Introduction to JavaScript

JS Syntax, JS IDE, Mix HTML and JS,
JS Data Types, Variables and Scope

SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

Software University

http://softuni.bg

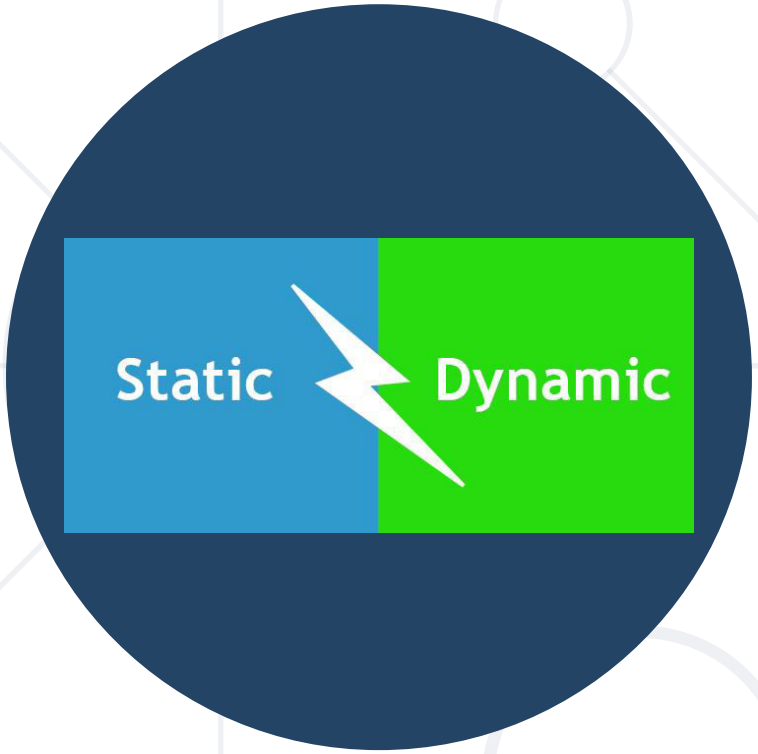
Have a Question?

SoftUni Foundation

sli.do

#JSCORE

Table of Content



Static vs. Dynamic Languages

C++ / C# / Java vs. JS / PHP / Python

Static vs. Dynamic Languages

SoftUni Foundation

Static Languages

- Explicit type declaration

```
string name = "Pesho";
int age = 25;
```

- Strongly typed
- Type checking occurs at compile-time
- Statically-typed languages: **C, C++, C#, Java**

Dynamic (Scripting) Languages

- Implicit type declaration

```
let name = "Pesho"
let age = 25;
```

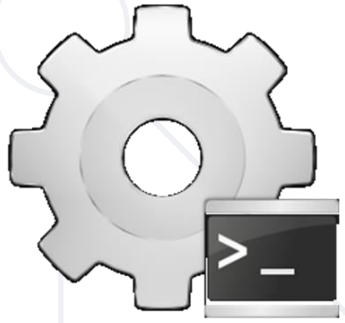
- Weakly typed
- Type checking occurs at run-time
- Dynamic languages: **JavaScript, PHP, Python, Ruby**

Compilers

- A **compiler** converts a high level language code (like C#, Java) into **machine code**
  - Machine code is executed by the **CPU** or **VM**
  - Most errors are found at **compile time**, before execution
  - **C** and **C++** use compilers to produce **native code** for the CPU
  - **C#** and **Java** use compilers to produce **intermediate code** for VM

Interpreters

- An **interpreter** executes a script code line by line
  - Code is analyzed at **run-time** (no compilation)
  - Errors are found at **run-time**, during the code execution
  - **JavaScript**, **Python**, **Ruby** and **PHP** use interpreters
- **JIT** (just-in time compilers) compile to **native code** at runtime
  - JIT occurs during the execution, on demand
  - Available for Java, JavaScript and other languages



Virtual Machines (VM)

- **Virtual machine (VM)** is
  - A virtual computer (computer inside the computer)
  - Runs an intermediate code for a virtual CPU
- VM-based languages:
  - **JVM** (Java Virtual Machine) for Java
  - **CLR** (Common language Runtime) for C# / .NET
  - **HHVM** for PHP

Compiler vs. Interpreter vs. Virtual Machine

- Compiled languages are **faster** than interpreted languages
  - C, C++, Go and Swift are very **fast**
    - Compiled to native code for CPU
  - C# and Java are **slower**
    - Compiled to intermediate code for VM
  - JS, Python, Ruby, PHP are **even slower**
    - Interpreted / JIT-compiled

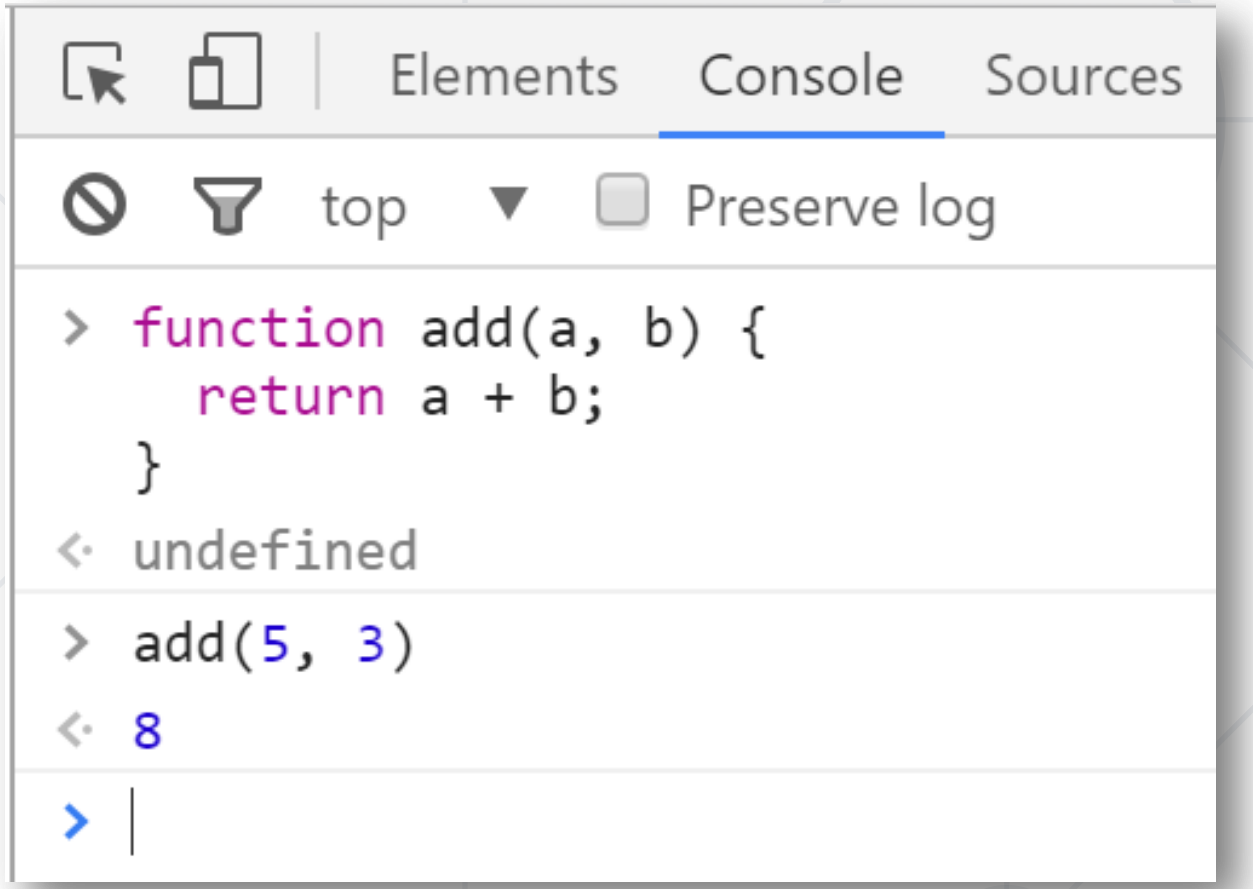
JavaScript IDEs

Development Environments for JS

Chrome Web Browser

SoftUni Foundation

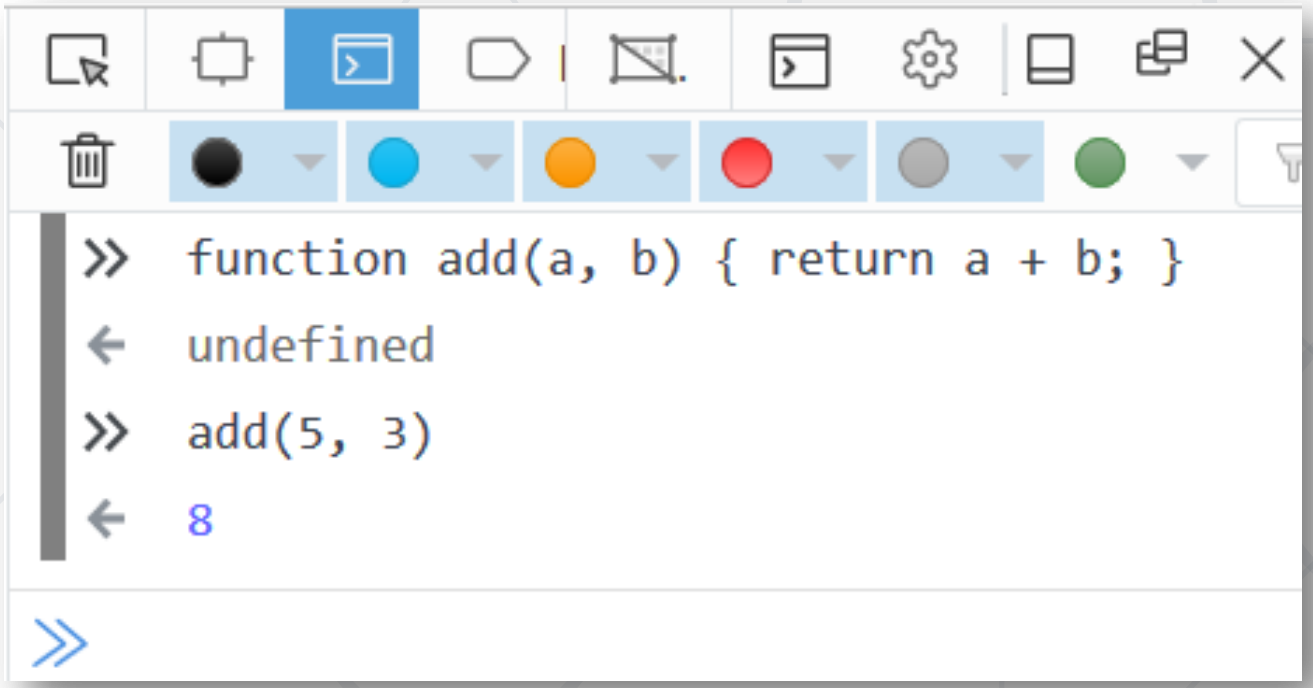
Developer Console: [F12]



Firefox Web Browser

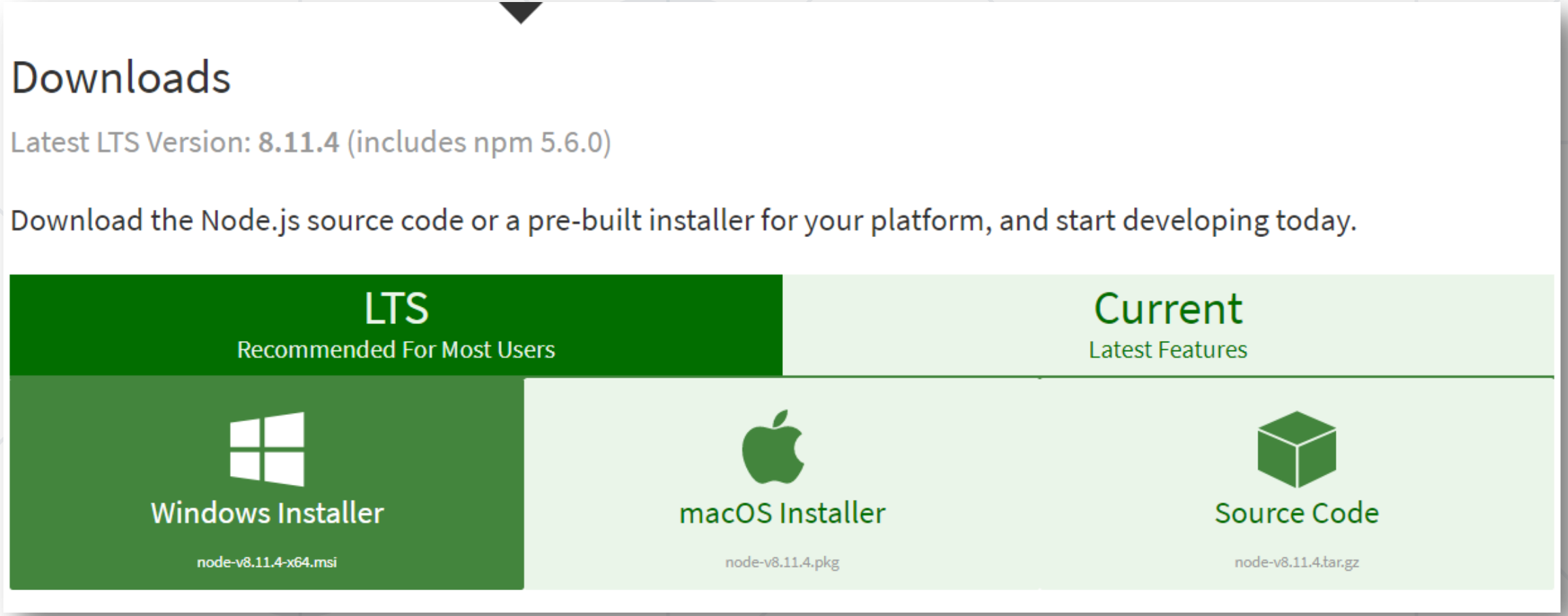
SoftUni Foundation

Developer Console: **[Ctrl] + [Shift] + [i]**



Install the Latest Node.js

SoftUni Foundation



Node.js

- What is **Node.js**?
  - Server-side JavaScript runtime
  - Chrome **V8** JavaScript engine

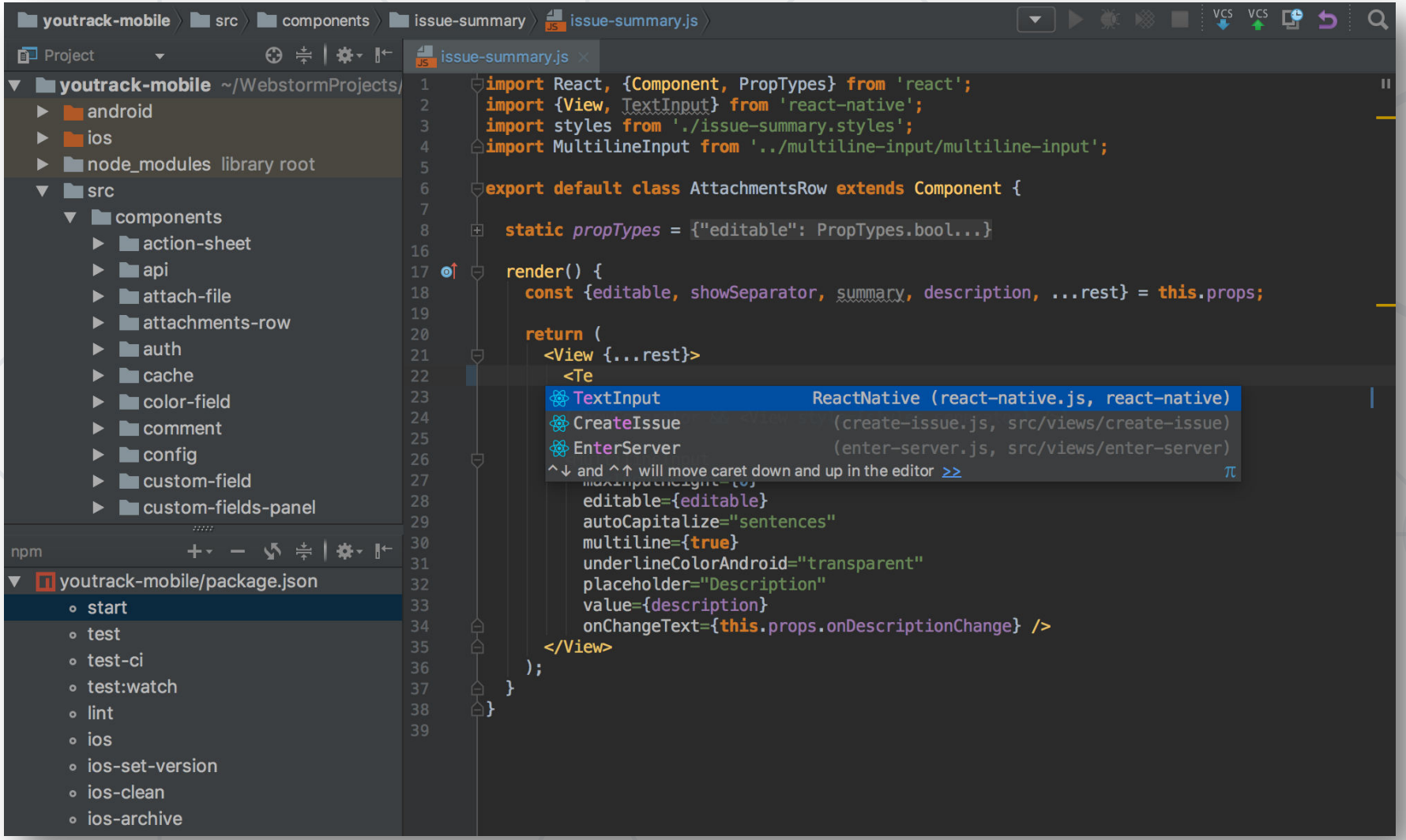
```
Command Prompt - node
>node
> let a = 5;
undefined
> console.log(a);
5
undefined
>
```

```
Command Prompt
>npm install express
C:\Trash\node-test
`-- express@4.14.0
  +-- accepts@1.3.3
  | +-- mime-types@2.1.11
  | | `-- mime-db@1.23.0
  | `-- negotiator@0.6.1
  +-- array-flatten@1.1.1
  +-- content-disposition@0.5.1
```

- **npm** package manager
  - Install node packages

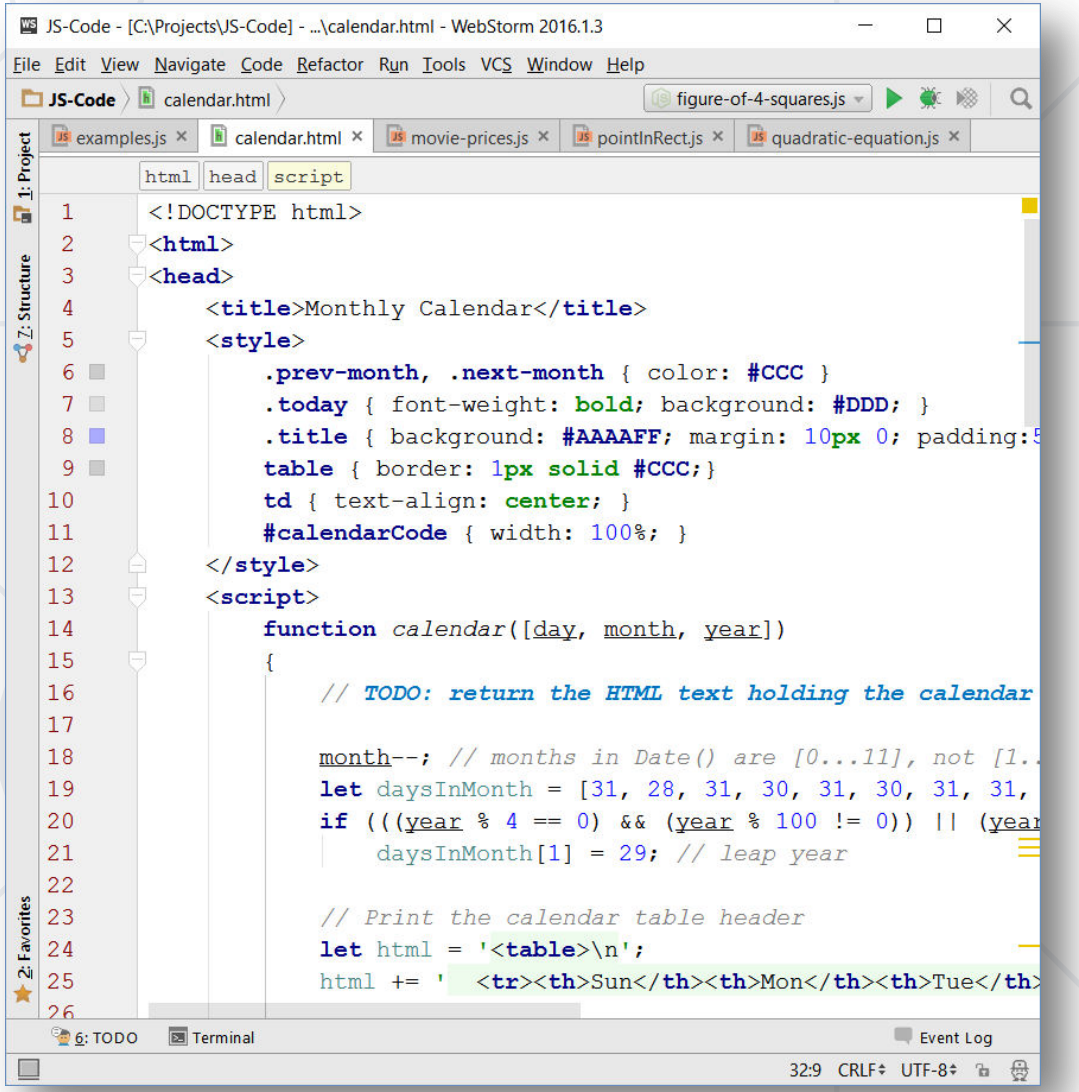
Install WebStorm

SoftUni Foundation



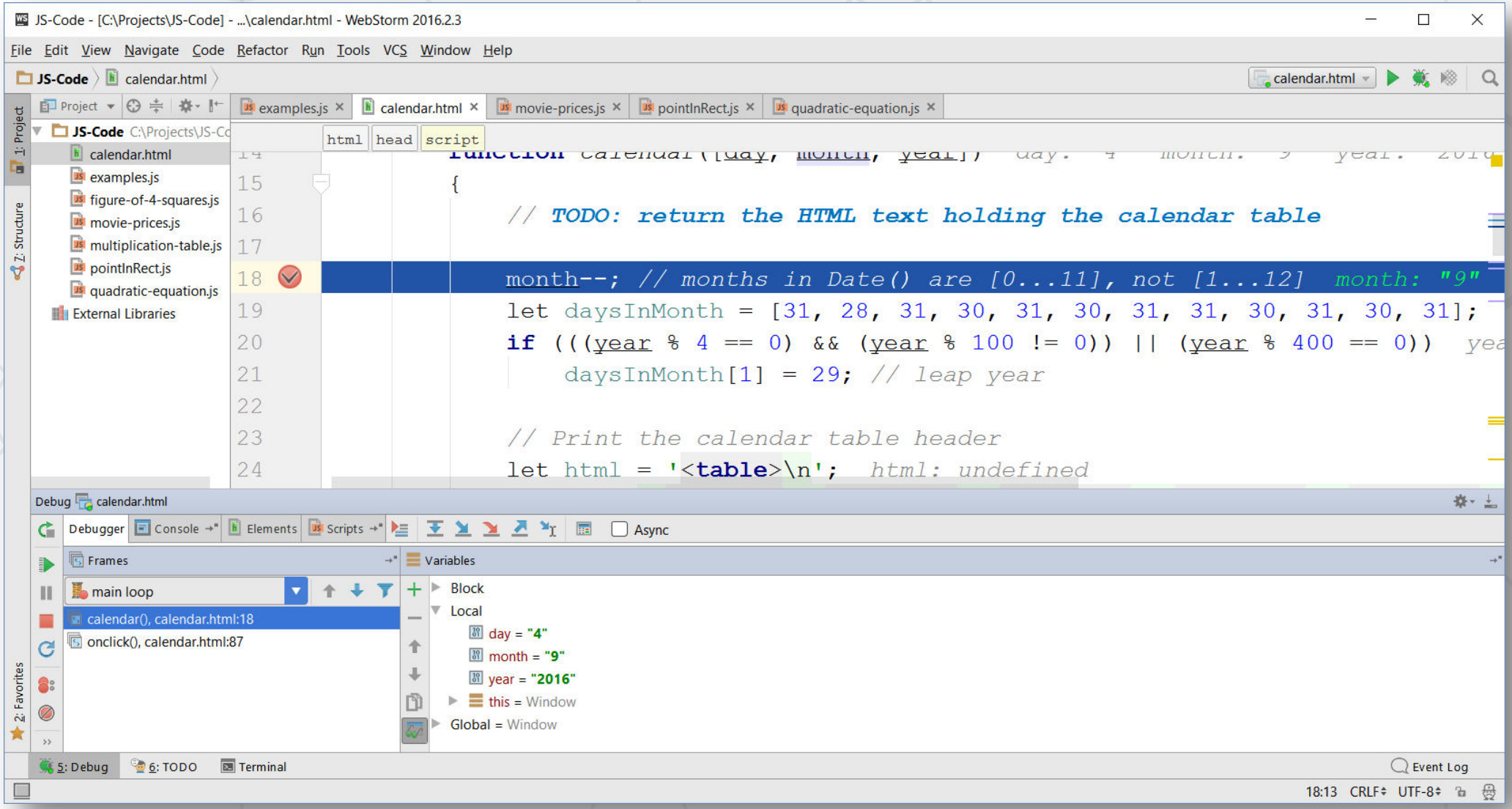
WebStorm

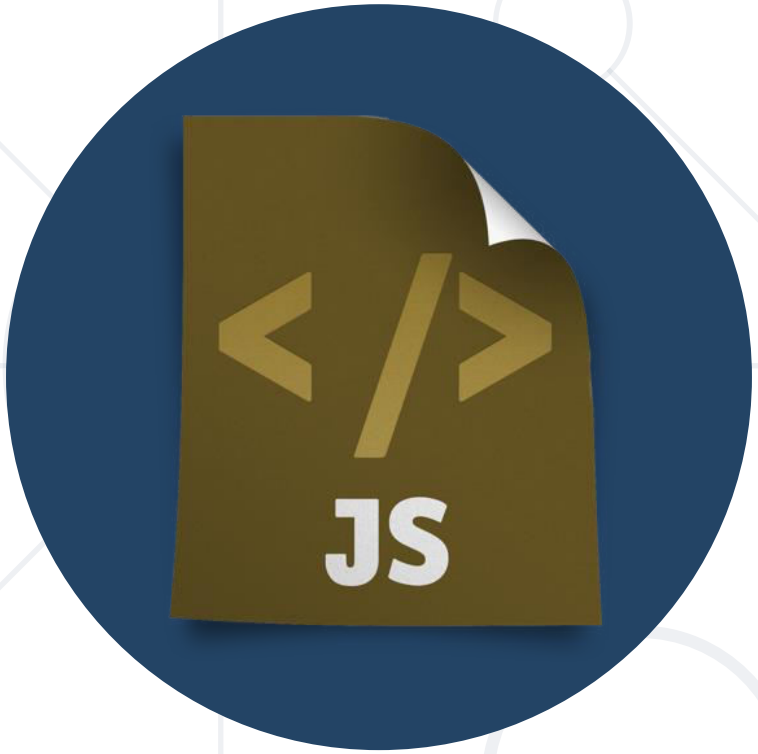
- Powerful IDE for HTML5
  - JavaScript, HTML, CSS
  - Babel, TypeScript, Sass / LESS, …
- Code, run, debug JS code
- Paid product
  - Free license for SoftUni students



WebStorm Debugger

SoftUni Foundation





JavaScript Syntax

JavaScript Syntax

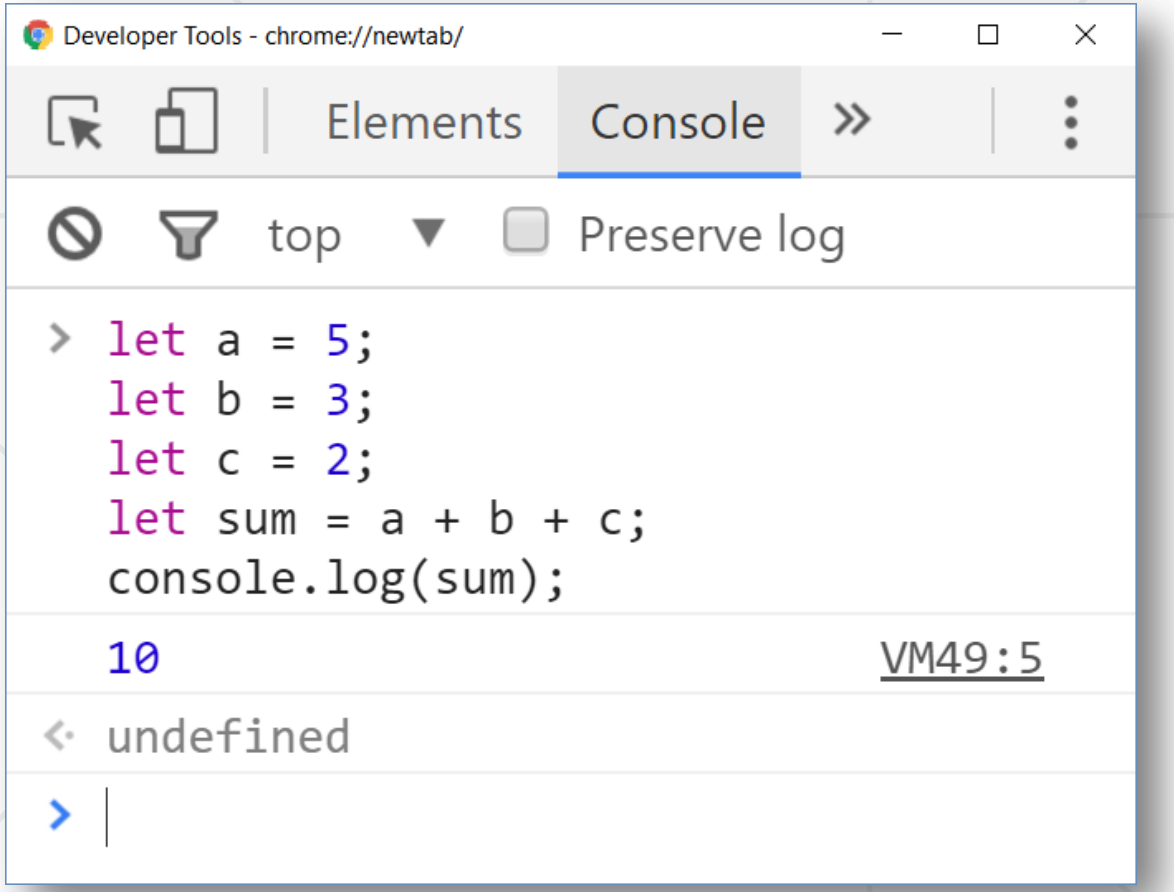
SoftUni Foundation

- The JavaScript syntax is similar to C#, Java and PHP
  - Operators (**+, *, =, !=, &&, ++, …**)
  - Variables (typeless)
  - Conditional statements (**if, else**)
  - Loops (**for**, **while**)
  - Functions (with parameters and return value)
  - Arrays (**arr[5]**) and associative arrays (**arr['maria']**)
  - Objects and classes

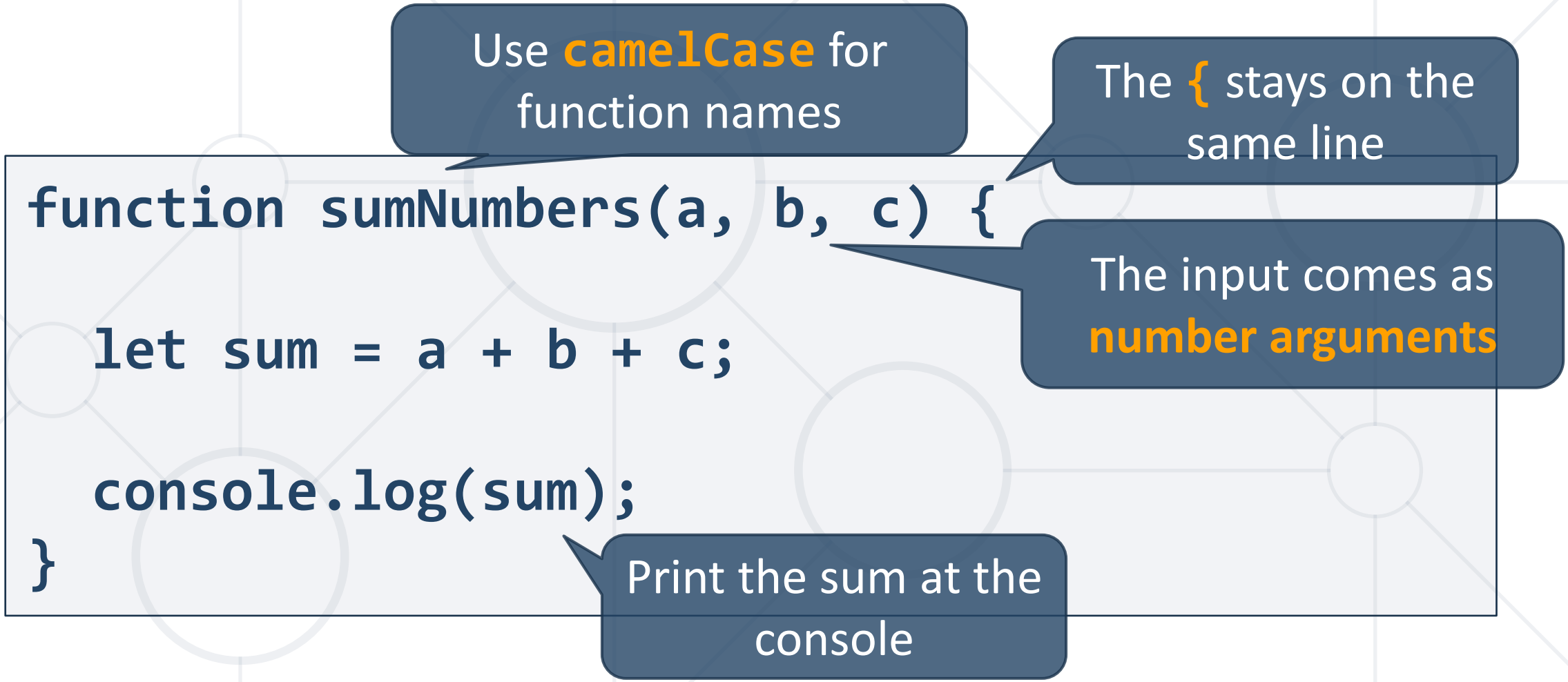
JavaScript Code: Example

- JS variables, operators, expressions, statements

```
let a = 5;
let b = 3;
let c = 2;
let sum = a + b + c;
console.log(sum);
```



Problem: Function to Sum 3 Numbers



Check your solution here: https://judge.softuni.bg/Contests/287

Test the Function in Your Browser

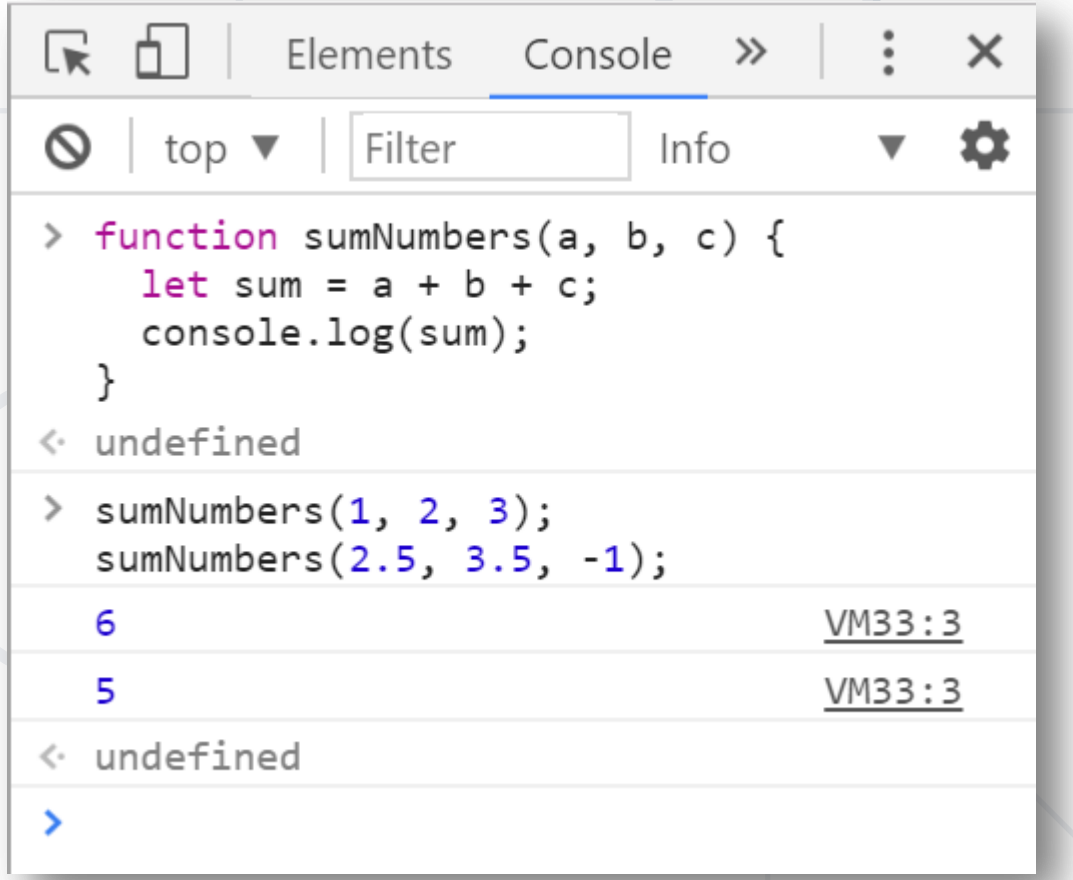
SoftUni Foundation

- Call the function with three arguments holding numbers:

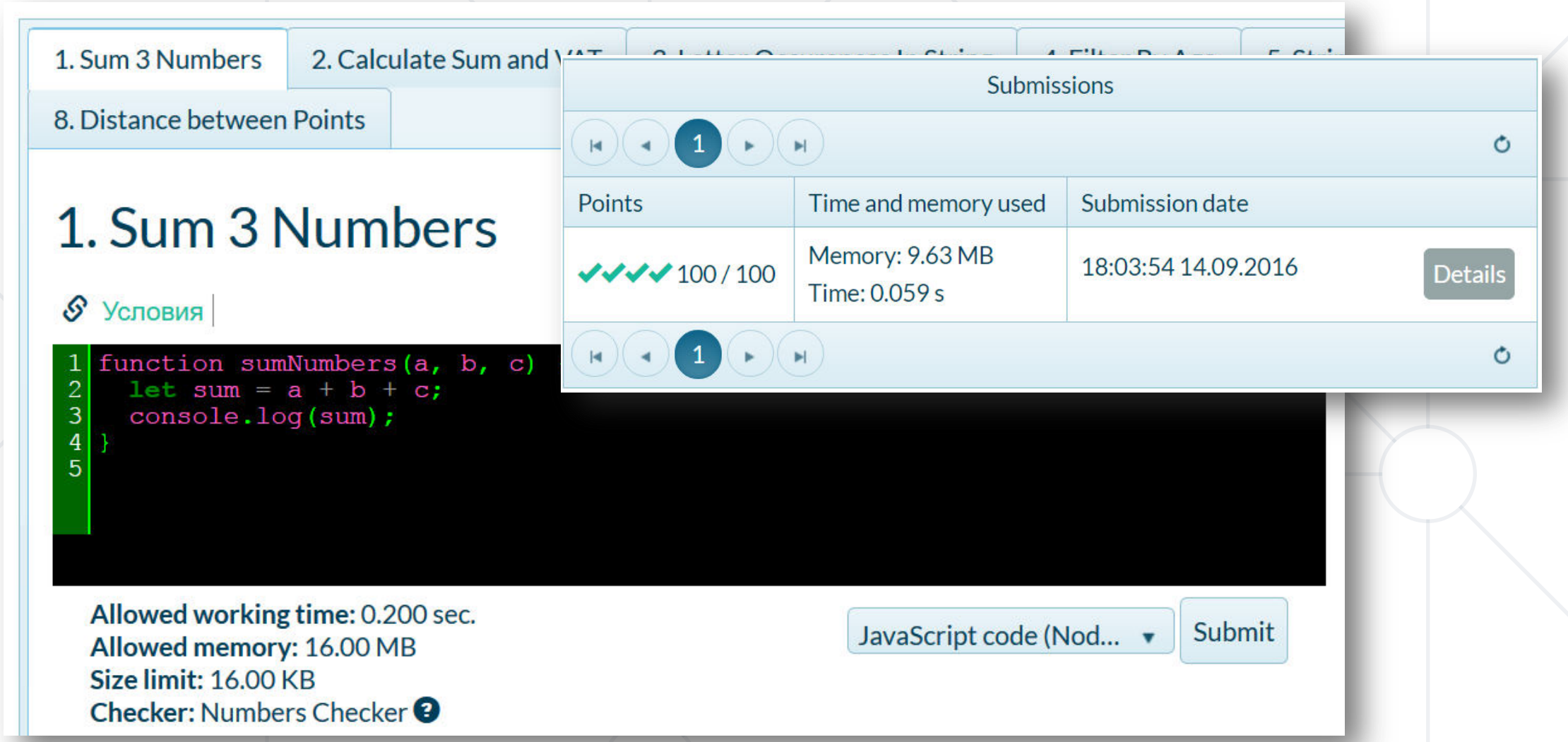
```
sumNumbers(1, 2, 3);
// 6
```

```
sumNumbers(2.5, 3.5, -1);
// 5
```

```
sumNumbers(-1, -2, -3);
// -6
```



Test the Function in the Judge System



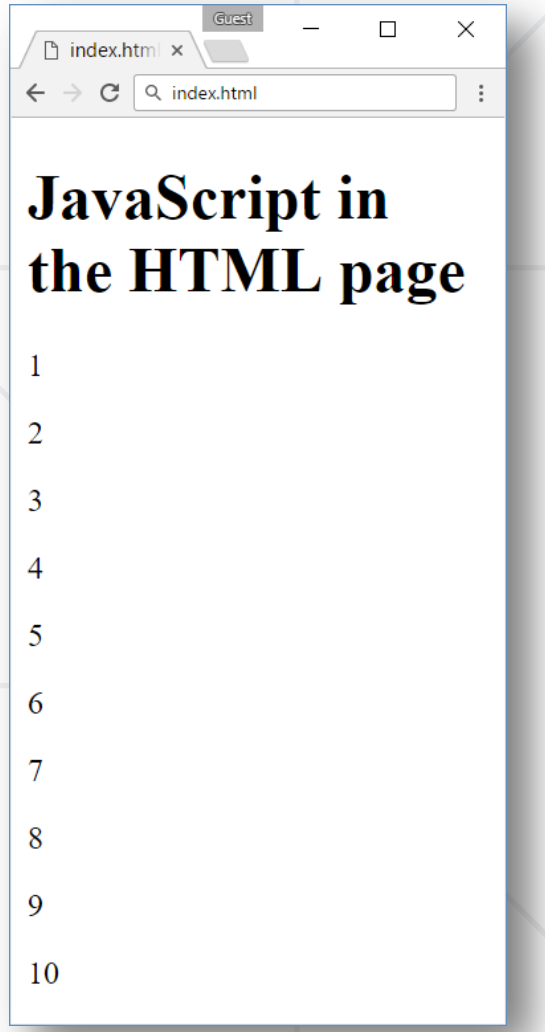
Mix HTML and JavaScript

Using JS Code from HTML Page

Mixing HTML + JavaScript

SoftUni Foundation

```
<!DOCTYPE html>
<html>
<body>
  <h1>JavaScript in the HTML page</h1>
  <script>
    for (let i=1; i<=10; i++) {
      document.write(`<p>${i}</p>`);
    }
  </script>
</body>
</html>
```



Sum Numbers with HTML Form

SoftUni Foundation

```
<form>
  num1: <input type="text" name="num1" /> <br>
  num2: <input type="text" name="num2" /> <br>
  sum: <input type="text" name="sum" /> <br>
  <input type="button" value="Sum" onclick="calcSum()" />
</form>
```

```
function calcSum() {
  let num1 = document.getElementsByName('num1')[0].value;
  let num2 = document.getElementsByName('num2')[0].value;
  let sum = Number(num1) + Number(num2);
  document.getElementsByName('sum')[0].value = sum;
}
```

Load JavaScript File from HTML Document

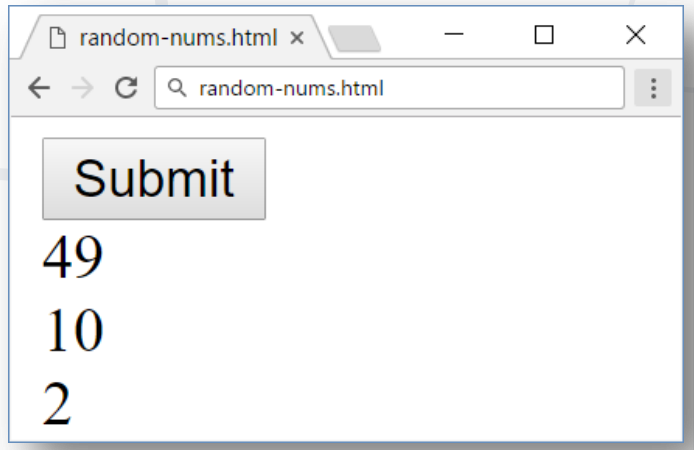
SoftUni Foundation

random-nums.html

```
<!DOCTYPE html>
<html>
<head>
  <script src="numbers.js">
  </script>
</head>
<body>
  <input type="submit"
onclick="printRandNum()" />
</body>
</html>
```

numbers.js

```
function printRandNum() {
  let num = Math.round(
    Math.random() * 100);
  document.body.innerHTML +=
    `<div>${num}</div>`;
}
```



Data Types in JS

Numbers, Strings, Booleans, Objects

Numbers in JavaScript

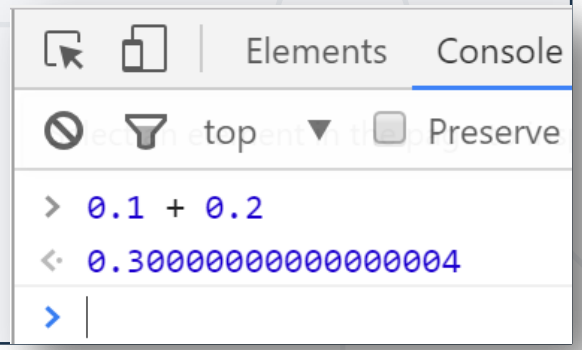
SoftUni Foundation

- All numbers in JS are floating-point (64-bit double-precision)

```
let n = 5;
console.log(typeof(n) + ' ' + n); // number 5

let bin = 0b10000001;
console.log(typeof(bin) + ' ' + bin); // number 129

let x = Number.parseInt("2.99");
console.log(typeof(x) + ' ' + x); // number 2

let y = Number("2.99");
console.log(typeof(y) + ' ' + y); // number 2.99
```

Numbers in JavaScript are floating-point

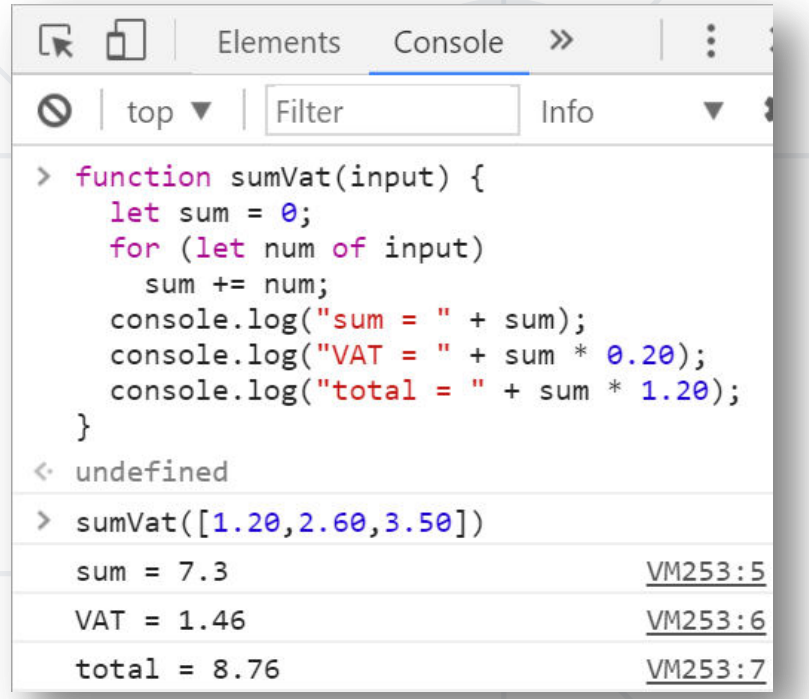
```
let a = 3, b = 4 / a;
console.log(`a=${a} b=${b}`); // a=3
b=1.3333333333333333
console.log(a / 0); // Infinity
console.log(a * Infinity); // Infinity
console.log(0 / 0); // NaN
let bigNum = 10000000000000000;
console.log(bigNum + 1); // 10000000000000000
let num = 0.01, sum = 0;
for (let i=0; i<100; i++)
  sum += num;
console.log(sum); // 1.0000000000000007
```



Problem: Calculate Sum + VAT

- You are given an array of numbers
  - Print their **sum, 20% VAT** and **total sum**

```
function sumVat(input) {
  let sum = 0;
  for (let num of input)
    sum += num;
  console.log("sum = " + sum);
  console.log("VAT = " + sum * 0.20);
  console.log("total = " + sum * 1.20);
}
sumVat([1.20,2.60,3.50])
```



Check your solution here: https://judge.softuni.bg/Contests/287

Strings in JavaScript

- **Strings** in JS are **immutable** sequences of **Unicode** characters

```
let ch = 'x';
console.log(typeof(ch) + ' ' + ch); // string x

let str = "hello";
console.log(typeof(str) + ' ' + str); // string hello
console.log(str.length); // 5

console.log(str[0]); // h
str[0] = 's'; // Beware: no error, but str stays
unchanged!
console.log(str); // hello
console.log(str[10]); // undefined
```

In **strict mode** this gives error

Problem: Letter Occurrences in String

- We are given a string (text) and a letter
  - Count how many times given letter occurs in given string

```
function countLetter(str, letter) {
  let count = 0;
  for (let i=0; i<str.length; i++)
    if (str[i] == letter)
      count++;
  return count;
}
```

```
countLetter('hello', 'l')
```

The input comes as two strings: `str` and `letter`

Check your solution here: https://judge.softuni.bg/Contests/287

Objects in JavaScript

SoftUni Foundation

- JS **objects** hold key-value pairs

```
let point = {x: 5, y: -2};
console.log(point); // Object {x: 5, y: -2}
console.log(typeof(point)); // object

let rect = {name:"my room", width: 4.5, height: 3.5};
rect.name = "old room";
rect.color = "red";
console.log(rect); // Object {name: "old room", width:
4.5, height: 3.5, color: "red"}
rect = undefined; // "rect" now has no value
console.log(rect); // undefined
```

Problem: Filter By Age

- We are given a **minimum age** and two pairs of **names** and **ages**
  - Store each name and age into an object
  - Print each object whose age ≥ minimum age

```
function filterByAge(minAge, nameA, ageA, nameB, ageB)
{
  let personA = {name:nameA, age:ageA};
  let personB = {name:nameB, age:ageB};
  if (personA.age >= minAge) console.log(personA);
  if (personB.age >= minAge) console.log(personB);
}
```

```
filterByAge(12, 'Ivan', 15, 'Asen', 9)
```

Check your solution here: https://judge.softuni.bg/Contests/287

Automatic Type Conversion

SoftUni Foundation

- JS converts between types automatically

```
let ch = 'x'; // string
console.log(typeof(ch));
ch++; // ch is now number
console.log(typeof(ch));
console.log(ch); // NaN
```

```
let num = 0xFF; // 255
console.log(typeof(num));
num = num + "x"; // string
console.log(typeof(num));
console.log(num); // 255x
```

```
let n = "1" * 1; // number
let b = Boolean(n);
console.log(typeof(b));
console.log(b); // true
```

```
let n = [1]; // array
console.log(typeof(n));
n++; // n is now 2 (number)
console.log(typeof(n));
```

Problem: String of Numbers 1…N

- Write a JS function that concatenates numbers as text
  - Input: a string parameter, holding a number **n**
  - Output: the numbers **1…n**, concatenated as single string
  - Example: **'11' → "1234567891011"**

```
function numbers(n) {
  let str = '';
  for (let i=1; i<=n; i++) str += i;
  return str;
}
```

```
numbers('11')
```

Check your solution here: https://judge.softuni.bg/Contests/287

Variables and Scope

`let` vs. `var`

Assigning Variables

- Variables in JS have no type, values have type

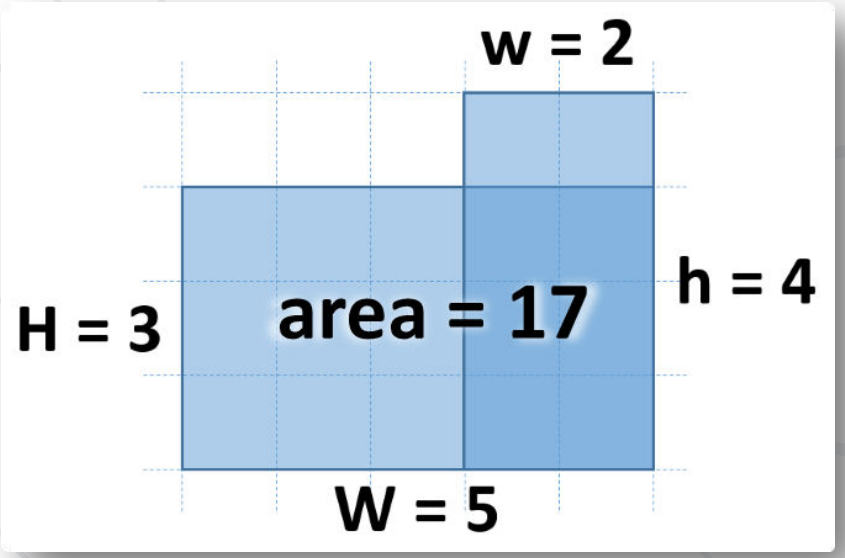
```
let x = 5; console.log(typeof(x)); // x is number
x = 'str'; console.log(typeof(x)); // x is now string
x = {x:5, y:7}; console.log(typeof(x)); // x is now
object
```

```
let n = 5, str = "hello"; // Assigning variables
```

```
// Group declaration + assignment (destructuring)
let [name, x, y] = ["point", 5, 7];
console.log(`${name}(${x}, ${y})`); // point(5, 7)
```

Problem: Figure Area

- Calculate the figure area by given **w**, **h**, **W** and **H**



  - Sample solution:

```
function figureArea(w, h, W, H) {
  let [s1, s2, s3] = [w * h, W * H,
    Math.min(w, W) * Math.min(h, H)];
  return s1 + s2 - s3;
}
```

```
figureArea(2, 4, 5, 3)
```

Check your solution here: https://judge.softuni.bg/Contests/287

Undefined Variables

- JS cannot access undefined variables

```
console.log(x);
// Uncaught ReferenceError: x is not defined(…)
```

- Once defined, variables cannot be deleted!

```
let x = 5;
x = undefined;
console.log(x); // undefined
// The variable x exists, but has no value defined
```

Variable Scope: let vs. var

- let has "block scope"

```
for (let x=1; x<=5; x++)
  console.log(x); // 1 … 5
console.log(x); // Ref. error
```

```
function test() {
  console.log(x); // Ref. error
  let x = 5;
}
test();
```

- var has "function scope"

```
for (var x=1; x<=5; x++)
  console.log(x); // 1 … 5
console.log(x); // 6
```

```
function test() {
  console.log(x); // undefined
  var x = 5;
}
test();
```

Variable Scope: let vs. var (2)

- let has "block scope"

```
let x = 100; // block scope
console.log(x); // 100
for (let x=1; x<=5; x++) {
  console.log(x); // 1 … 5
} // end of block scope
console.log(x); // 100
```

```
  console.log(x); // Ref. error
  let x;
```

- var has "function scope"

```
var x = 100; // function scope
console.log(x); // 100
for (var x=1; x<=5; x++) {
  console.log(x); // 1 … 5
} // same x inside the loop
console.log(x); // 6
```

```
  console.log(x); // undefined
  var x;
```

Global Variables

- In the Web browser

```
x = 5;
console.log(x); // 5
console.log(window.x); // 5
```

- Server side (in Node.js)

```
x = 5;
console.log(x); // 5
console.log(global.x); // 5
```

- Strict mode forbids using global variables

```
'use strict';
x = 10; // Reference error
window.x = 5;
console.log(x);
```

```
'use strict';
window.x = 5;
x = 10; // This will work
console.log(x); // 10
```

Constant Declarations

SoftUni Foundation

- JS supports **constants** – declared by the **const** keyword
  - Once assigned, constants cannot be modified

```
const path = '/images';

console.log(`${path}/logo.png`);
// Logo path: /images/logo.png

path = '/img/new';
// TypeError: Assignment to constant variable
```

Using the JS Console

The "console" Object

The "console" Object in JS

SoftUni Foundation

```
console.log("The date is: " + new Date());
```

```
> console.log("The date is: " + new Date());
  The date is: Fri Sep 16 2016 13:25:59 GMT+0300 (FLE Daylight Time)
```

Always appends a trailing new line: \n

```
console.error("Cannot load comments");
```

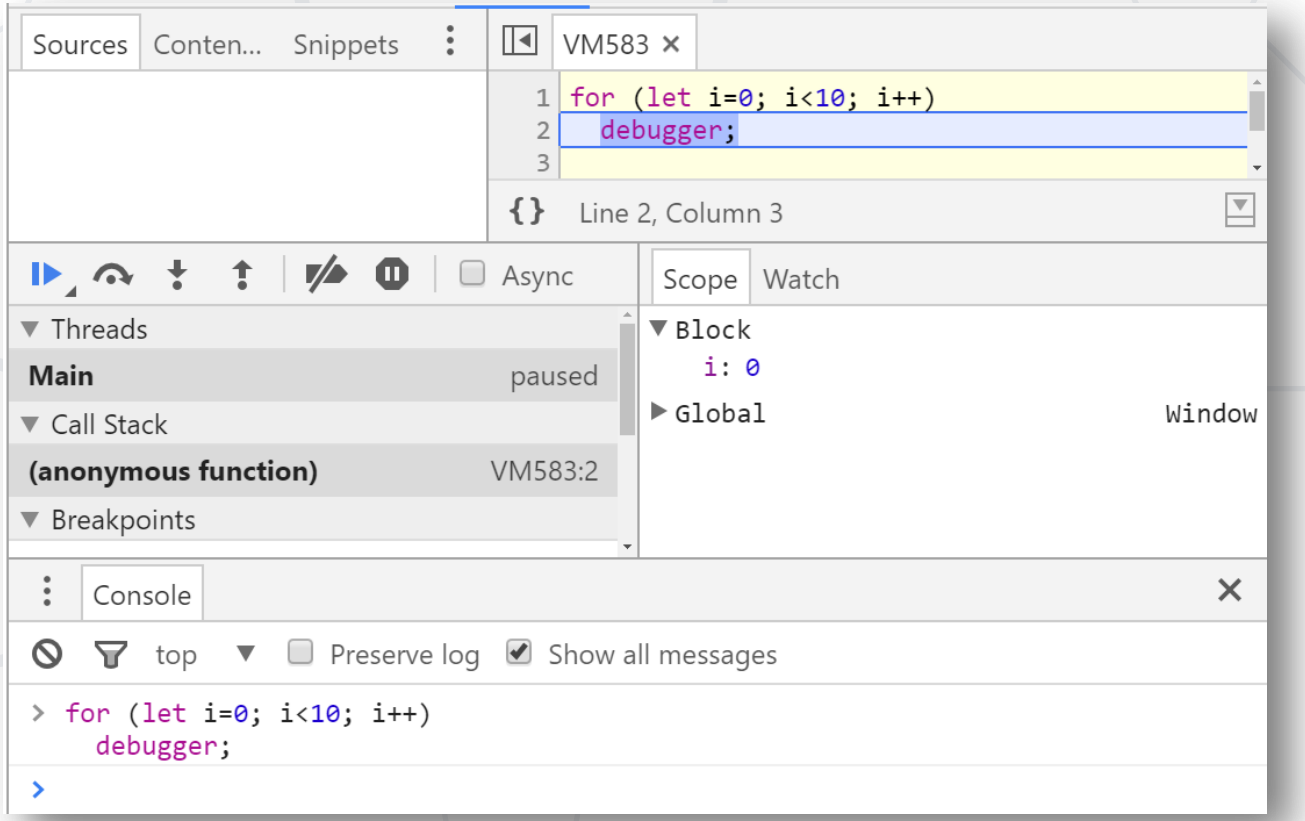
```
> console.error("Cannot load comments");
✕ ▶ Cannot load comments
```

```
console.warn("Article has no cover image");
```

```
> console.warn("Article has no cover image");
⚠ Article has no cover image
```

Invoking the JS Debugger

```
for (let i=0; i<10; i++) {
  debugger;
}
```



Problem: Next Day

- Print the next day by given **year, month, day**
  - Example: 2016, 9, 30 → 2016-10-1

```
function calcNextDay(year, month, day)  {
  let date = new Date(year, month-1, day);
  let oneDay = 24 * 60 * 60 * 1000; // milliseconds in 1 day
  let nextDate = new Date(date.getTime() + oneDay);
  return nextDate.getFullYear() + "-" +
    (nextDate.getMonth() + 1) + '-' +
    nextDate.getDate();
}
```

```
calcNextDay(2016, 9, 30)
```

Check your solution here: https://judge.softuni.bg/Contests/287

Problem: Distance Between Points

- Return the distance between two points
  - Sample solution:

```
function distanceBetweenPoints(x1, y1, x2, y2)  {
  let pointA = {x:x1, y:y1};
  let pointB = {x:x2, y:y2};

  let distanceX = Math.pow(pointA.x - pointB.x, 2);
  let distanceY = Math.pow(pointA.y - pointB.y, 2);
  return Math.sqrt(distanceX + distanceY);
}
```

Use the Pythagorean theorem

```
distanceBetweenPoints(2, 4, 5, 0)
```

Check your solution here: https://judge.softuni.bg/Contests/287

Live Exercises

Summary

- We learned what compilers, interpreters and VM's are
- We made our first program in WebStorm
- We learned basic JS syntax
- We learned how to mix HTML with JS
- We learned about data types in JS

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

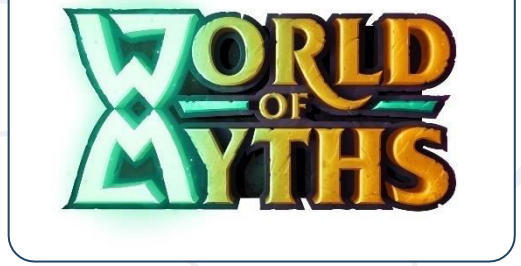
SoftUni Kids

SoftUni Diamond Partners

SoftUni Foundation

SoftUni Organizational Partners

SoftUni Foundation



Trainings @ Software University (SoftUni)

SoftUni Foundation

- 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