Programming Languages

Very Brief Introduction to the World of Programming

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://about.softuni.bg

Have a Question?

Software University

sli.do

#qa-fund

Table of Contents

What is "Coding"?

Programming Code, Commands, Programs

What is "Programming" (Coding)?

Software University

- Give **commands** to the computer

```
console.log(3+5)
```

```
console.log("Hello")
```

```
x = 5
```

```
console.log(x * x)
```

- Commands are arranged one after another into a **computer program** / **source code**

```
leva = prompt("Enter amount in BGN:")
euro = leva / 1.95583
console.log("Euro: ", euro)
```

Programming Languages

Machine, Assembler, C, C++, Java, Python, JS, …

Definition of Programming Language

Software University

- **Programming language** - formal language that is used to write instructions or programs that can be executed by a computer
- It consists of a set of rules and symbols that allow programmers to write code that is easily understood by both humans and machines
- Programming languages can be classified into different categories based on their level of abstraction, syntax, and execution model
- They can be **low-level** or **high-level**, **scripting** or **compiled**, procedural or object-oriented and so on

Definition of Programming Language

Software University

- **Programming language**: a formal language (syntax)
  - Used to write **instructions** (commands or programs) that can be **executed by a computer**
  - A **set of rules** (syntax and command format) used to construct computer programs (programming code)
- Different **types** of programming languages:
  - **Low-level** / **high-level**, **scripting** / **compiled**, **statically-typed** / **dynamic**, **procedural** / **object-oriented** / **functional**, etc.

History: Machine & Assembly Language

Software University

- **Machine language** (1st generation, 1940s)
  - The first programming language, used in earliest computers
  - **Binary code**, directly executed by the CPU (central processing unit)
- **Assembly language** (2nd generation, 1950s)
  - Simplify coding in machine language
  - Uses **mnemonics** to represent **machine instructions**, easier to understand and learn

machine code

```
10110100 10110111 00101011
00011010 00010100 10111011
10001000 11110111 00101000
10101010 00101101 00010001
01010010 11101100 11010001
10010100 10010100 00100000
00000000 10100001 00110001
10101001 00010101 00101010
00100100 10010100 01110001
11110101 11101011 00101111
01010010 10000101 11111110
```

```
.MODEL SMALL
.STACK 100H
.CODE

MOV AX, 0x3C
MOV BX, 0000000000001010B
ADD AX, BX
MOV BX, 14
SUB AX, BX

MOV AH, 04CH
INT 21H
```

Ancient Languages: Fortran, COBOL, BASIC

- **Fortran** (1957) – the first high-level language
  - For scientific and engineering computations
  - Easier to read and write than assembler
- **COBOL** (1959) – developed for business applications (and is still in use today)
  - Easily to understand by analysts and managers
- **BASIC** (1964) – developed for educational purposes, easy to learn
  - Still used today for simple coding tasks

Modern Languages: C, C++, Python, Java, JS

- **C** (1972) – powerful and efficient mid-level language for system programming (e. g. the Linux kernel is written in C)
- **C++** (1983) – complex, object-oriented, highly-efficient language for system and high-performance apps and games
- **Python** (1991) – high-level, simple scripting language for scientists, easy to learn, for Web apps, data science and AI
- **Java** (1995) – popular high-level, cross-platform, object-oriented language for Web, mobile and business apps
- **JavaScript** (1995) – simple, dynamic scripting language for Web, runs in the Web browsers, also server-side

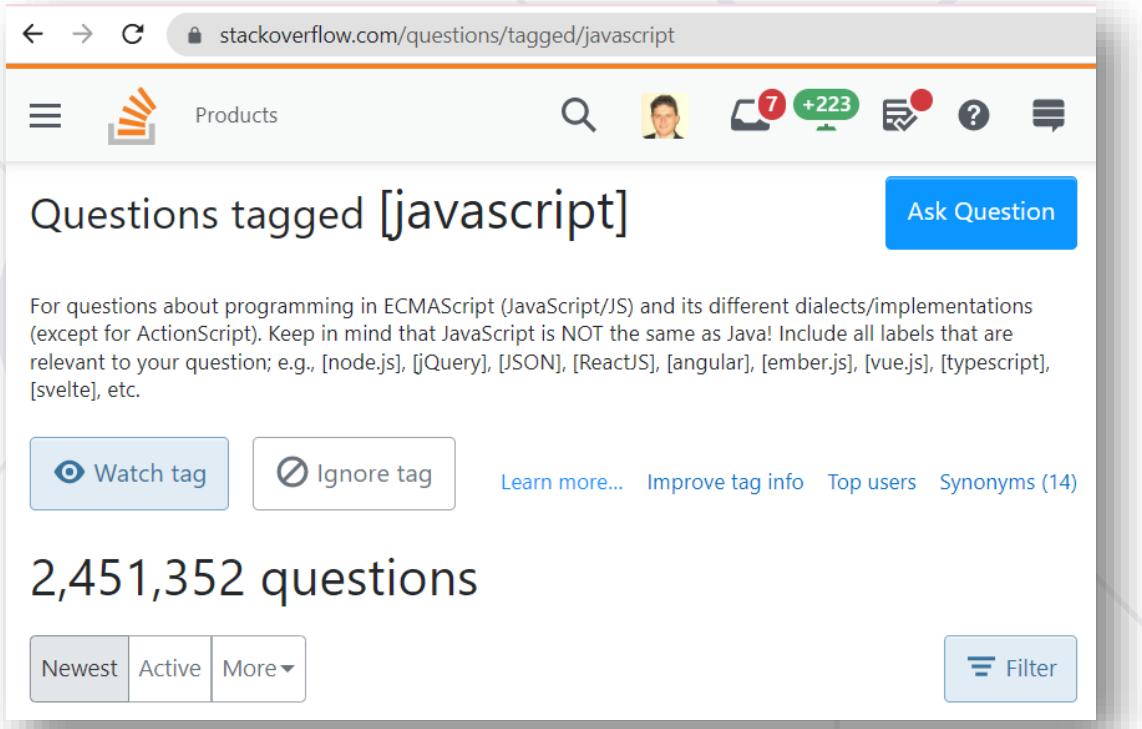
Modern Languages: C#, PHP, Go, TypeScript…

- **PHP** (1995) – scripting language for Web sites and server-side Web development
- **C#** (2000) – modern object-oriented language for universal use: business apps, Web apps, mobile apps
- **Go** (2009) – simple, efficient lang for high-performance apps
- **TypeScript** (2012) – strongly-typed JS, less prone to errors
- **Swift** (2014) – general purpose lang, for the Apple ecosystem
- Other modern mainstream languages:
  - **Kotlin**, **Scala**, **Rust**, **Ruby**, **Dart**, **Perl**

Most Popular Programming Languages

Software University

- **Stack Overflow** all time questions asked **by tag** (Nov 2022)
  https://stackoverflow.com/questions/tagged/java
  - **JavaScript** ⟶ 2.45M questions
  - **Java** ⟶ 1.88M questions
  - **Python** ⟶ 2.07M questions
  - **C#** ⟶ 1.57M questions
  - **PHP** ⟶ 1.45M questions
  - **C++** ⟶ 0.78M questions



Non-Programming Tech Languages

Software University

- Many tech languages are **not real programming languages!**
- **SQL**: database query and manipulation language
  - **PL/SQL**, **Transact SQL**, etc. – more powerful, still DB specialized
- **HTML** and **CSS**: visualize Web content (text + images + links)
- **XML**, **JSON** and **YAML**: represent, store and transport data
- **Bash** / **PowerShell**: system administration scripting tools
- **HCL**: describe and configure virtual infrastructure (IaC)

Low-Level & High-Level Languages

Assembler vs. Modern Languages

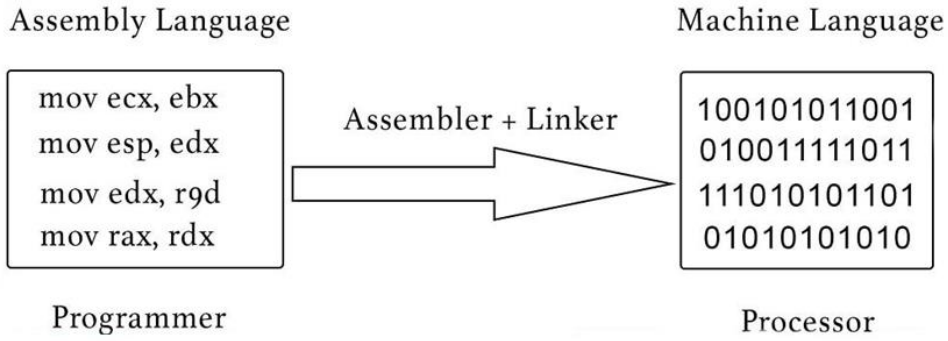
Low-level and high-level languages

Software University

- Programming languages may differ in terms of **level of abstraction** and their **relationship to the hardware**
- **Low-level languages** runs **closer to the hardware** and computer's architecture
- **High-level languages** are more **abstract** and easier
  - **Easy to use**, read, write and understand than low-level languages
  - Developers **write less code** in higher-level languages (e. g. Python) than in lower-level languages (e. g. C)

Low-Level Languages

- **Low-level languages** == **machine code** and **assembly languages**
  - Write code for **direct execution the computer's CPU**
- Difficult to read and write
- Rarely used today
  - **Modern compilers** are better than humans in machine code
- Example: Assembler



Machine Language

ADD contents of 2 registers, store result in third.	1010000100 RR RR RR ex: R0 = R1 + R2 1010000100 00 01 10
SUBTRACT contents of 2 registers, store result into third	1010001000 RR RR RR ex: R0 = R1 – R2 1010001000 00 01 10
Halt the program	1111111111111111

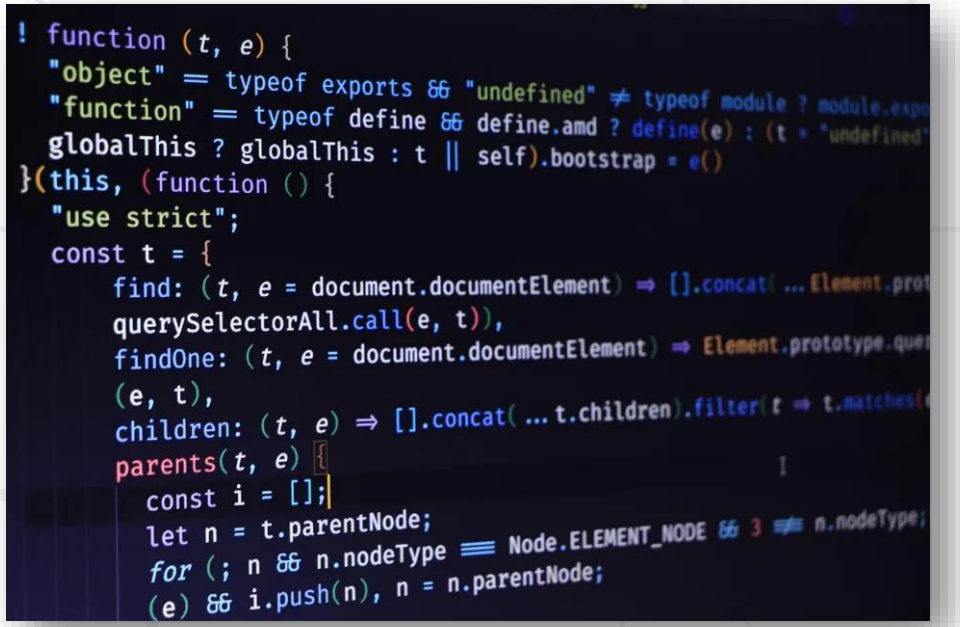
Characteristics of Low-Level Languages

- Specific to a **particular CPU architecture** and machine
- Direct control over the **hardware** and **memory**
- **Difficult** to read, write and maintain
- Used in developing operating systems, device drivers, and firmware
- Examples of low-level languages include Assembly language and machine code

High-Level Languages

Software University

- High-level languages – **closer to the natural language**
- Often use English-like keywords and syntax to make programming **more intuitive** and easier to learn
- High-level languages may be **compiled** into low-level code
- Or can be execute line by line by a language **interpreter**



Characteristics of High-Level Languages

- More **abstract** and use natural **language-like syntax**
  - Require **less code** to be written
  - **Easier to learn** and use than low-level languages
  - **Independent** of the CPU architecture and machine
- Used in developing Web and Mobile apps, scientific simulations, and enterprise software
- Examples of high-level languages:
  - Java, Python, C#, and JavaScript, PHP

Scripting vs. Compiled Languages

Interpreters vs. Compilers

Scripting Languages

Software University

- A **scripting** language is designed to be **interpreted**
  - Executed command-by-command by an interpreter
  - Typically, slower than compiled languages
  - Support the **REPL** model: Read-Evaluate-Print Loop
  - Dynamic, flexible in terms of syntax and error handling
  - Used for automating tasks, such as system administration, web development, and data analysis
  - Examples: **Python**, **JavaScript**, **Perl**, **VB.NET**

Compiled Languages

- A **compiled** language is **compiled** into machine code, which can be directly executed by a computer's CPU
- **More efficient** in terms of **memory usage** and **execution speed** than scripting languages
- Used for developing large-scale applications, such as operating systems, games, and enterprise software
- Less dynamic, strict typing system, strict syntax, more complex than scripting languages
- Examples: **C++**, **Java**, **C#**, **Go**

Language Execution Model

- **Compiled** languages
  - Source code is first **compiled** to machine code, then executed
  - Syntax errors are found during the **compilation** (at compile time)
  - Examples: **C#**, **Java**, **C**, **C++**, **Swift**, **Go**, **Rust**

- **Interpreted** languages
  - Each command is read, parsed and executed by an **interpreter**
  - Syntax errors are found at **run-time**, during execution
  - Examples: **Python**, **JavaScript**, **PHP**, **Perl**, **Ruby**

Most Popular Languages

C#, Java, JavaScript, Python

C# Programming Language

- **C#** is modern, flexible, general-purpose programming language
- **Object-oriented** by nature, statically-typed, compiled
- Runs on .NET Framework / .NET Core

```
static void Main()
{
   Console.WriteLine(3+5);
}
```

Program starting point

C#: Declaring Variables

- Defining and initializing variables

```
{data type / var} {variable name} = {value};
```

- Example

```
int number = 5;
```

Variable name

Variable value

Data type

```
Console.WriteLine(number);
```

Printing a variable

Java – Introduction

- **Java** is modern, flexible, general-purpose programming language
- **Object-oriented** by nature, statically-typed, compiled

```
static void main(String[] args) {
    int a = 5;
    System.out.println(a * a);
}
```

Program starting point

Java – Declaring Variables

- Defining and Initializing variables

```
{data type / var} {variable name} = {value};
```

- Example

```
int number = 5;
```

Variable name

Variable value

Data type

```
System.out.println(number);
```

Printing a variable

Python

- Popular, easy to learn, scripting language
- A good start for beginners
- The syntax is close to plain **English language**
- **Interpreted** language
- Dynamic typing
- Supports object-oriented programming
- Cross-platform, tons of libraries

Python – Basic Syntax

- **Variables** – they are way to **store information** and are characterized by **name**, **type** and **value**

```
age = 25
```

Name

Value

- Printing data:

```
print("age = ", age)
```

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 Syntax

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

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

Variable name

Declare a variable with let

Variable value

- Conditional statement:

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

Body of the conditional statement

Functions and Input Parameters

- **Functions** are named pieces of code
  - Can take **input data** (**parameters**) and return **output data** (**result**)
- A function can be **invoked** (called) to execute its body

Declaration

Parameters

```
function calcSum(num1, num2) {
  return num1 + num2;
}

let sum = calcSum(2, 3);
console.log("sum = " + sum);
```

Call the function

Printing to the Console

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

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

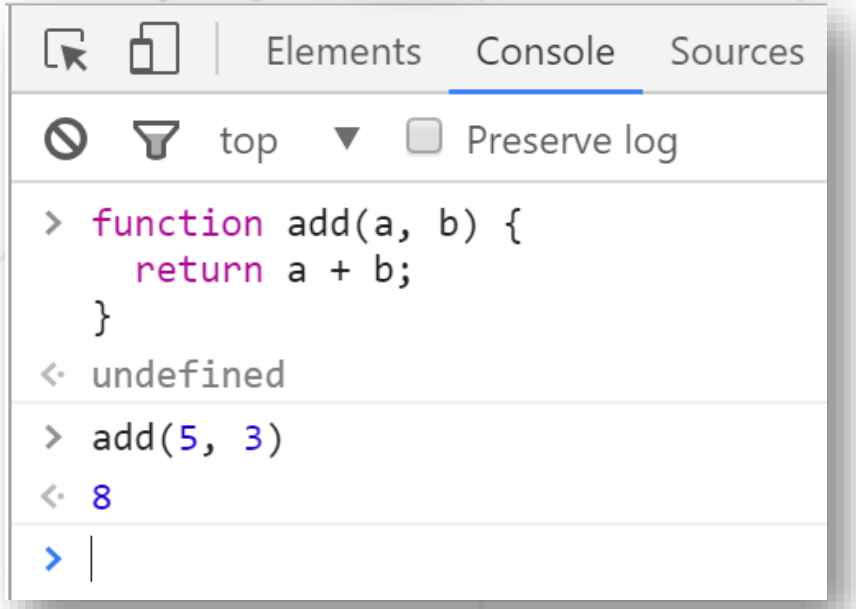
- To format a number, use the **toFixed()** method (converts to **string**):

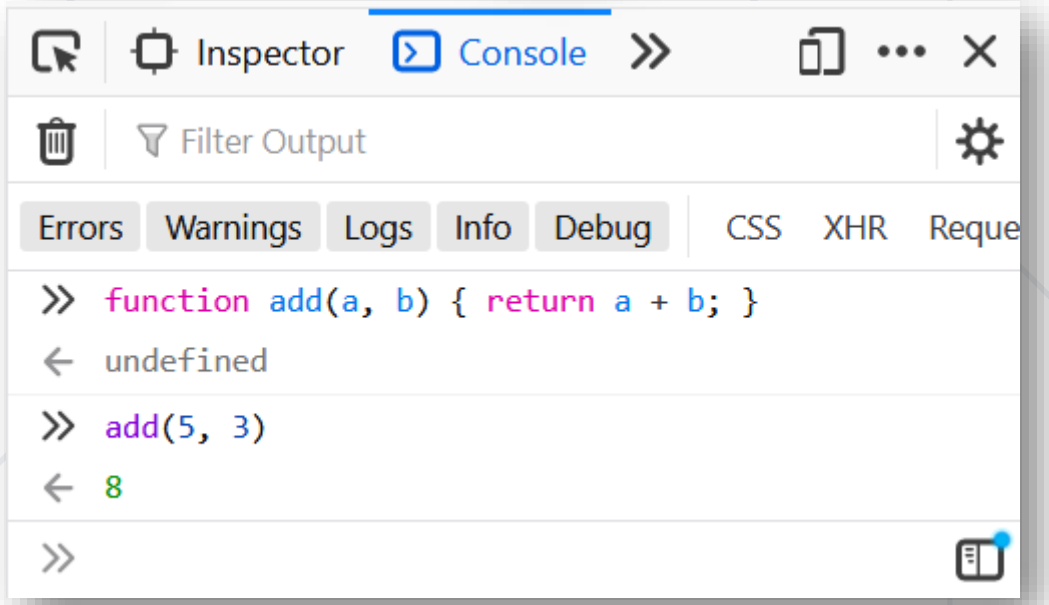
```
grade.toFixed(2);  //The name is: Petar, grade: 3.56
```

JS in Chrome Web Browser / Firefox

Software University

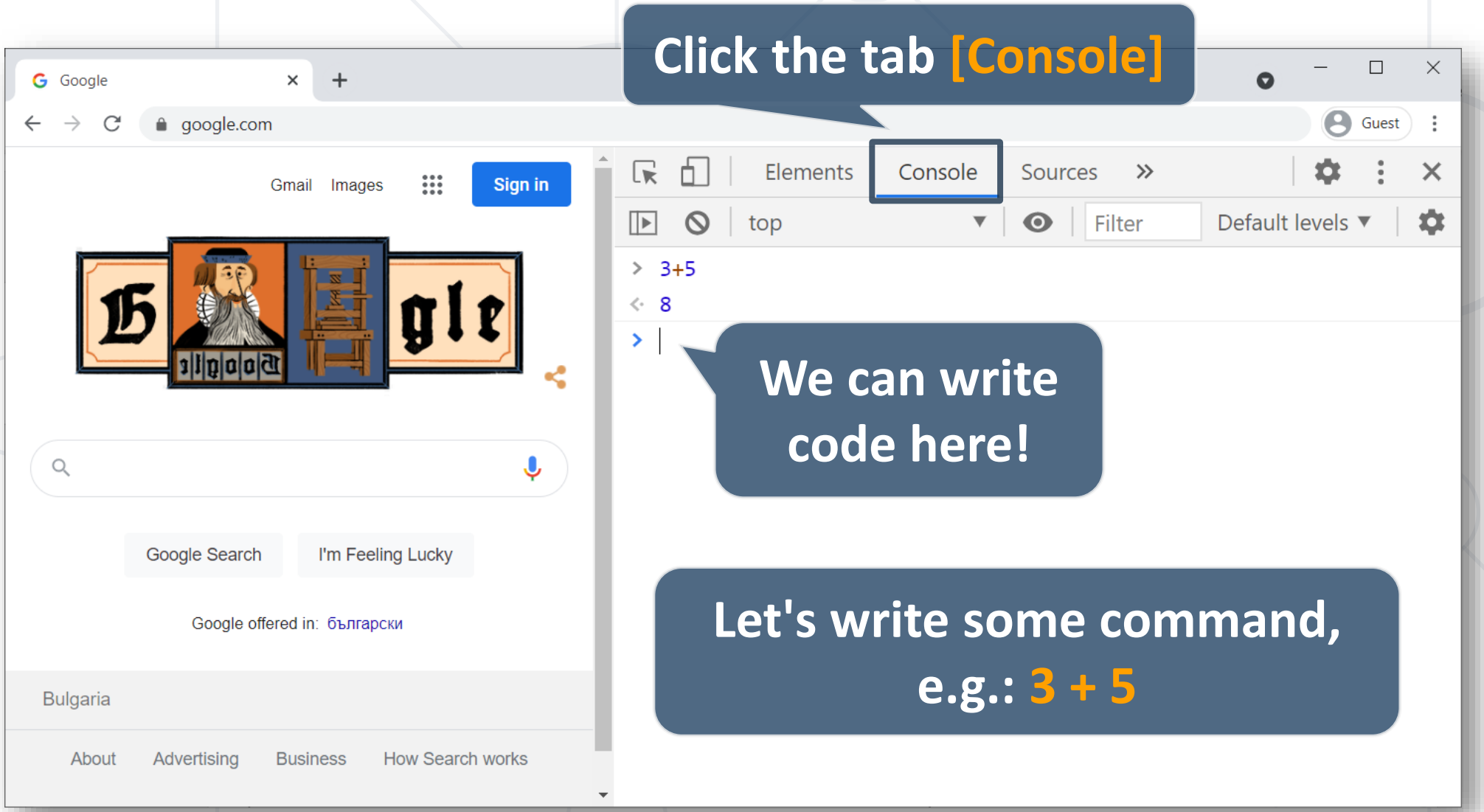
- Developer Console: **[F12]**





Writing JavaScript Code in the Browser

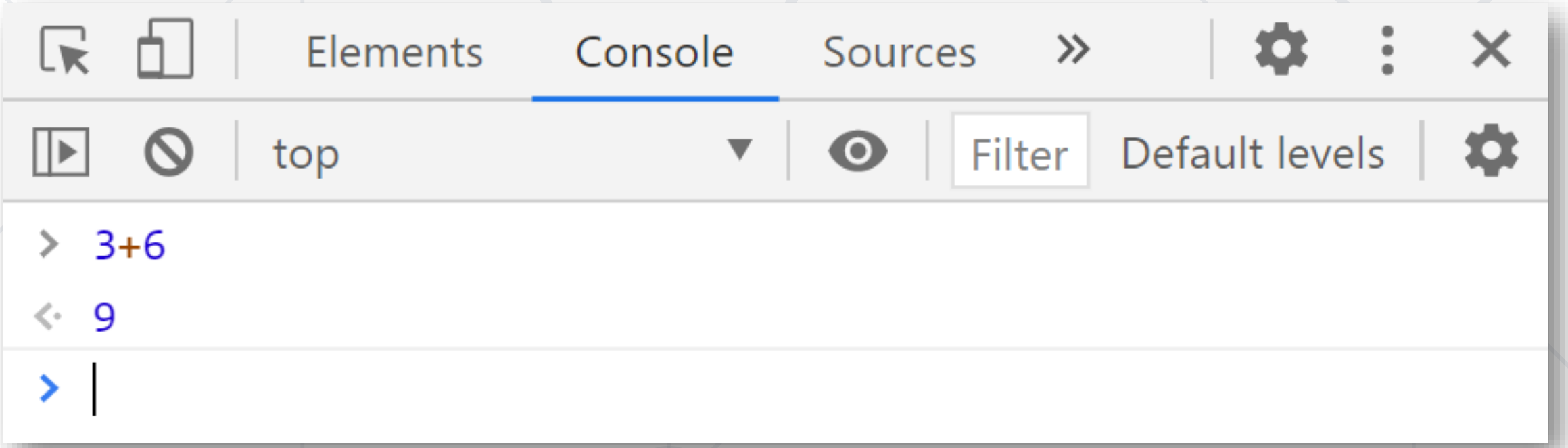
Software University



Problem: Calculate 3 + 6

- Calculate the expression **3+6**

```
3 + 6
```

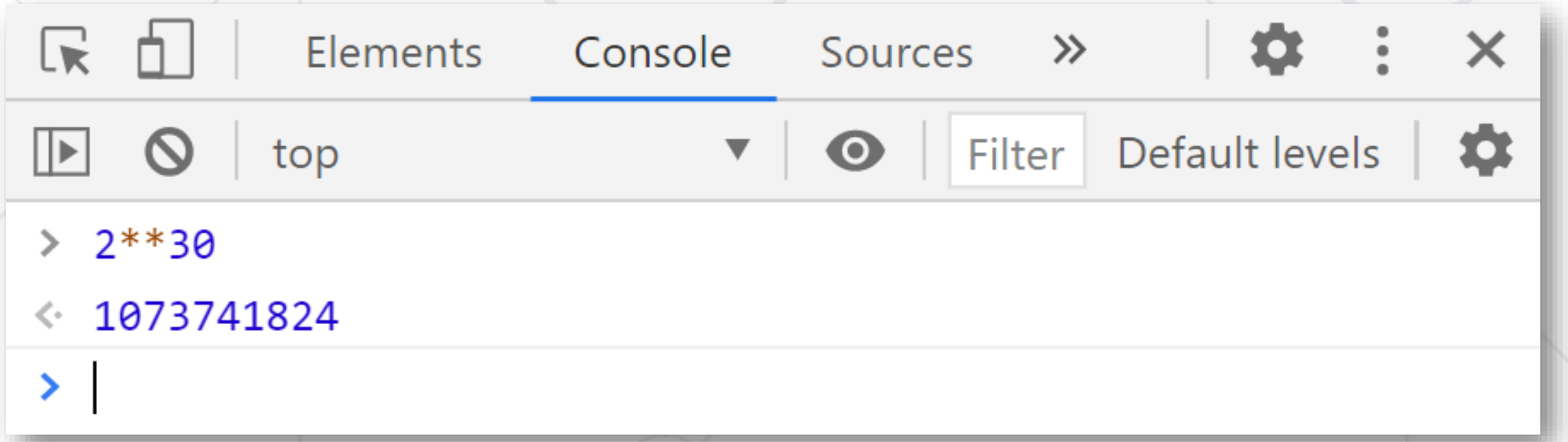


Problem: 2^{30}

- Calculate 2^{30} (2*2*….*2 multiplied 30 times)

```
2 ** 30
```

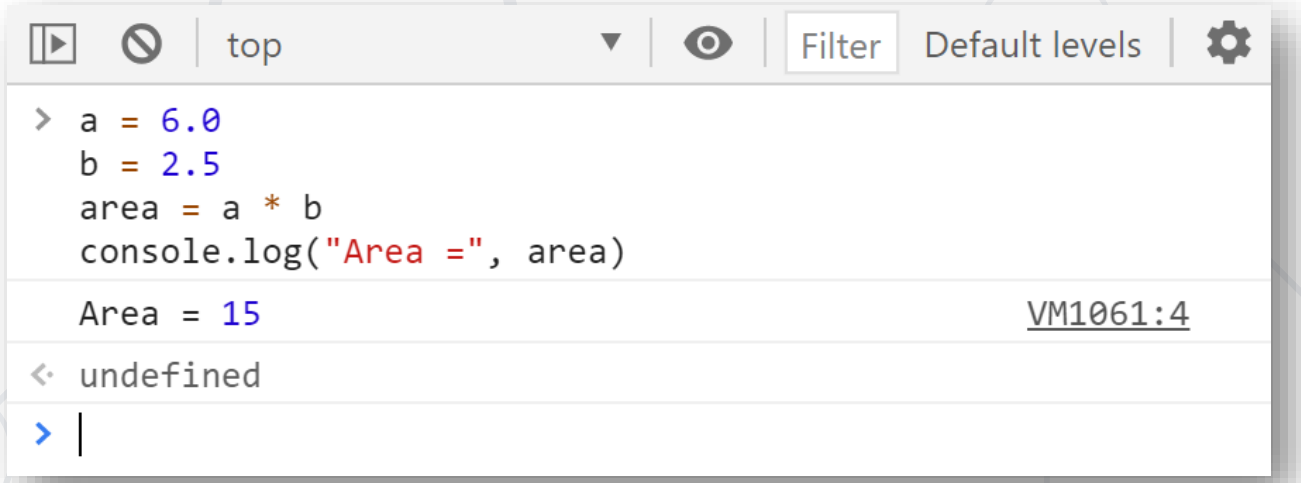
Look in Internet
"JavaScript exponentiation"



Problem: Area of Rectangle

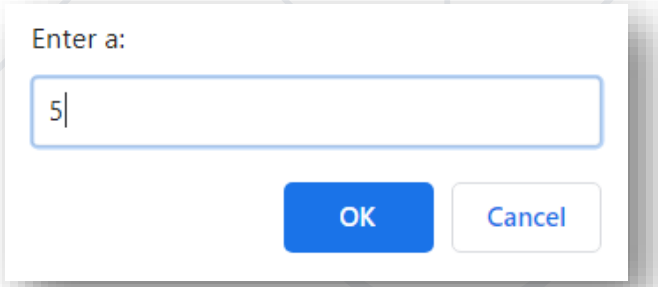
- Calculate the **area of a rectangle** with sides **6.0** and **2.5**

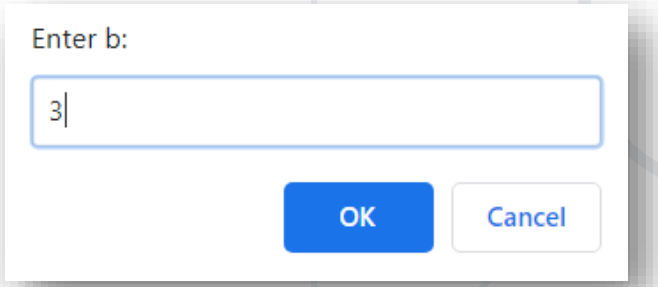
```
a = 6.0
b = 2.5
area = a * b
console.log("Area =", area)
```

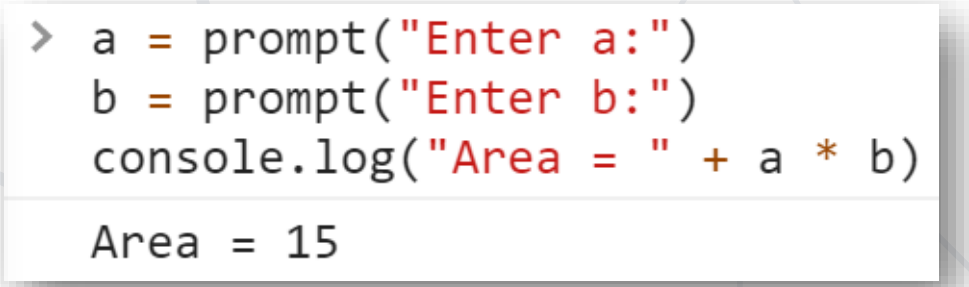


Reading Input Data

```
a = prompt("Enter a:")
b = prompt("Enter b:")
console.log("Area = " + a * b)
```





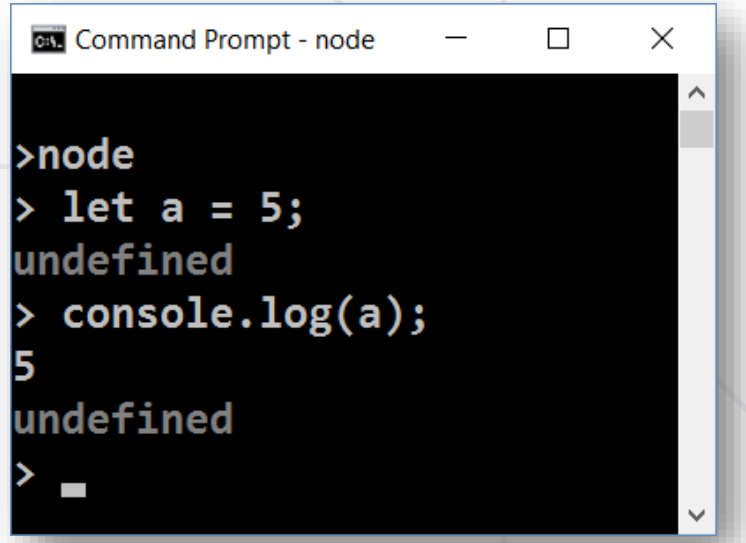


Setting up Node.js + VS Code

Running JavaScript in VS Code

Node.js

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



Install the Latest Node.js

Software University

Downloads

Latest LTS Version: **18.16.0** (includes npm 9.5.1)

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-v18.16.0-x64.msi

macOS Installer

node-v18.16.0.pkg

Source Code

node-v18.16.0.tar.gz

Windows Installer (.msi)	32-bit	64-bit
Windows Binary (.zip)	32-bit	64-bit
macOS Installer (.pkg)	64-bit / ARM64	
macOS Binary (.tar.gz)	64-bit	ARM64
Linux Binaries (x64)	64-bit	
Linux Binaries (ARM)	ARMv7	ARMv8
Source Code	node-v18.16.0.tar.gz	

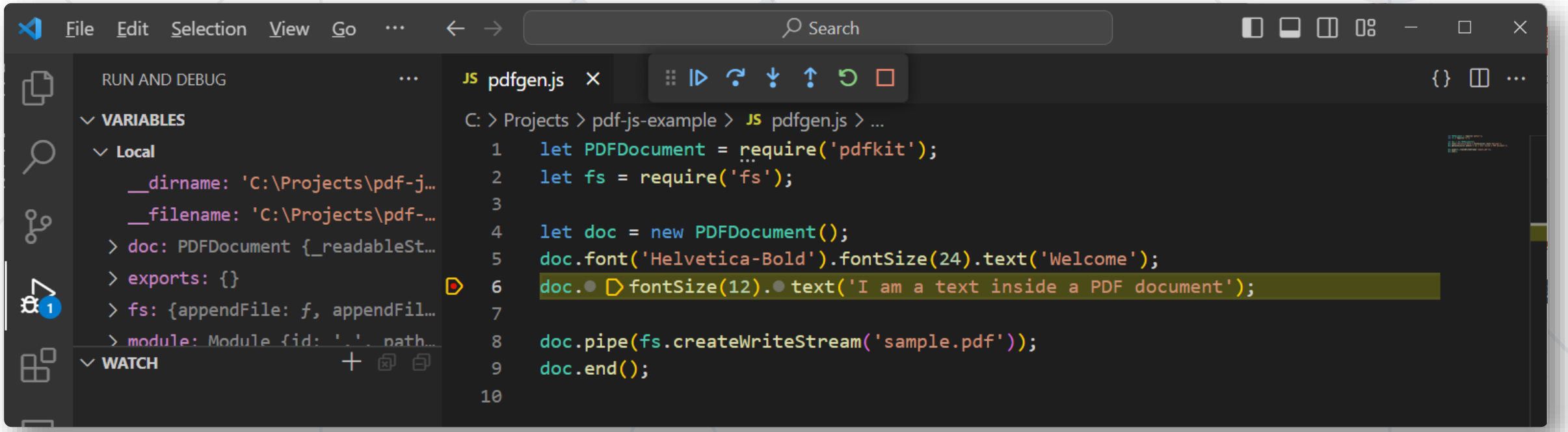
What is an IDE?

- **IDE** == Integrated Development Environment
  - Write code, run the code, debug the code
  - Syntax highlighting, auto-complete, plugins
- Examples:
  - **Visual Studio** – IDE for C# and .NET
  - **PyCharm** – IDE for Python
  - **IntelliJ IDEA** – IDE for Java
  - **Web Storm** – IDE for HTML, CSS and JS

What is an IDE? What is VS Code?

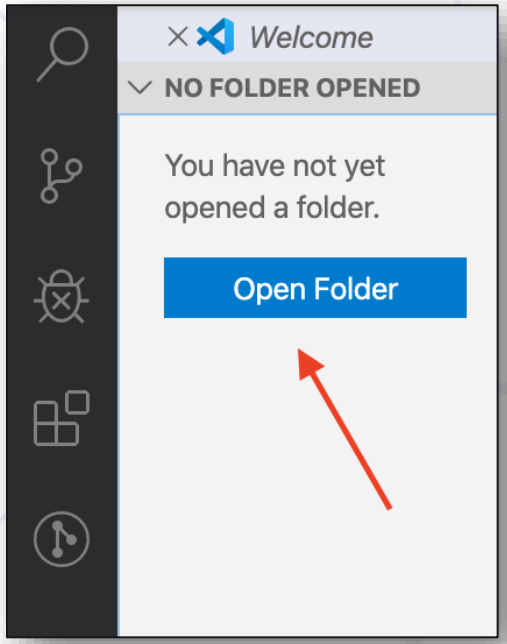
Software University

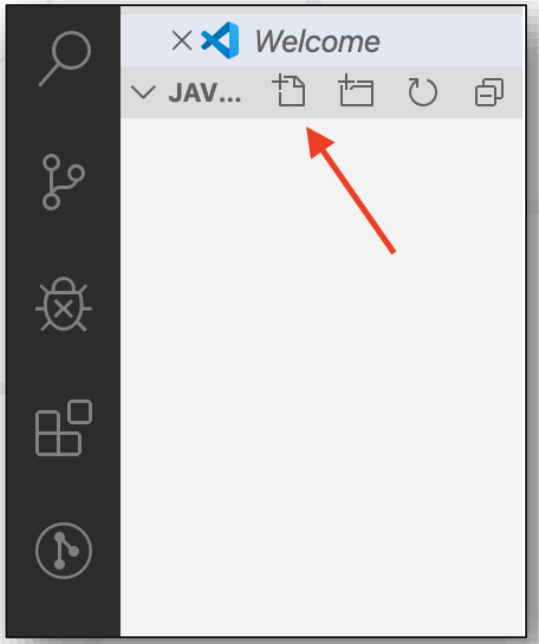
- **VS Code** (Visual Studio Code)
  - Universal IDE for JS, Python, Java, HTML, CSS, C# and others
  - https://code.visualstudio.com

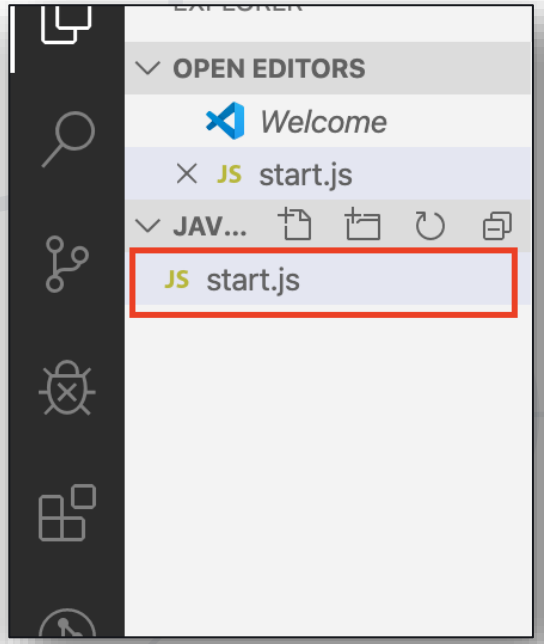


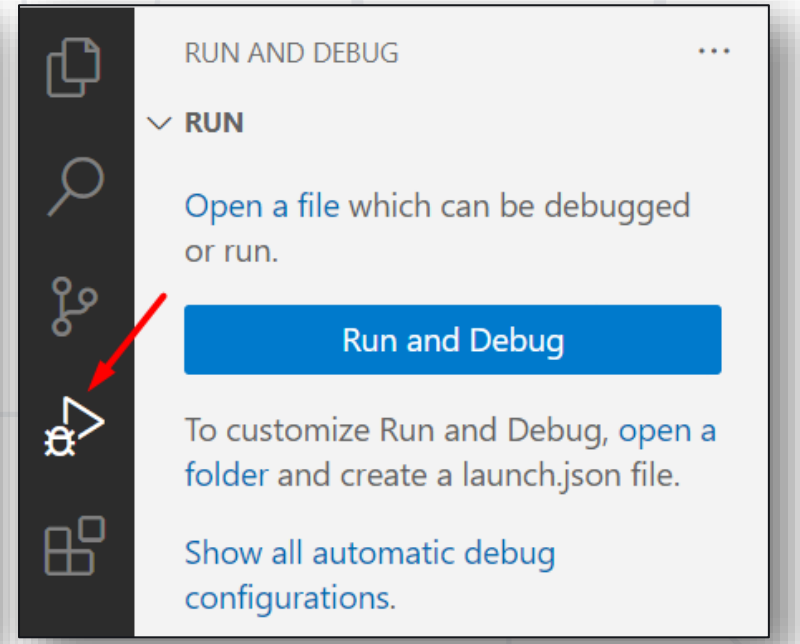
Using Visual Studio Code

- **Visual Studio Code** is powerful text editor for JavaScript and others
- Creating an running a JS **project**:









HTML & CSS & JavaScript

Writing Simple Front-End Code

What is HTML?

- The **HTML** language describes Web content (Web pages)
  - Text with formatting, images, lists, hyperlinks, tables, forms, etc.
  - Uses **tags** to define **elements** in the Web page

Opening tag

```
<p>
  <b>Document</b> content goes here…
</p>
```

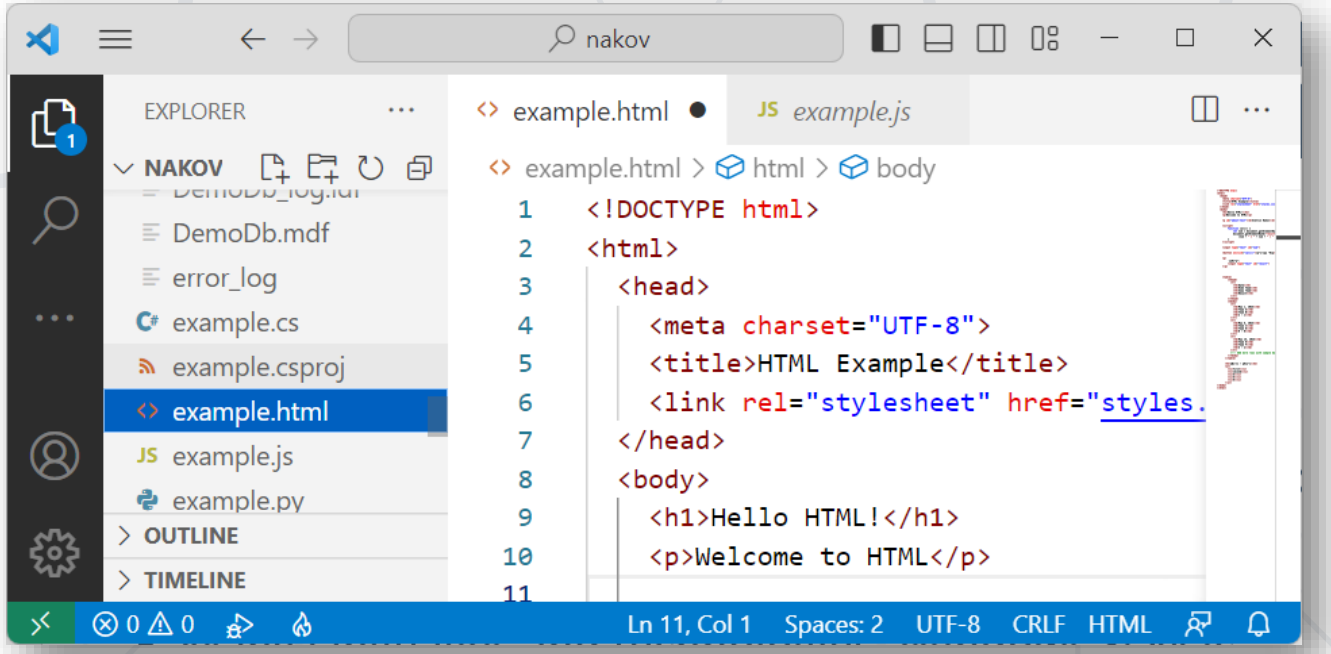
Element

Closing tag

Document content goes here…

HTML Developer Environments

- **Visual Studio Code**, **Brackets**, **NetBeans**
  - Good free tools for HTML5, cross-platform
- **WebStorm**
  - Powerful IDE for HTML, CSS and JavaScript
  - Paid product
- **Visual Studio**
  - Many languages and technologies, Windows & Mac



HTML Page – Example

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="UTF-8">
    <title>HTML Example</title>
  </head>
  <body>
    <h1>Hello HTML!</h1>
  </body>
</html>
```

Problem: Welcome to HTML

- Create your first HTML page
  - File name: **welcome.html**
  - Title: Welcome
  - Paragraph of text:
    I am learning **HTML** and **CSS**!

- Hints:
  - Modify the code from the previous slide, use **** tag

What is CSS?

- **CSS** defines styling of the HTML elements
  - Specifies fonts, colors, margins, sizes, positioning, floating, …
  - CSS rules format: `selector { prop1:val1; prop2:val2; … }`
- CSS rule example:

Combining HTML and CSS Files (External Style)

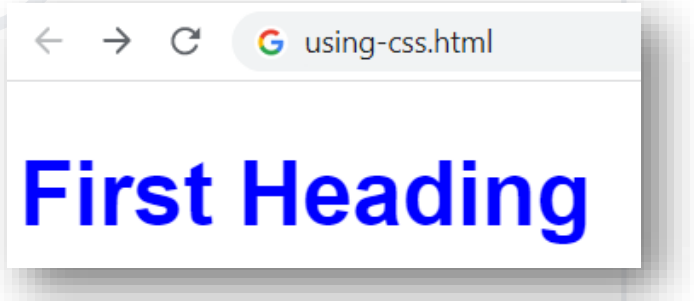
Software University

using-css.html

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet"
          href="styles.css">
  </head>
  <body> <h1>First Heading</h1>
  </body>
</html>
```

styles.css

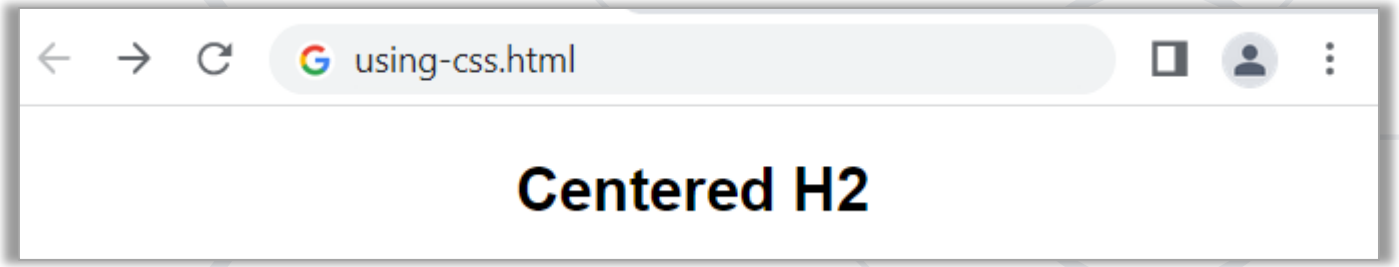
```
/* CSS here */
h1 {
  font-size: 42px;
  color: blue;
}
```



Class Selector

- Uses the HTML class attribute, and is defined with a "."

```
.centered {
  text-align: center;
}
```




```
<h2 class="centered">Centered H2</h2>
```

- Style all **paragraphs**, having class "**right**"

```
p.right {
  text-align: right;
}
```

Right-aligned paragraph of text

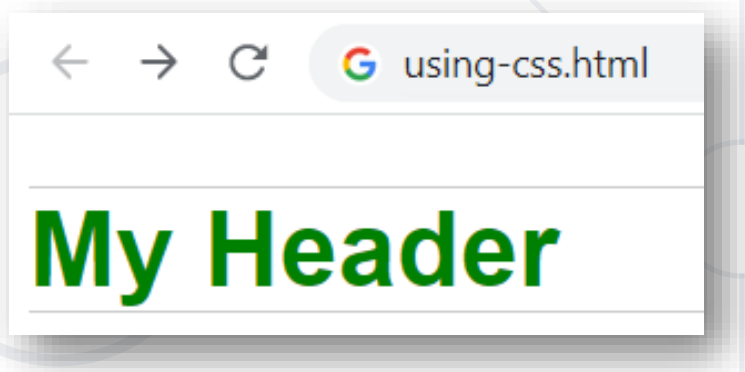
```
<p class="right">Right-aligned
paragraph of text</h2>
```

Id Selector

- The **#id** selector styles the element with the specified **id**:

```
<h1 id="header">My Header</h1>
```

```
#header {
   color: green;
   border: 1px solid #CCC;
   border-width: 1px 0;
}
```



Internal Style Sheet

- Put a `<style>` element in the HTML `<head>` section

```
<!DOCTYPE html>
<html>
<head>
  <style>
    .red {
      color:red;
    }
  </style>
</head>
</html>
```

```
<body>
  <p class="red">This is red</p>
</body>
```

Inline CSS Style

- The `style` attribute in HTML elements

Attribute "style"

```
<h1 style="color:blue">This is a blue heading</h1>
```

Property

Value

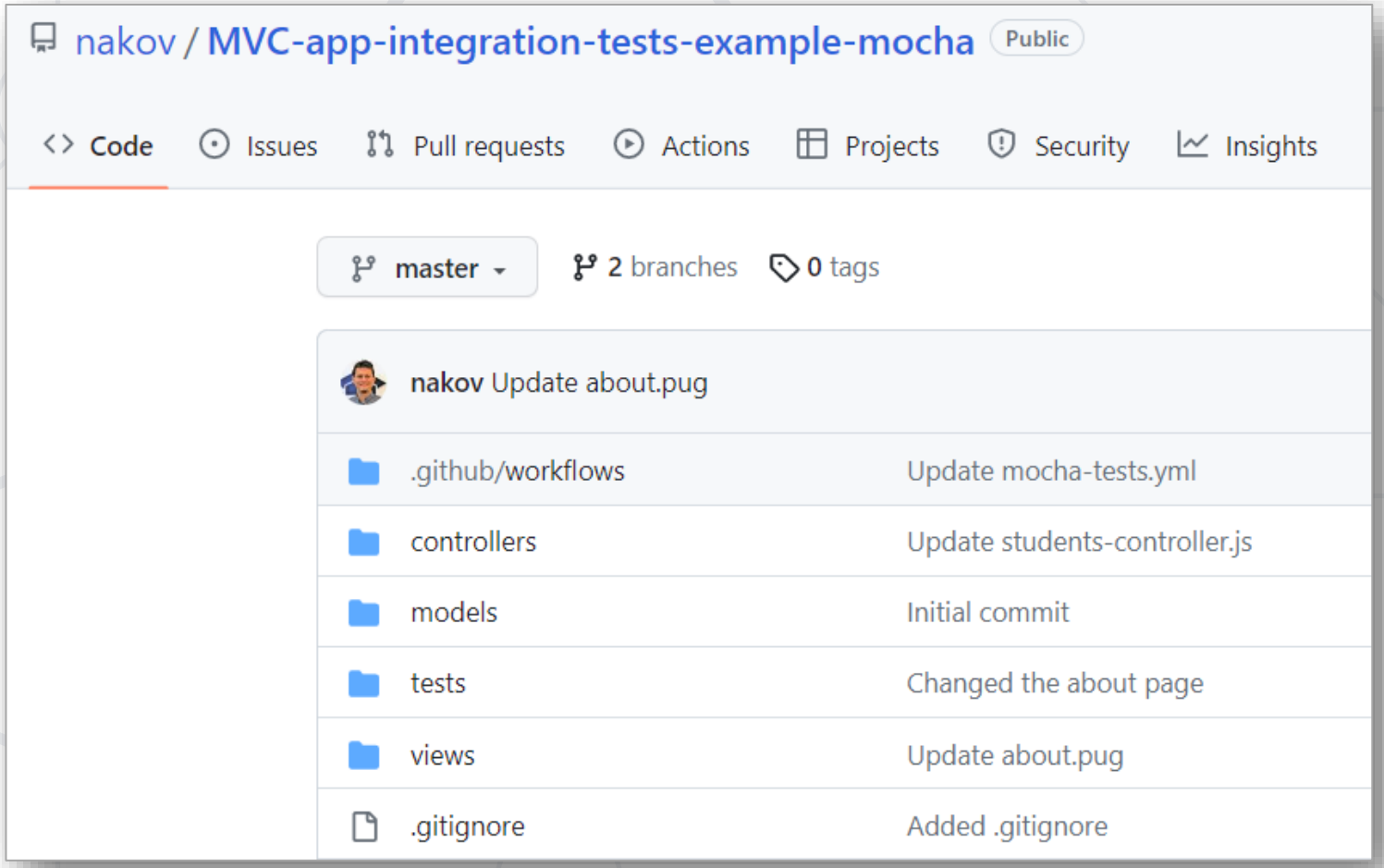
Real Software Project

Explore, Test and Run and Node.js Web App

Explore Project @ GitHub Repository

Software University

- Link to project: [nakov/MVC-app-integration-tests-example-mocha]



Running the Sample JS App

```
git clone https://github.com/nakov/MVC-app-integration-tests-example-mocha
```

```
cd MVC-app-integration-tests-example-mocha
```

```
npm install
```

```
npm test
```

```
Home page
  ✓ Page title (45ms)
  ✓ Students count (38ms)

View Students page
  ✓ Page title (40ms)
  ✓ Students list (38ms)


9 passing (1s)

[node1] (local) root@192.168.0.13 ~/MVC-app-integration-tests-example-mocha
$ npm start

> start
> node index.js

App started. Listening at http://localhost:8080
```

```
npm start
```

Summary

- Programming languages: JS, Python, Java, C#
  - Low-level & high-level languages
  - Scripting vs. compiled languages
- Most popular programming languages: JavaScript, Python, Java, C#, C++, PHP
- IDE setup: Visual Studio Code for JS coding
- HTML, CSS and JavaScript – front-end coding
- Running a real-world Node.js software project

Questions?

SoftUni

Software University

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

Finance Academy

SoftUni Diamond Partners

Software University

License

- 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

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
- Software University Forums
  - forum.softuni.bg