log(x);
// 2
```

- **let** and **const**
  - Use **block scope**
  - Can **NOT** be accessed from outside the **{ }** block where initially declared

```
{
  let x = 2;
}
console.log(x);
// Error
```

let vs const

- `let`
  - Can be reassigned after initial assignment
  - Variable's value can change
  - `let` is used when reassignment is necessary

- `const`
  - Cannot be reassigned after initial assignment, remains constant
  - Variable's value remains fixed
  - `const` is used when variable will not be reassingned

Undefined

- A variable without a value has the value **undefined**
  - The **typeof** is also **undefined**

```
let car;
// Value is undefined, type is undefined
```

- A variable can be emptied, by setting the value to **undefined**
  - The type will also be **undefined**

```
let car = undefined;
// Value is undefined, type is undefined
```

Null

- Null is "nothing"
- It is supposed to be something that doesn't exist
- The typeof null is an object

```
let person = {
  firstName:"John",
  lastName:"Doe",
  age:50
};
person = null;
console.log(person);         // null
console.log(typeof(person)); // object
```

Operators

Overview of Different Types of Operators

Arithmetic Operators

- **Arithmetic operators**
  - Take numerical values (either literals or variables) as their operands
  - Return 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; // 15^5 = 759375c
```

Comparison Operators

- Used in logical statements to determine equality or difference between various variables or values

Operator	Notation in JS
Equal value	==
Equal value and type	===
Not equal value	!=
Not equal value/type	!==
Greater than	>
Greater than or Equal	>=
Less than	<
Less than or Equal	<=

Comparison Operators – Examples

```
console.log(1 == '1');   // true
console.log(1 === '1'); // false
console.log(3 != '3');   // false
console.log(3 !== '3'); // true
console.log(5 < 5.5);    // true
console.log(5 <= 4);     // false
console.log(2 > 1.5);    // true
console.log(2 >= 2);     // true

console.log((5 > 7) ? 4 : 10); // 10
```

Ternary operator

Typeof Operator

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

```
const val = 5;
console.log(typeof val);     // number
```

```
const str = 'hello';
console.log(typeof str);     // string
```

```
const obj = {name: 'Maria', age:18};
console.log(typeof obj);    // object
```

Conditional Statements

Implementing Control-Flow Logic

What is a Conditional Statement?

- The **if-else** statement
  - Do action depending on a specified condition

```
let a = 5;
if (a >= 5) {
  console.log(a);
}
```

If the condition is met, the code will execute

  - You can chain conditions

```
else {
 console.log('no');
}
```

Continue on the next condition, if the first is not met

Chained Conditional Statements

- The **if-else if-else…** construct is a series of checks

```
let a = 5;
if (a > 10)
  console.log("Bigger than 10");
else if (a < 10)
  console.log("Less than 10");
else
  console.log("Equal to 10");
```

Only "**Less than 10**" will be printed

- If one condition is true, it does not proceed to verify the next conditions

Logical Operators

- **Logical operators** give us the ability to write multiple conditions in one `if` statement
- They return a boolean result (`true` or `false`)

Operator	Description	Example
!	NOT	!false → true
&&	AND	true && false → false
||	OR	true || false → true

The Switch-Case Statement

- Works as a series of `if-else if-else if…`

```
switch (...) {
   case ...:
    // code
    break;
   case ...:
    // code
    break;
   default:
    // code
    break;
}
```

List of conditions (values) for the inspection

The condition in the **switch case** is a value

Code to be executed if there is no match with any case

Loops

Code Block Repetition

Loops in JavaScript

- Loops execute a block of code a number of times
- JavaScript supports 5 kinds of loops
  - `for`
  - `for-in`
  - `for-of`
  - `while`
  - `do-while`

Types of Loops

- The **for** loop
  - Loops through a block of code a specified number of times

```
for (let i = 0; i < 5; i++) {
  console.log(i);
}
```

- The **for-of** loop
  - Iterates through all **elements** in an iterable object
  - Cannot access the current index

```
for (let el of collection) {
    // Process the value here
}
```

Types of Loops

- The **while** loop
  - Executes a block of code as long as the specified condition is true

```
while (condition) {
  // code to be executed
}
```

- The **do-while** loop
  - Executes a block of code once, then checks the condition

```
do {
  // code to be executed
}
while (condition);
```

Debugging Techniques

Strict Mode, IDE Debugging Tools

Strict Mode

- **Strict mode** limits certain "sloppy" language features
  - Silent errors will **throw exception** instead

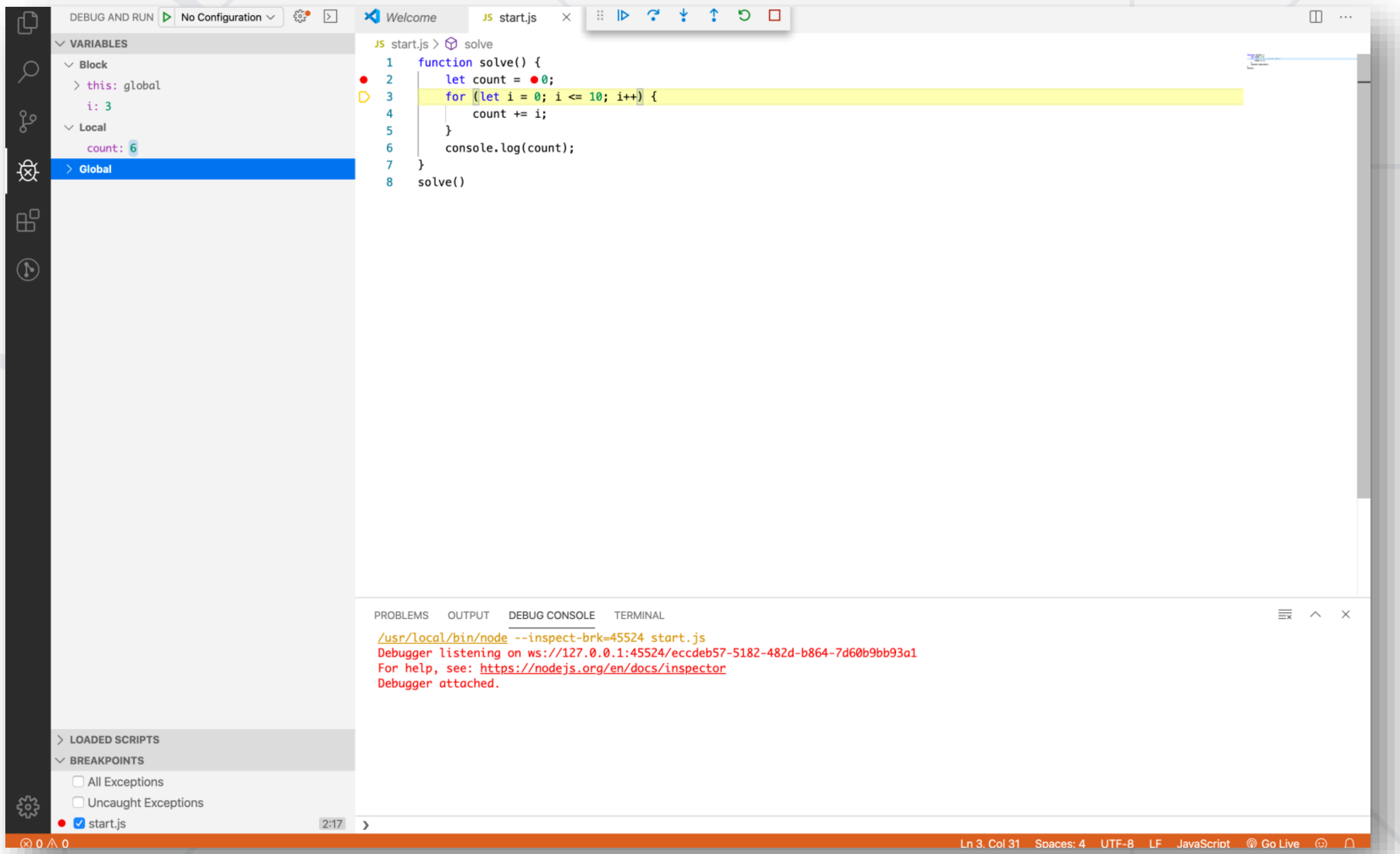
```
'use strict';               // File-level
mistypeVariable = 17; // ReferenceError
```

```
function strict() {
    'use strict';           // Function-level
    mistypeVariable = 17;
}
```

- Enabled by default in **modules**

Debugging in Visual Studio Code

- Visual Studio Code has a built-in **debugger**
- It provides
  - **Breakpoints**
  - Ability to **trace** the code execution
  - Ability to **inspect** variables at runtime



```
function solve() {
    let count = 0;
    for (let i = 0; i <= 10; i++) {
        count += i;
    }
    console.log(count);
}
solve()
```

```
/usr/local/bin/node --inspect-brk=45524 start.js
Debugger listening on ws://127.0.0.1:45524/eccdeb57-5182-482d-b864-7d60b9bb93a1
For help, see: https://nodejs.org/en/docs/inspector
Debugger attached.
```

Using the Debugger in Visual Studio Code

- Start without Debugger: **[Ctrl+F5]**
- Start with Debugger: **[F5]**
- Toggle a breakpoint: **[F9]**
- Trace step by step: **[F10]**
- Force step into: **[F11]**

Summary

- JS == a high-level programming language
- Node.js == server-side JS runtime
- There are objects and 7 primitive data types
- 3 variable types – let, const, var
- Conditional statement – if-else, switch-case
- Loops – for, for-in, for-of, while, do-while
- Different debugging techniques

Software University

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

Software University

SmartIT

Coca-Cola HBC Bulgaria

SOFTWARE GROUP

DXC TECHNOLOGY

createX

Postbank

Решения за твоето утре

SUPER HOSTING .BG

DRAFT KINGS

VIVACOM

INDEAVR

Serving the high achievers

AMBITIONED

PHAR VISION

License

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, about.softuni.bg
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity