Programming Languages

Very Brief Introduction to the World of Programming

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

Table of Contents

sli.do

#qa-fund

What is "Coding"?

Programming Code, Commands, Programs

What is "Programming" (Coding)?

- Give **commands** to the computer

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

```
console.log("Hello")
```
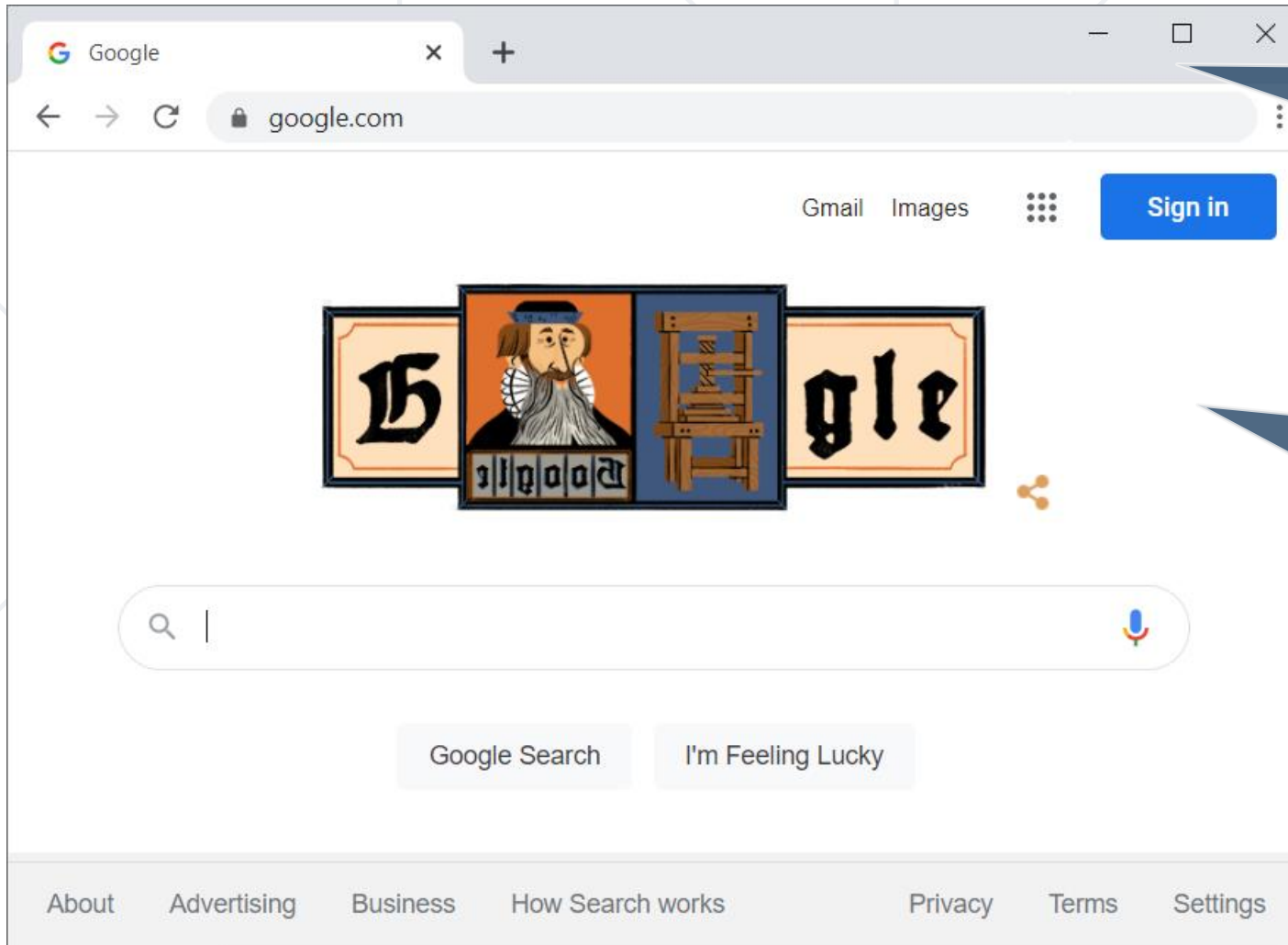
```
x = 5
```

```
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)
```
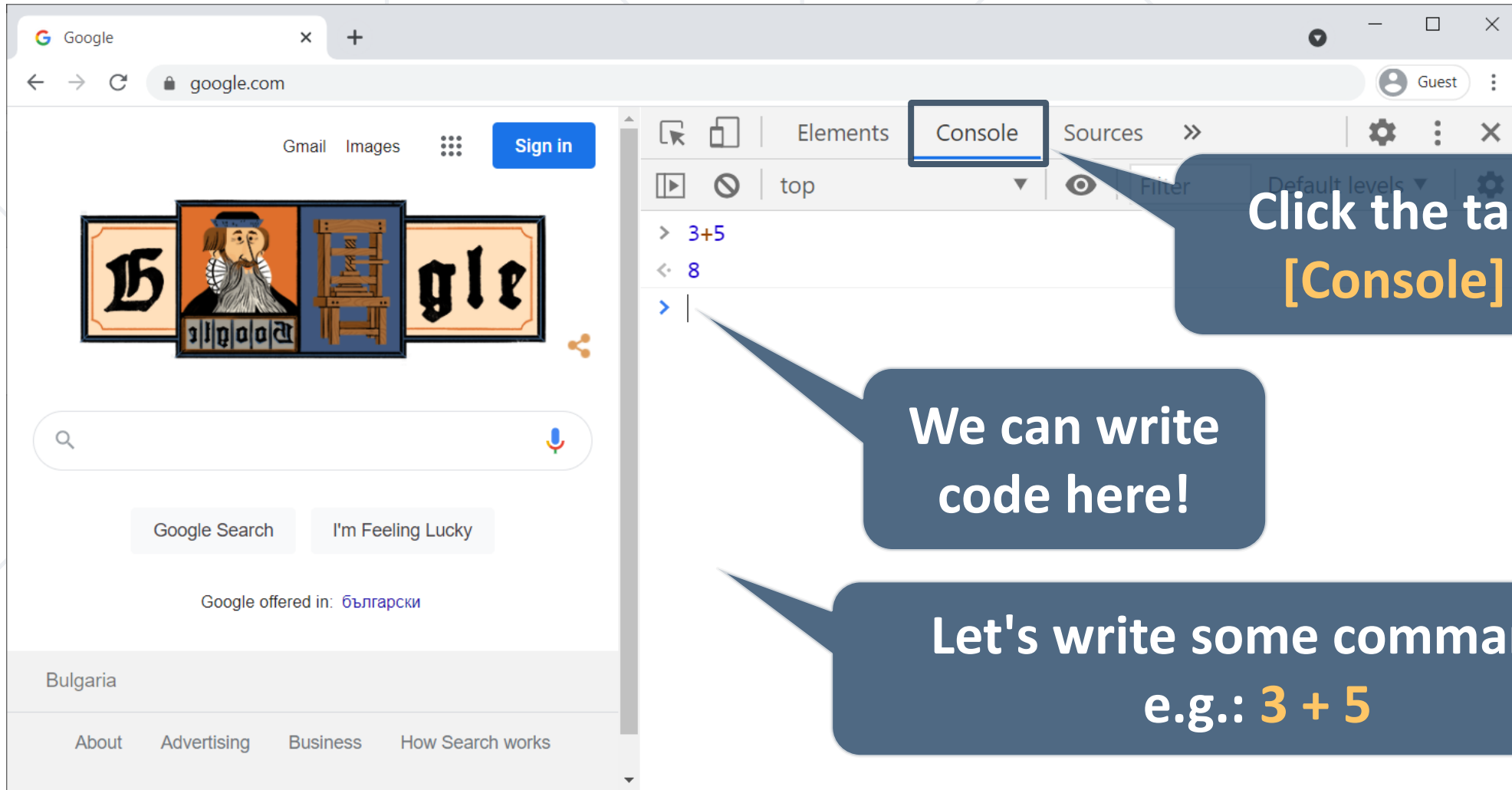
Writing JavaScript Code in the Browser

Start a Web Browser (e. g. Chrome)

Press the key [F12], to show the Dev Tools

Writing JavaScript Code in the Browser (2)
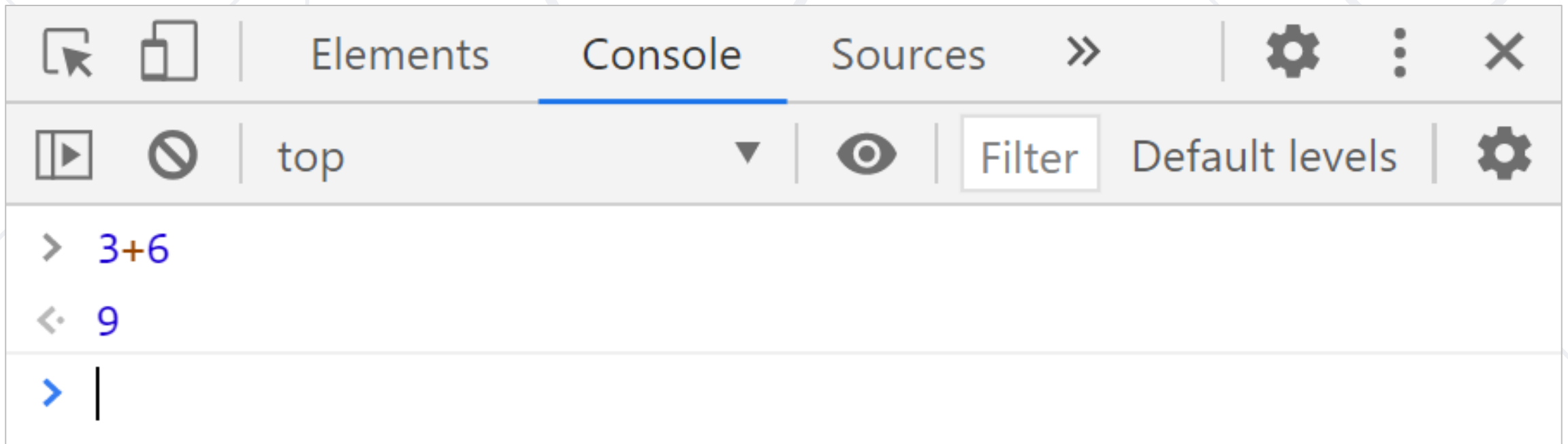

Click the tab [Console]

We can write code here!

Let's write some command, e.g.: 3 + 5

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"

Or ask ChatGPT:
Calculate 2 to the power of 30 in JS

Elements | Console | Sources | » ⚙ ⋮ ✕

▶ 🚫 | top ▼ | 👁 | Filter Default levels | ⚙

```
> 2**30
< 1073741824
>
```
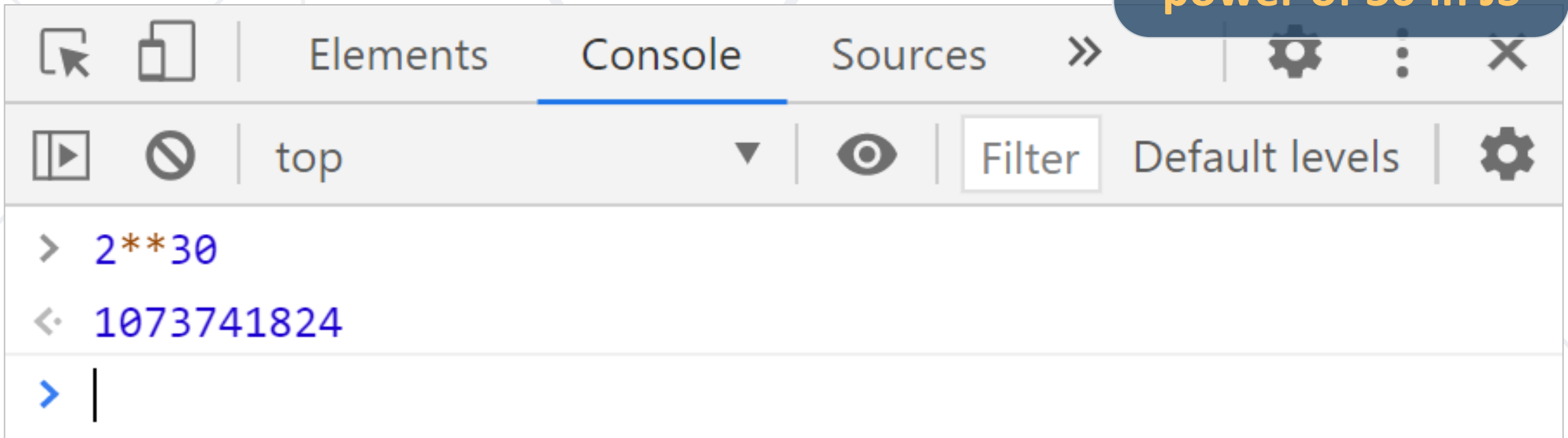
Problem: Area of Rectangle

■ 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)
```
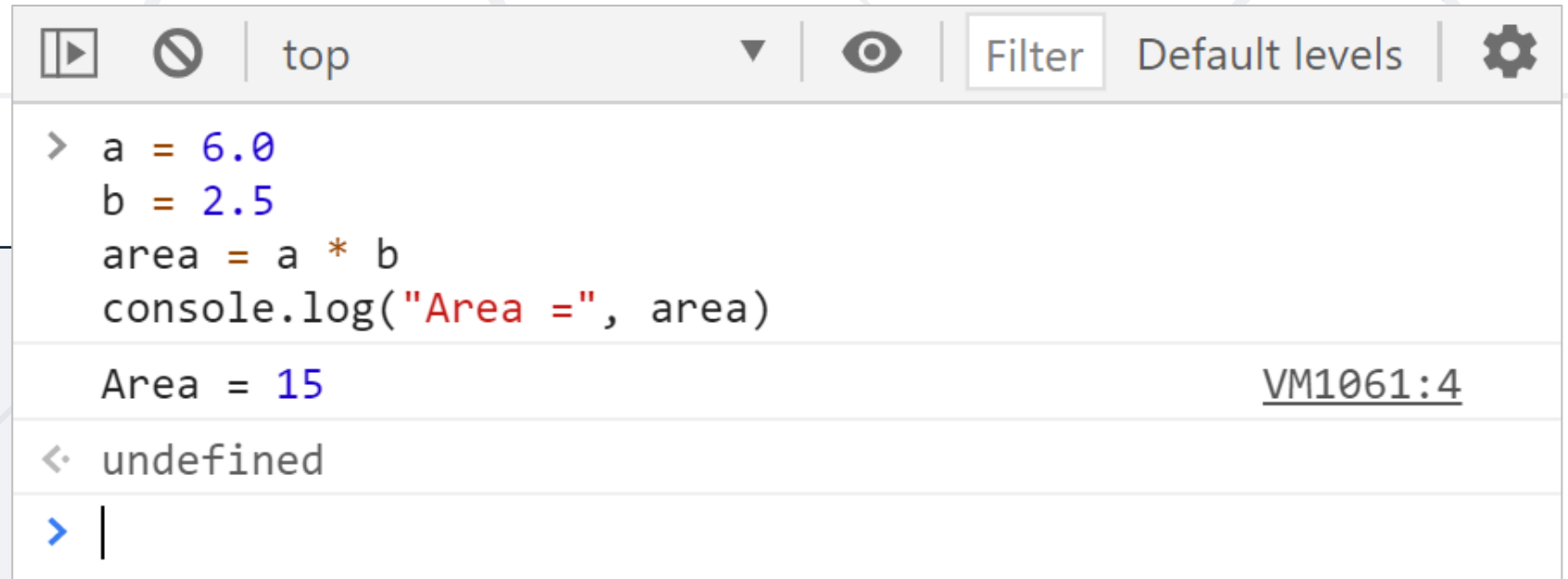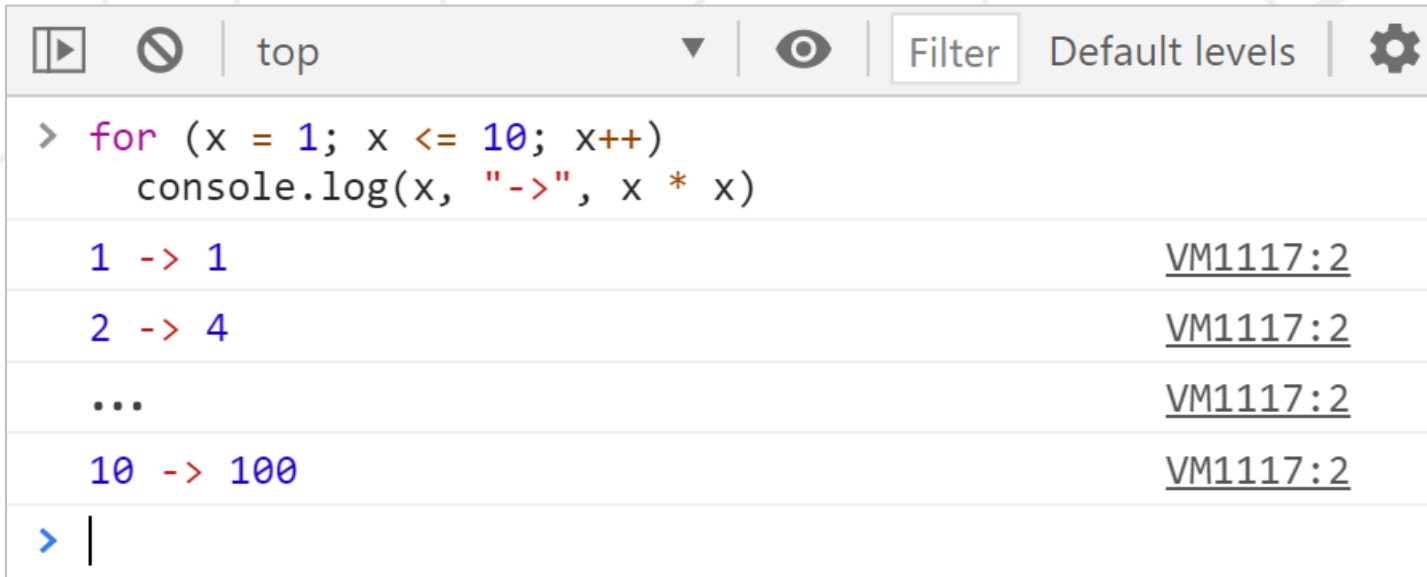
Enter a:

5

OK   Cancel

Enter b:

3

OK   Cancel

```
> a = prompt("Enter a:")
  b = prompt("Enter b:")
  console.log("Area = " + a * b)
Area = 15
```

- Print in JS the numbers **x** from **1** to **10** and their square x^2

```
for (x = 1; x <= 10; x++)
  console.log(x + " -> " + x * x)
```

```
  for (x = 1; x <= 10; x++)
    console.log(x, "->", x * x)

  1 -> 1                                      VM1117:2
  2 -> 4                                      VM1117:2
  ...                                         VM1117:2
  10 -> 100                                   VM1117:2
>
```

```
1 -> 1
2 -> 4
3 -> 9
4 -> 16
5 -> 25
6 -> 36
7 -> 49
8 -> 64
9 -> 81
10 -> 100
```

12

Programming Languages

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

Definition of Programming Language

- **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

- **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
```
100111010001101000000
011000110100011100110
100000101111011011110
111101100010110111000
100000100111000110011
100100110001110000000
```

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

MOV AX, 0x3C
MOV BX, 000000000001010B
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

- **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, HTML, CSS, SQL

- **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

- **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
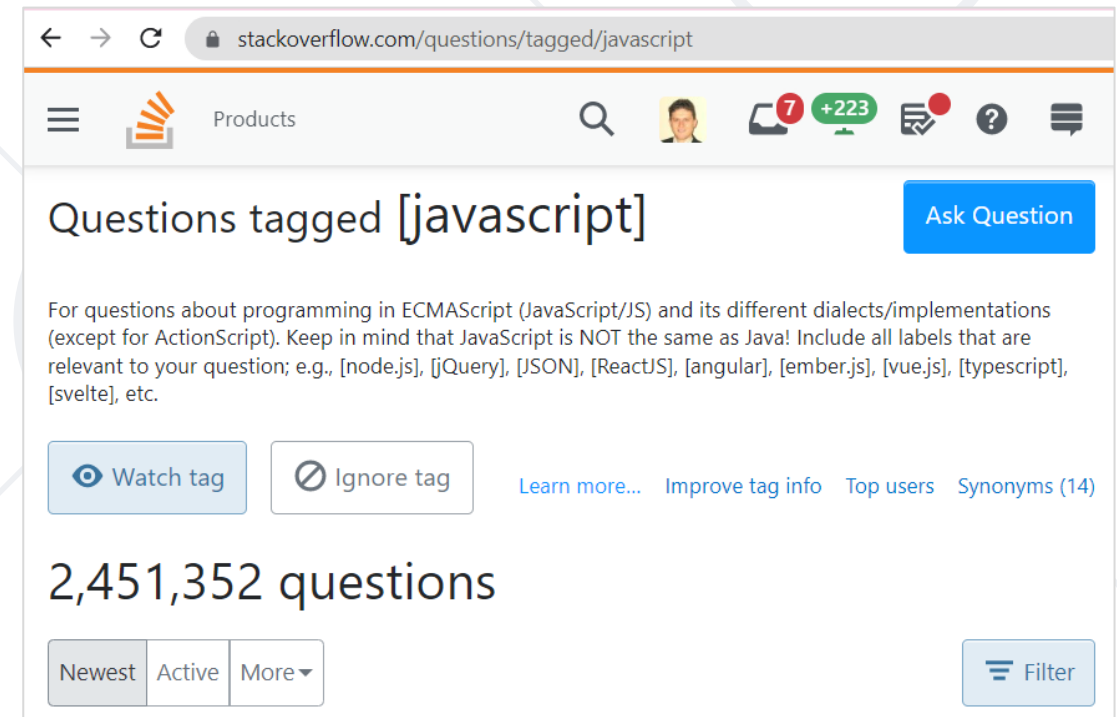
- Many tech languages are **not real programming languages!**

- **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)

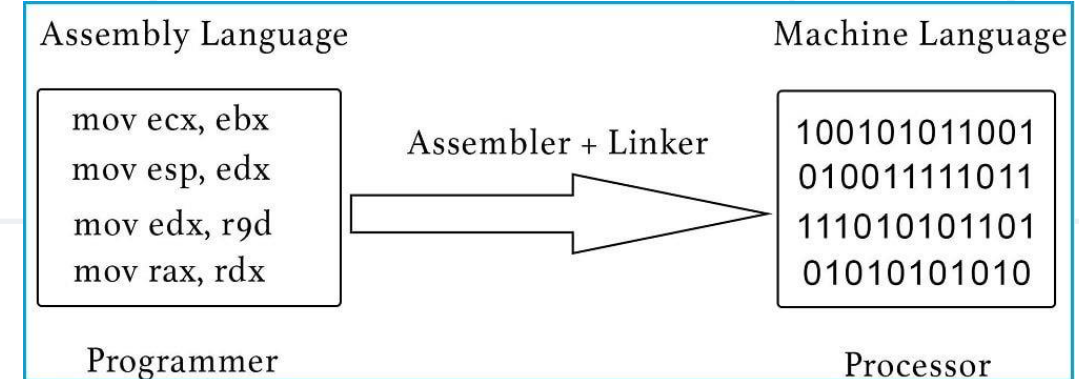# Low-Level & High-Level Languages

Assembler vs. Modern Languages

Low-level and high-level languages

- 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

- Almost **not used today**
  - **Modern compilers** are better than humans in machine code

Assembly Language		Machine Language

mov ecx, ebx
mov esp, edx
mov edx, r9d
mov rax, rdx
	Assembler + Linker →	100101011001
		010011111011
		111010101101
		01010101010

Programmer		Processor

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	111111111111111

23

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

- 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 Low-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

- 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 → compiler → machine code → execution

  - 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   source code → execution

  - 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

Variable name

```
int number = 5;
```

Data type

Variable value

Java – Introduction

- **Java** is modern, flexible, general-purpose programming language

- **Object-oriented** by nature, statically-typed, compiled

```
static void main(String[] args) {
    //Source Code

}
```

Program starting point

Java - Declaring Variables

- Defining and Initializing variables

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

- Example:

Variable name

```
int number = 5;
```

Data type

Variable value

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:

Variable name

Declare a variable with let

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

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

Printing to the Console

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

```javascript
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:

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

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

> Number of decimal places

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

Problem: Multiply Number by Two

- Write a function that receives a **number** and prints as a result that number **multiplied by two**

Input	Output
2	4

```javascript
function solve (num) {
  console.log(num * 2);
}
solve(2);
```
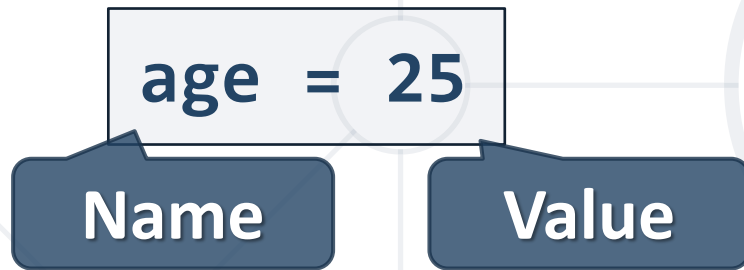
Python

- One of the **top 3** most popular programming languages
- One of the **most suitable** for beginners
- The syntax is close to plain **English language**
- **Interpreted** language
- Dynamic typing
- Object-oriented
- Cross-platform

Python - Basic Syntax

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

age = 25

Name   **Value**
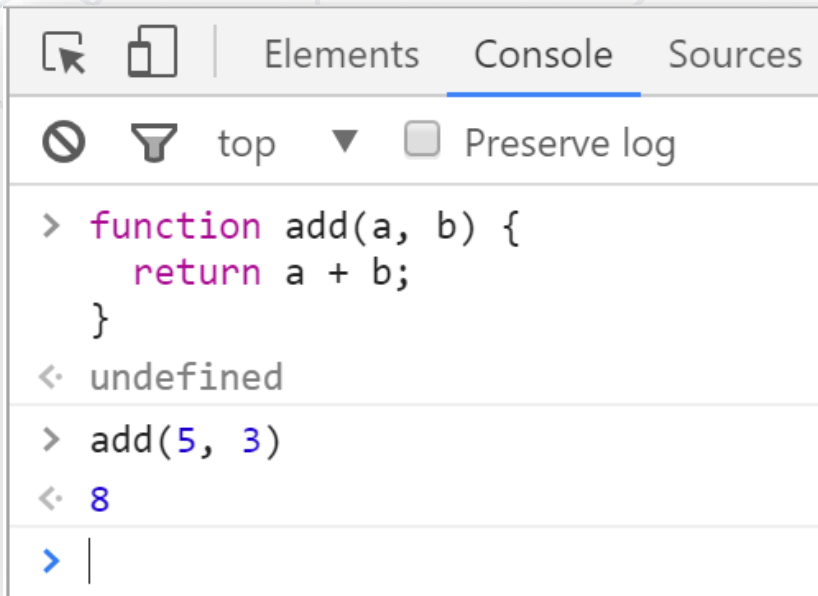
- Data types – variables are used to hold different data types

  - **int** - integer number: **1, 2, 3, 4, …**

  - **float** - real number: **0.5, 3.14, -0.5, …**

  - **str** - string and chars: **"a", "Hello", …**

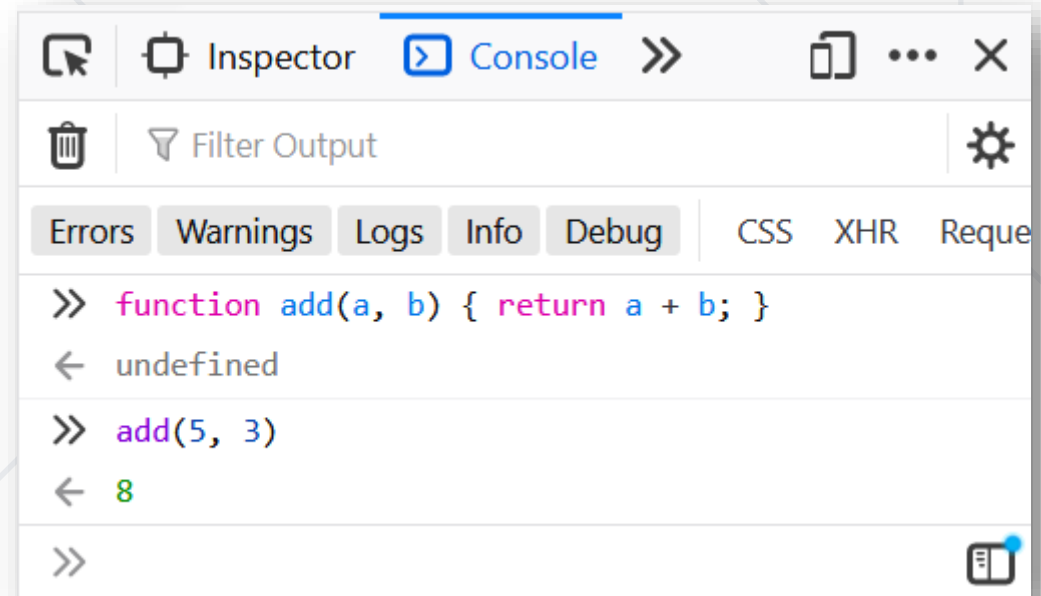 - **bool** - boolean: **True**, **False**

Live Demonstration

Setting up Node.js + VS Code

Chrome Web Browser

Developer Console: **[F12]**
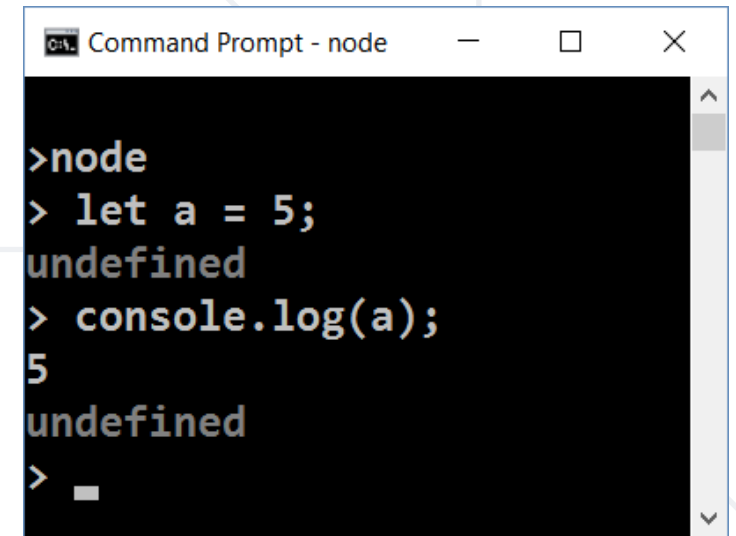
Node.js

- What is **Node.js**?

  - **Server-side** JavaScript runtime

  - Chrome V8 JavaScript engine

  - NPM **package manager**

  - Install node packages

```
Command Prompt - node                    —  □  ✕

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

Install the Latest Node.js

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	

Using Visual Studio Code

- **Visual Studio Code** is powerful text editor for JavaScript and other projects

- In order to create your **first project**:
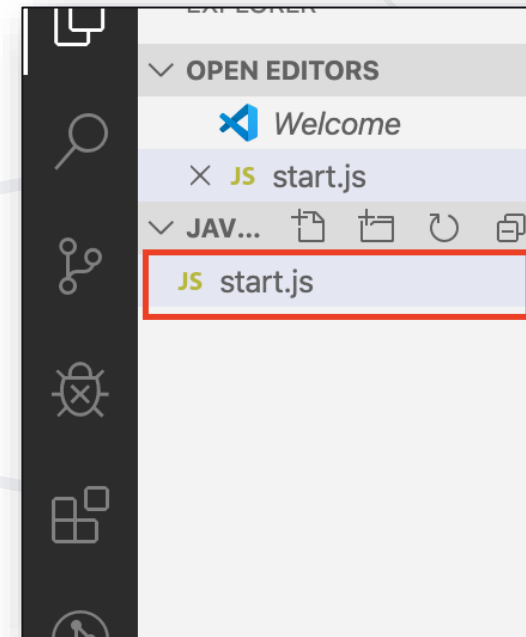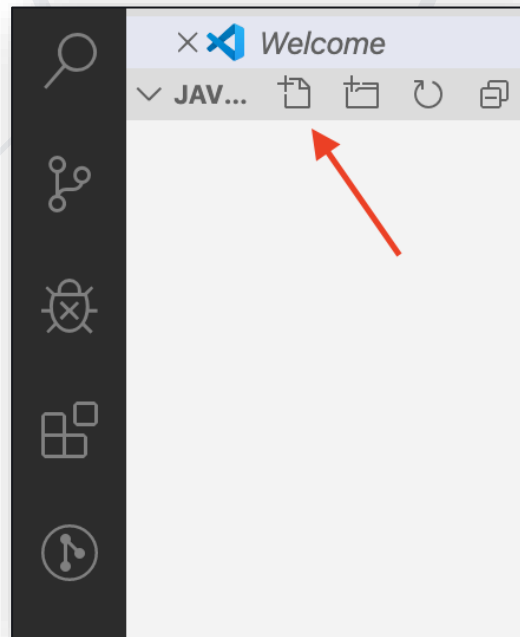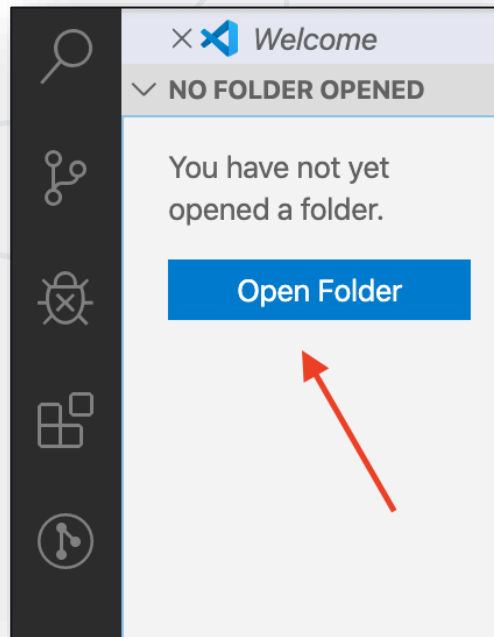
HTML & CSS & JavaScript

Front-End Design

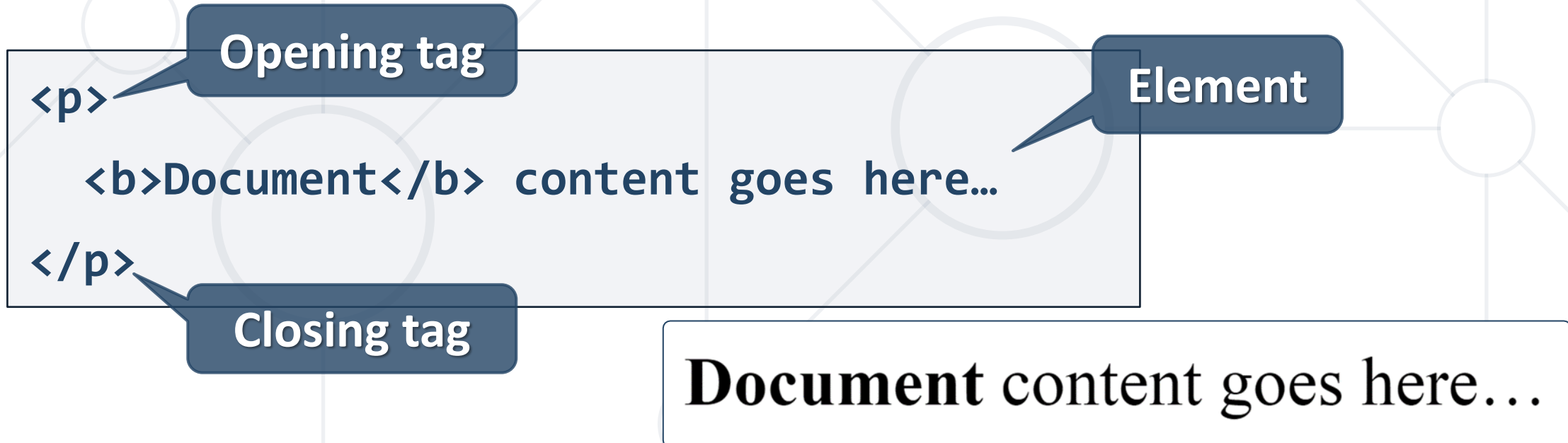# What is HTML?

- 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

Element

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

Closing tag

Document content goes here…

HTML – Developer Environments

- **WebStorm**
  - Powerful IDE for HTML, CSS and JavaScript, paid product

- **Visual Studio**
  - Many languages and technologies, Windows & Mac

- **Visual Studio Code**, **Brackets**, **NetBeans**
  - Good free tools for HTML5, cross-platform

HTML Page – Example

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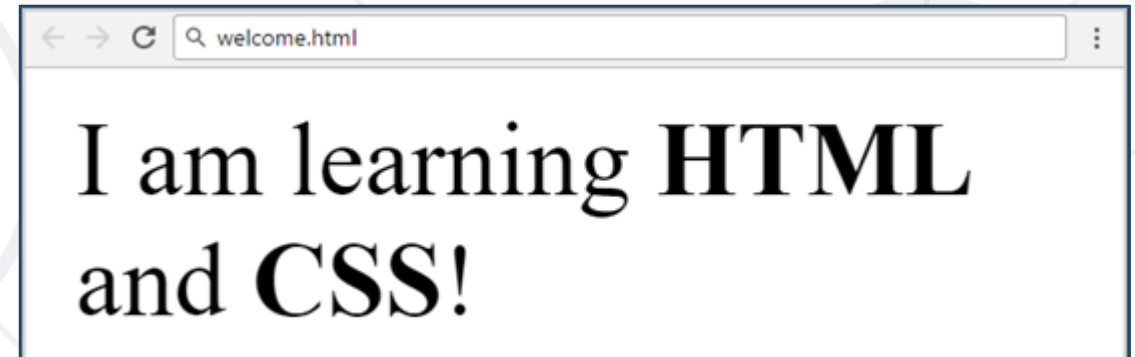
Problem: Welcome to HTML

- 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 **<strong>** 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:

```
h1 {
    font-size: 42px;
    color: yellow;
}
```

Selector

Declaration

Property

Value

53

Combining HTML and CSS Files (External Style)

using-css.html

```
<!DOCTYPE html>

<html>

    <head>

        <link rel="stylesheet" type="text/css" href="styles.css">

    </head>

    <body>

    </body>

</html>
```

styles.css

```css
/* CSS here */
```

Class Selector

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

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

```
<h2 class="class">…</h2>
```

- Only one HTML tag can be specified to be affected by a class

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

Id Selector

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

- Uses the **id attribute** of the HTML element

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

- Defined with a "**#**" in the CSS

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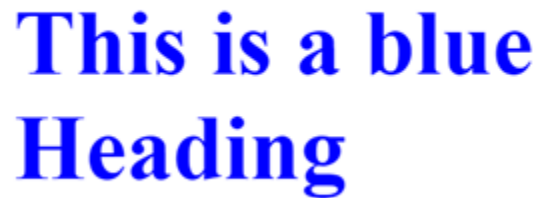
```
<body>
  <p class="red">This is red</p>
</body>
```

example.html

This is red

Inline CSS Style

- The **style** attribute in HTML elements

Attribute "style"

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

Property

Value

This is a blue Heading

JavaScript in the Browser

- Code can be **executed in the page** in different ways:

  - Directly in the **developer console** – when **debugging**

  - As a page **event handler** – e.g., user **clicks** on a button

  ```html
  <button onclick="console.log('Hello, DOM!')">Click Me</button>
  ```
  `event`

  - Via **inline** script, using **`<script>`** tags

  ```html
  <script>
      function sum(a, b) {
          let result = a + b;
          return result;
      }
  </script>
  ```
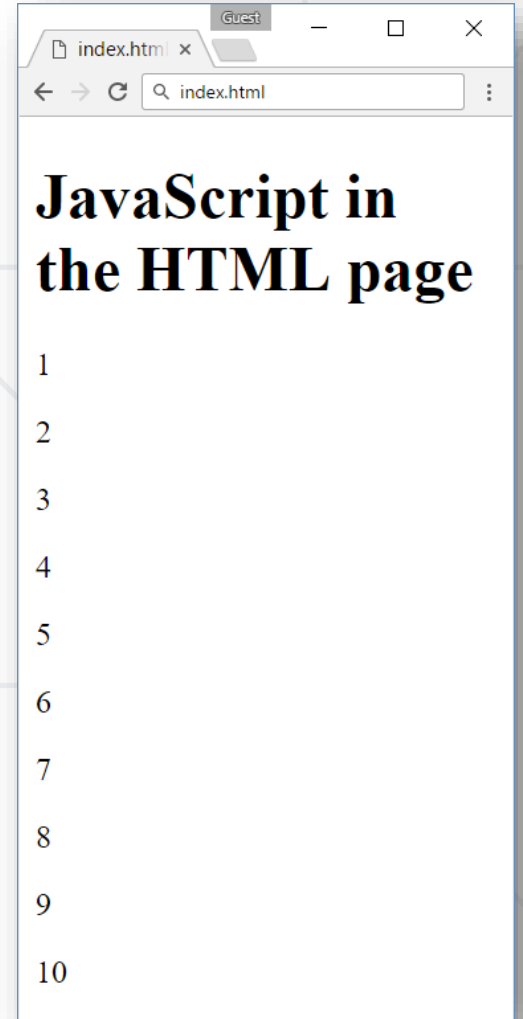
  - By **importing** from external file – most **flexible method**

Mixing HTML + JavaScript

```
<!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

```html
<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>
```

```javascript
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;
}
```
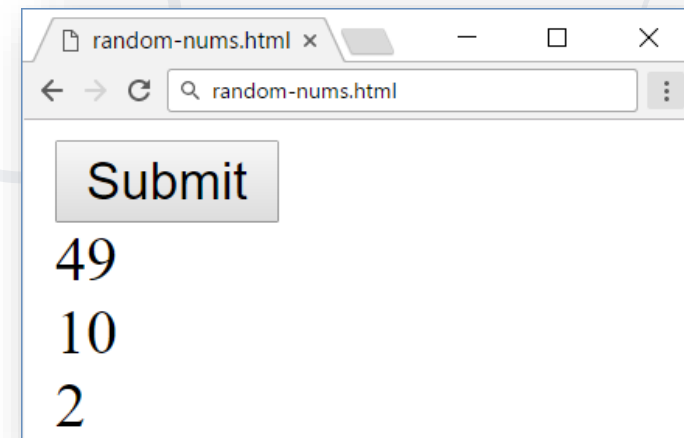
Load JavaScript File from HTML Document

random-nums.html

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

numbers.js

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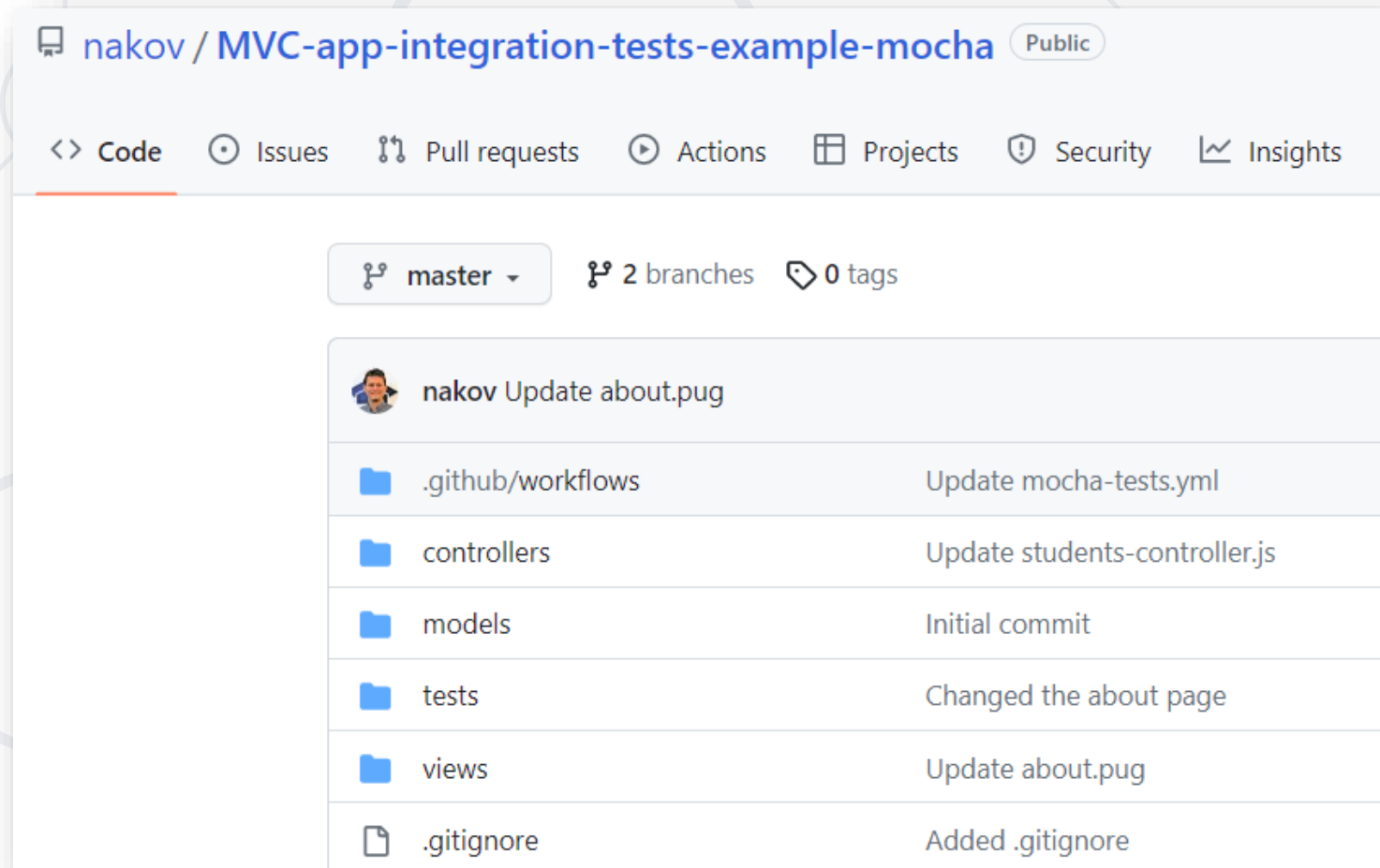
Real Software Project

Explore & Run

Explore Project @ GitHub Repository

- Link to Project: nakov/MVC-app-integration-tests-example-mocha

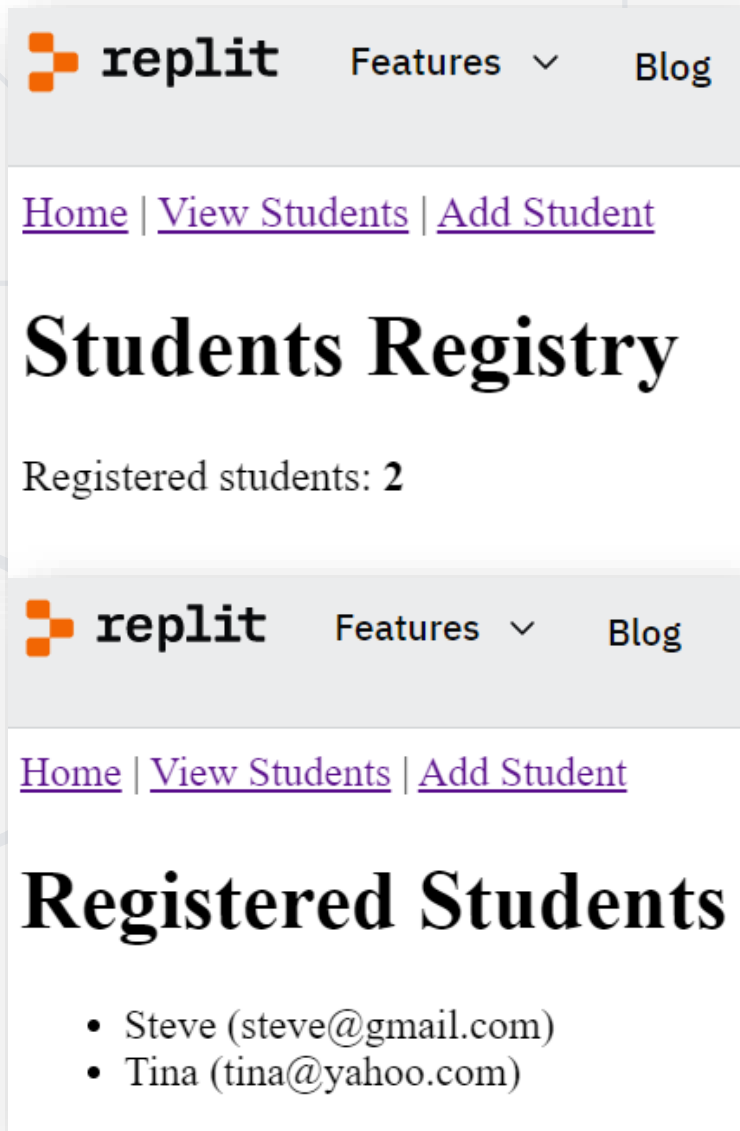# Run Project

README.md

MVC App with Integration Tests

mocha-tests

Integration tests of Web MVC app with Mocha, running in GitHub actions.

Production @ Heroku: https://nakov-mvc-node-app.herokuapp.com

Live demo at https://repl.it/@nakov/MVC-app-integration-tests-example-mocha

GitHub: https://github.com/nakov/MVC-app-integration-tests-example-mocha

Test results: https://github.com/nakov/MVC-app-integration-tests-example-mocha/actions

replit   Features ∨   Blog

Home | View Students | Add Student

Students Registry

Registered students: **2**

replit   Features ∨   Blog

Home | View Students | Add Student

Registered Students
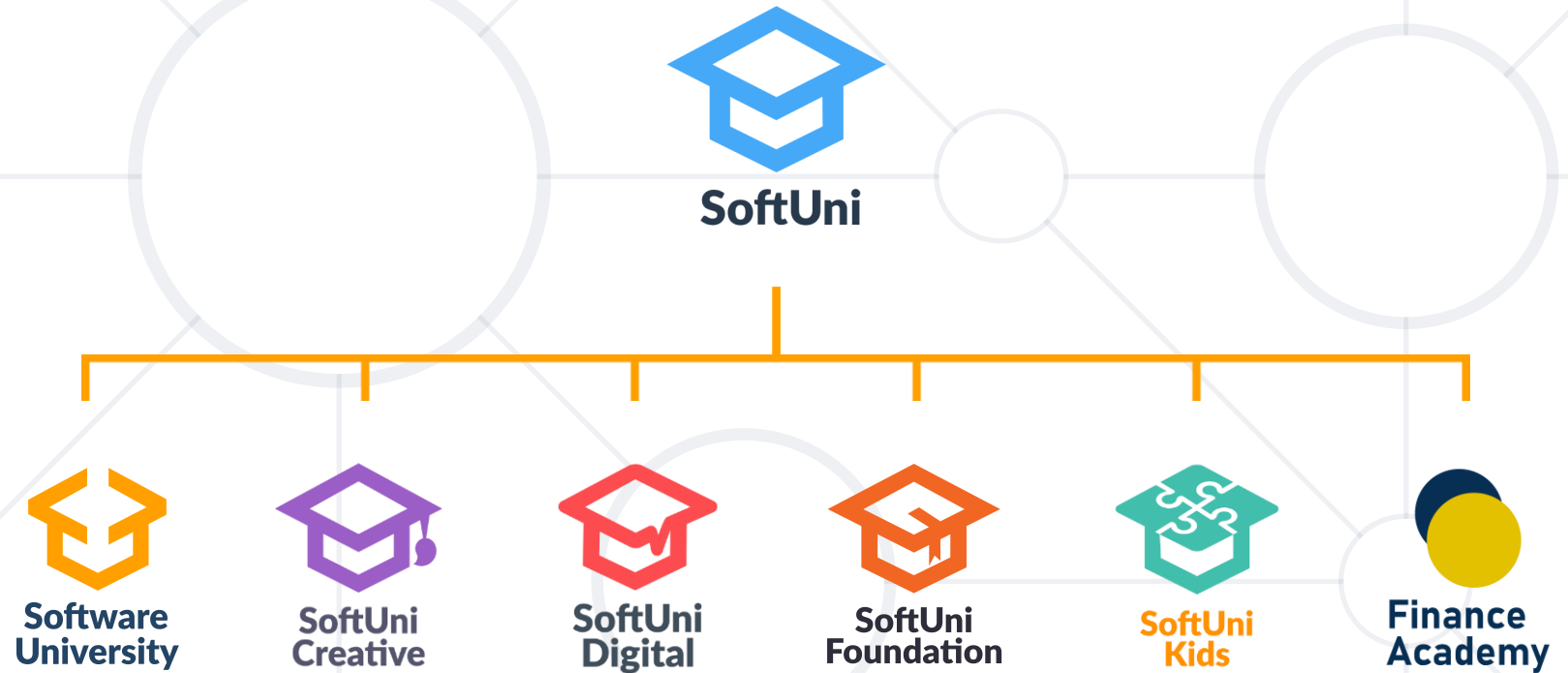
- Steve (steve@gmail.com)
- Tina (tina@yahoo.com)

Summary

- **Programming Languages:** definition & history

- **Low-Level** & **High-Level** languages

- **Scripting** vs. **Compiled** languages

- Most Popular Programming Languages: JavaScript, Python, Java, C#

- IDE Setup: Visual Studio Code

- **HTML**, **CSS** and **JavaScript** – Front-End Design
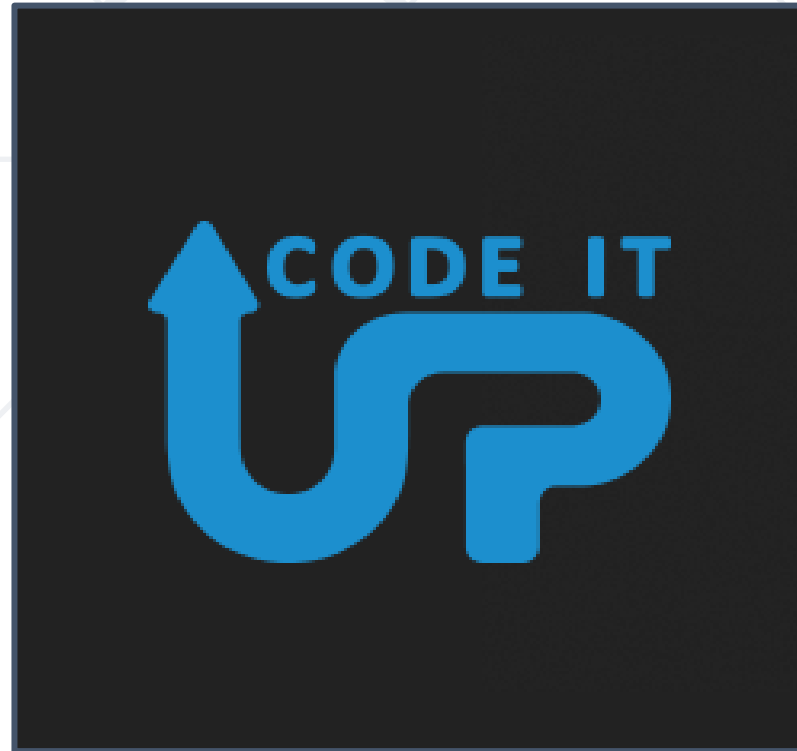
- Real-World Software Project

Questions?

SoftUni Diamond Partners

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