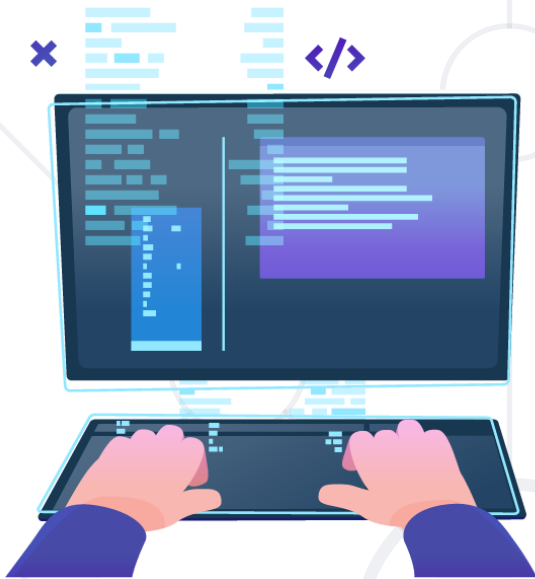


# JS Syntax Fundamentals

Syntax, Conditional Statements, Loops, Data  
Type and Variables, Array



**SoftUni Team**

**Technical Trainers**



**SoftUni**



**Software University**

<https://softuni.bg>

# Table of Contents

1. JavaScript Syntax
2. Data Types and Variables
3. Conditional Statements
4. Loops
5. Undefined and Null
6. Debugging Techniques





sli.do

#js-front-end



# JavaScript Overview

Definition, Execution, IDE Setup

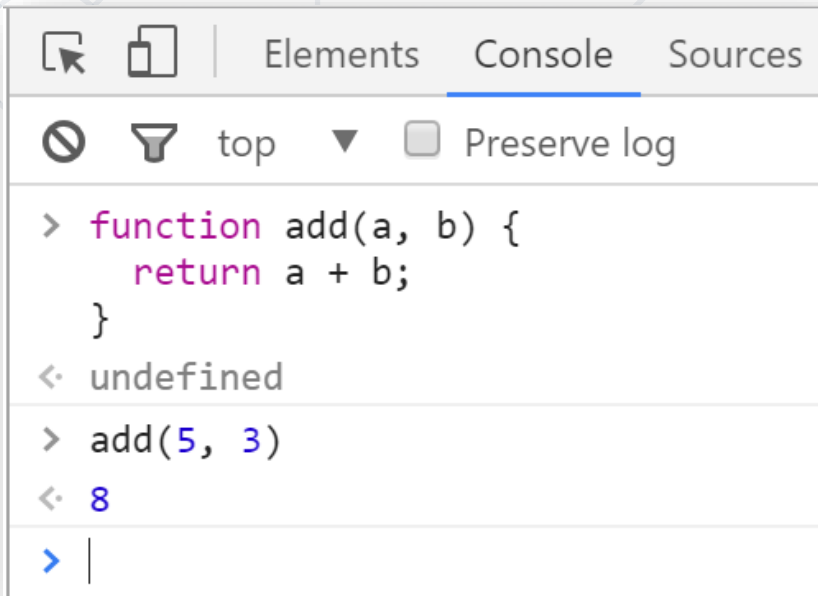
# What is JavaScript?



- 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**

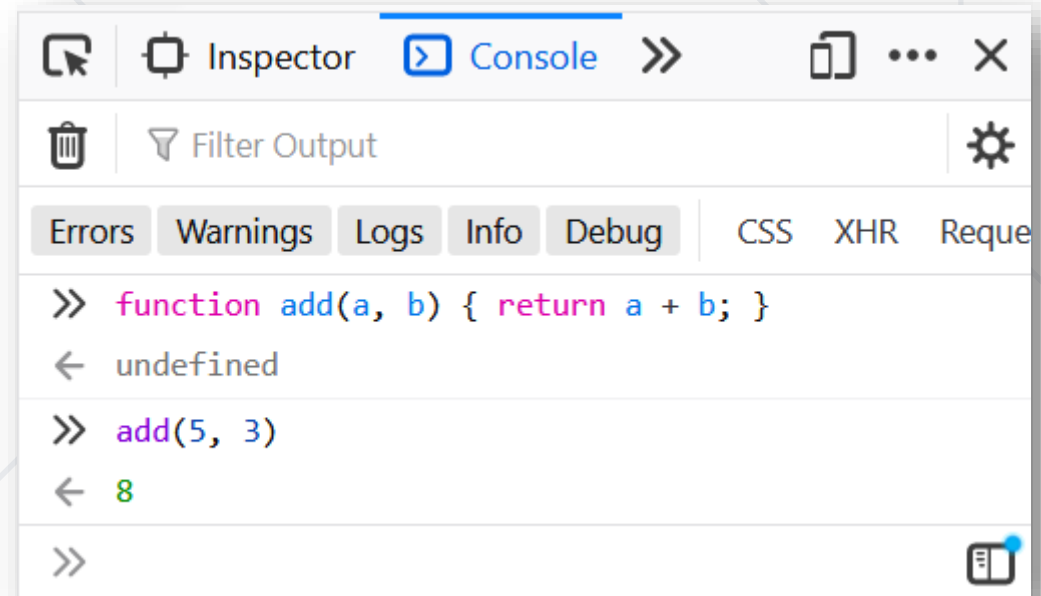
- JavaScript is a **dynamic programming language**
  - Operations otherwise done at **compile-time** can be done at **run-time**
- It is **possible** to change the **type** of a variable or add new properties or methods to an object **while** the program is **running**
- In **static programming languages**,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

Developer Console: **[F12]**



The Chrome Developer Console interface shows the 'Console' tab selected. It includes a toolbar with a close button, a filter icon, a dropdown menu set to 'top', and a 'Preserve log' checkbox. The console log contains the following code and output:

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

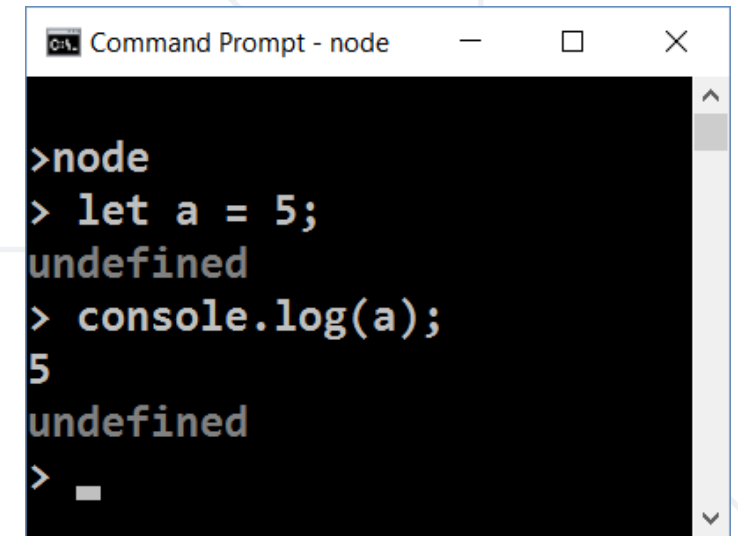


The Firefox Developer Console interface shows the 'Console' tab selected. It includes a toolbar with a close button, a filter icon, and a 'Filter Output' dropdown menu. The console log contains the following code and output:

```
>> 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
>
```



# Install the Latest Node.js

## Downloads

Latest LTS Version: 14.15.4 (includes npm 6.14.10)

Download the Node.js source code or a pre-built installer for your platform, and start developing today.

LTS  
Recommended For Most Users

Current  
Latest Features

  
Windows Installer  
node-v14.15.4-x64.msi

  
macOS Installer  
node-v14.15.4.pkg

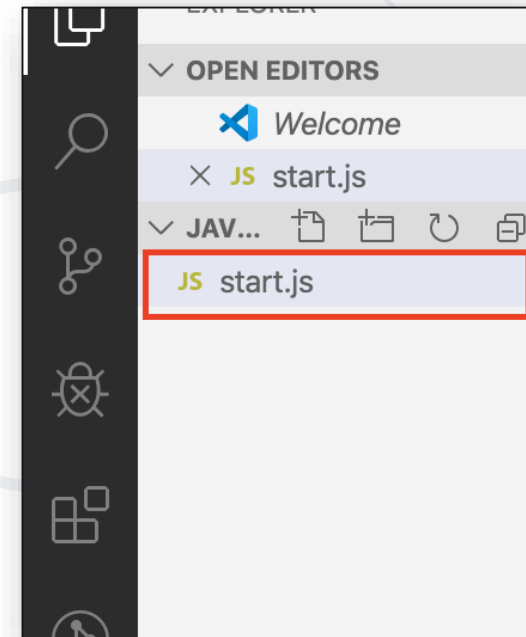
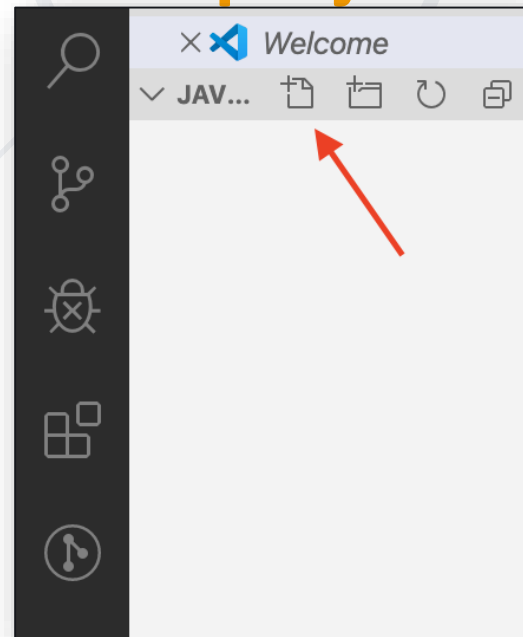
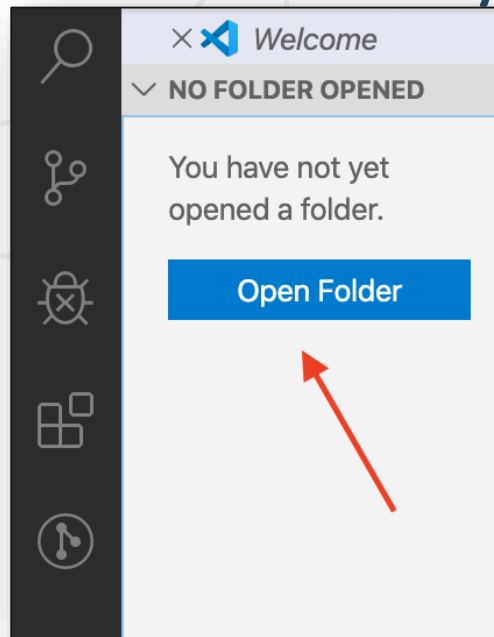
  
Source Code  
node-v14.15.4.tar.gz

- Windows Installer (.msi)
- Windows Binary (.zip)
- macOS Installer (.pkg)
- macOS Binary (.tar.gz)
- Linux Binaries (x64)
- Linux Binaries (ARM)
- Source Code

32-bit	64-bit
32-bit	64-bit
64-bit	
64-bit	
64-bit	
ARMv7	ARMv8
node-v14.15.4.tar.gz	

# Using Visual Studio Code

- **Visual Studio Code** is powerful text editor for JavaScript and other projects
- In order to create your **first project**:






# JavaScript Syntax

Functions, Operators, Input and Output

# JavaScript Syntax

- C-like **syntax** (curly-brackets, identifiers, operator)
- Defining and Initializing variables:



Declare a variable with let

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

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**, that executes when called

declaration

parameters

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

```
solve(2, 3);
```

calling the function

- 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:

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

- To format a number, use the **toFixed()** method (converts to **string**):

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

Number of decimal places



# **Data Types and Variables**

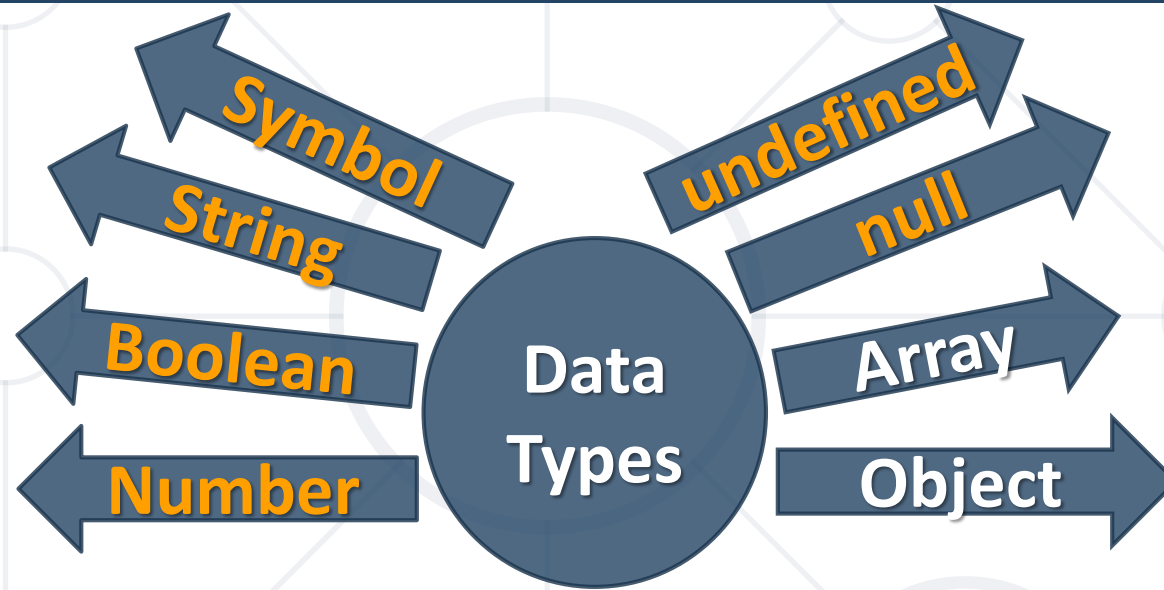
Definition and Examples

# What is a Data Type?

- A **data type** is a classification that specifies what type of operations can be applied to it and the way values of that type are stored
- After **ECMAScript** 2015 there are **seven primitive** data types:
  - Seven **primitive**: Boolean, null, undefined, Number, String, Symbol, BigInt
  - and **Objects** (including Functions and Arrays)







```
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
- **let** and **const** – use **block scope** – when declared inside a block **{ }** can **NOT** be accessed from outside the block

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

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



# Variable Comparison let vs const

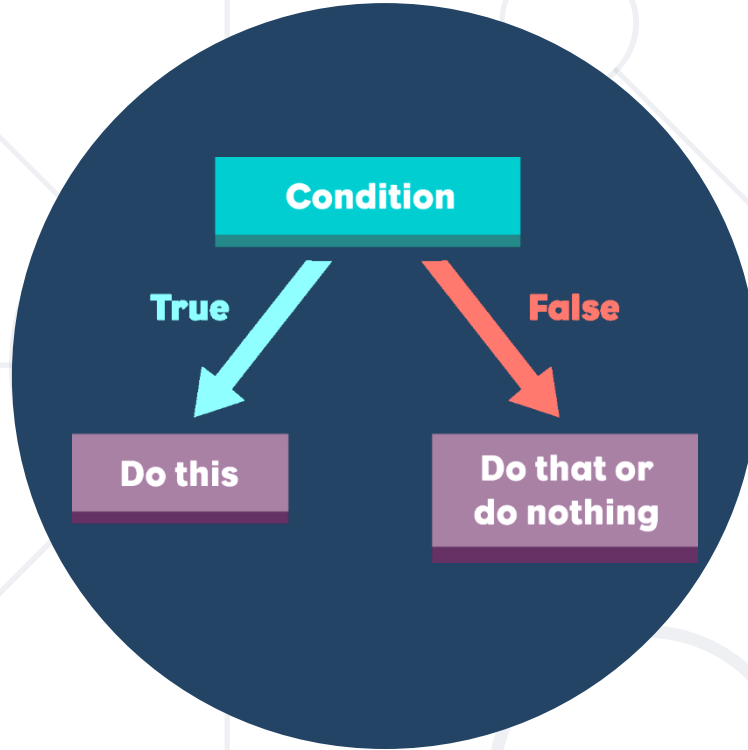
## ■ **let**

- 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 reassigned





# Conditional Statements

Implementing Control-Flow Logic

# 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; // 155
= 759375c
```



# Comparison Operators

```
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

# What is a Conditional Statement?

The **if-else** statement:

- Do action depending on 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 following conditions



# 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

- **Logical operators** are used to determine the logic between variables or values. They return the value of one of the operands based on certain rules, not always just (**true** or **false**).

Operator	Description	Example
!	NOT	!false -> true
&&	AND	true && false -> false
	OR	true    false -> true

- Logical **"AND"**

- Checks the fulfillment of several conditions simultaneously

```
let a = 3;  
let b = -2;  
console.log(a > 0 && b > 0); // expected output: false
```

- Logical **"OR"**

- Checks that at least one of several conditions is met

```
let a = 3;  
let b = -2;  
console.log(a > 0 || b > 0); // expected output: true
```

# Logical Operators: Examples (2)

- Logical "NOT"
  - Checks if a condition is **not** met

```
let a = 3;  
let b = -2;  
console.log(!(a > 0 || b > 0));  
// expected output: false
```

# 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
```

# Truthy and Falsy Values

- **"truthy"** - a value that **coerces** to **true** when **evaluated** in a boolean context
- The following values are **"falsy"** - **false**, **null**, **undefined**, **NaN**, **0**, **0n** and **""**

```
function logTruthiness (val) {  
  if (val) {  
    console.log("Truthy!");  
  } else {  
    console.log("Falsy.");  
  }  
}
```

```
logTruthiness (3.14);           //Truthy!  
logTruthiness ({});             //Truthy!  
logTruthiness (NaN);            //Falsy.  
logTruthiness ("NaN");          //Truthy!  
logTruthiness ([]);             //Truthy!  
logTruthiness (null);           //Falsy.  
logTruthiness ("");             //Falsy.  
logTruthiness (undefined);      //Falsy.  
logTruthiness (0);              //Falsy.
```



# Loops

Code Block Repetition

# What is a Loop?

The **for** loop:

- Repeats until the condition is evaluated

```
for (let i = 1; i <= 5; i++){  
  console.log(i)  
}
```

Incrementation **in**  
the condition

The **while** loop:

- Does the same, but has different structure

```
let i = 1  
while (i <= 5) {  
  console.log(i)  
  i++  
}
```

Incrementation  
**outside** the  
condition







**Undefined  
Null**

**Undefined and Null**

Non-Existent and Empty

# 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
```

# Null and Undefined

- **Null** is an assigned value. It means nothing
- **Undefined** typically means a variable has been declared but not defined yet
- **Null** and **Undefined** are **falsy** values
- **Undefined** and **Null** are equal in value but different in type:

```
null !== undefined    // true  
null == undefined    // true
```



# Debugging Techniques

Strict Mode, IDE Debugging Tools

# Strict Mode

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



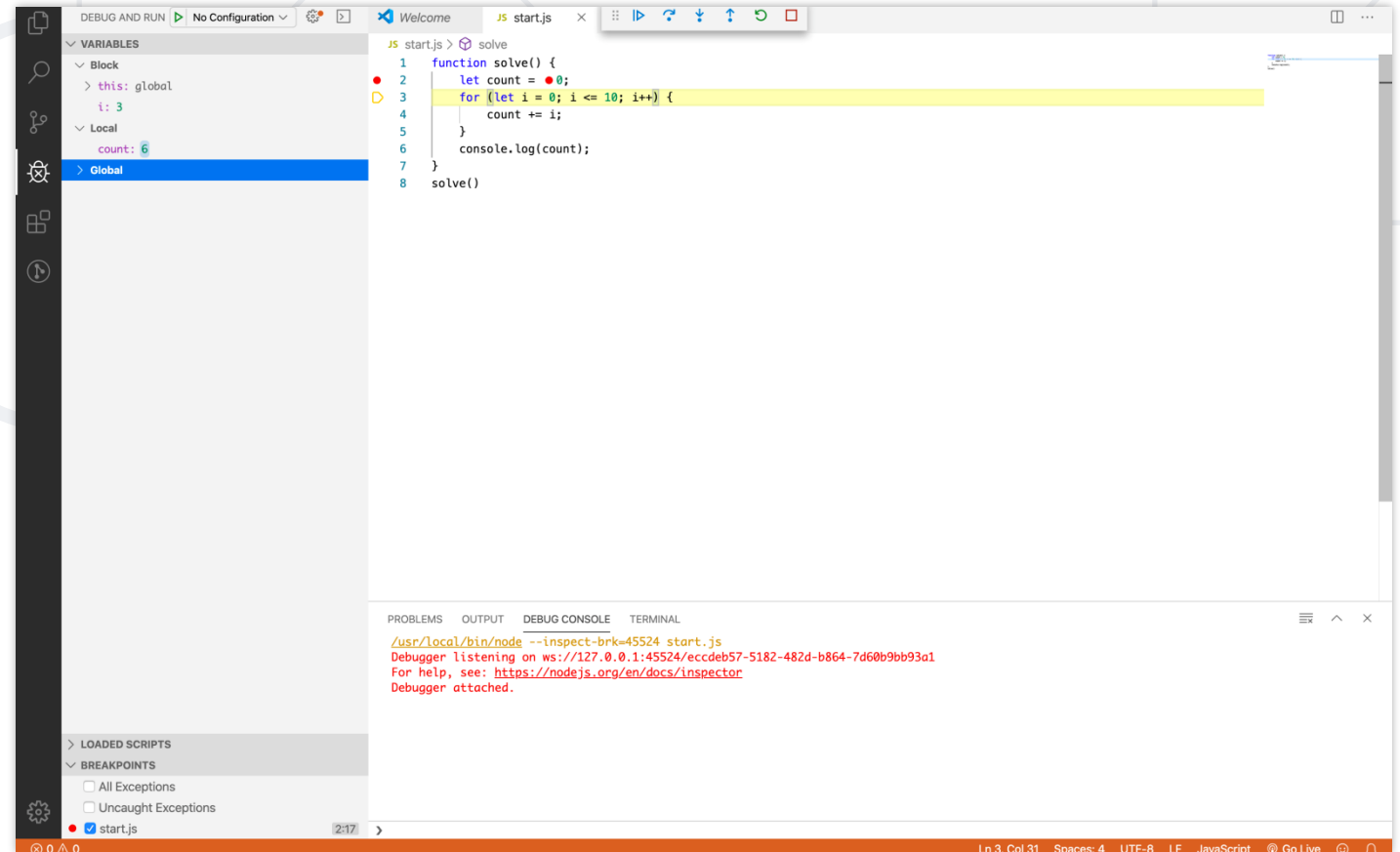
```
'use strict';           // File-level  
mistypeVariable = 17;    // ReferenceError
```

```
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



# Using the Debugger in Visual Studio Code

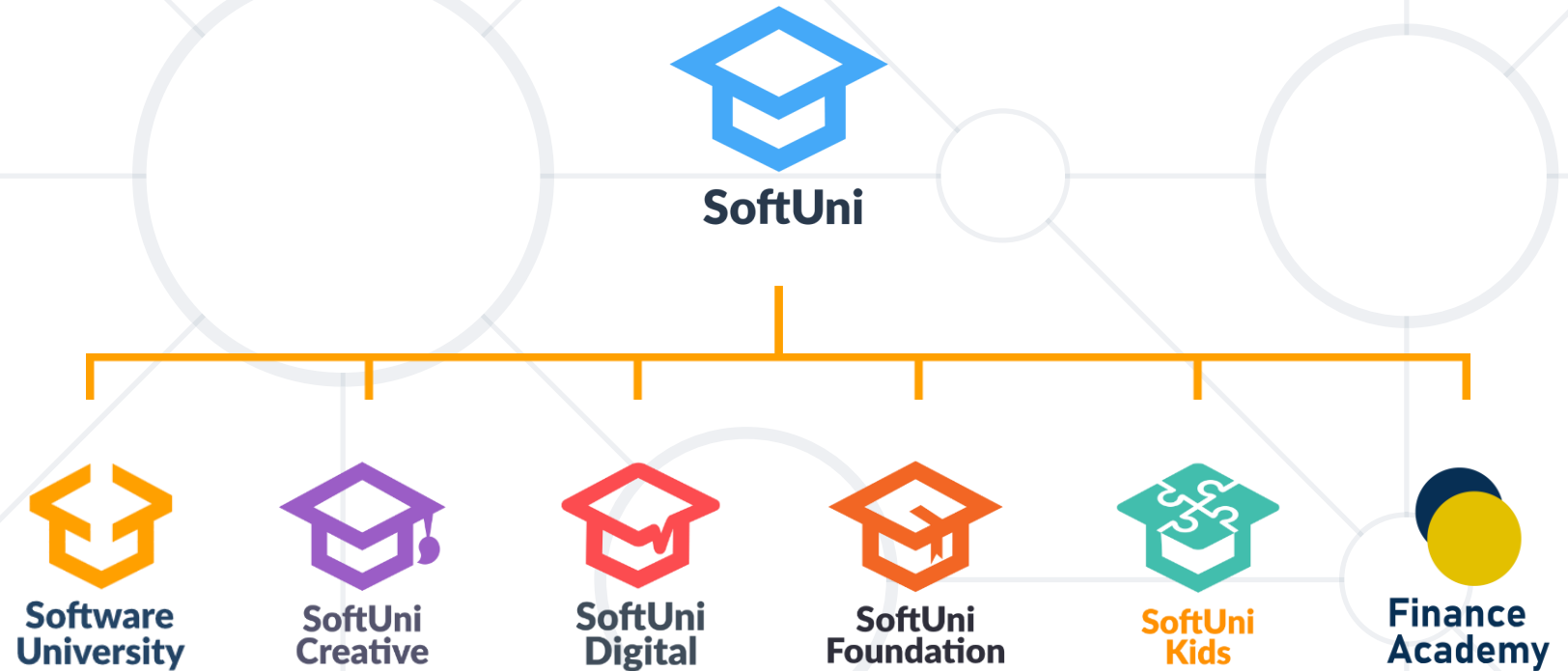
- Start without Debugger: **[Ctrl+F5]**
- Start with Debugger: **[F5]**
- Toggle a breakpoint: **[F9]**
- Trace step by step: **[F10]**
- Force step into: **[F11]**



- JS is a **high-level** programming language
- Conditional statement – **If-else, Switch-case**
- Loops – **For-loop, While-loop**
- Data Types
  - **String, Number, Boolean, Null, Undefined**



# Questions?



# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**

 **Flutter**<sup>TM</sup>  
International

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

 **DRAFT  
KINGS**



**BOSCH**

 **Postbank**  
*Решения за твоето утре*

 **PHAR  
VISION**



**SmartIT**

**DXC**  
TECHNOLOGY

**createX**

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – <https://about.softuni.bg/>
- © Software University – <https://softuni.bg>



- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [about.softuni.bg](http://about.softuni.bg)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)

