Functions and Forms

SoftUni Team

Technical Trainers

Software University

http://softuni.bg

Table of Content

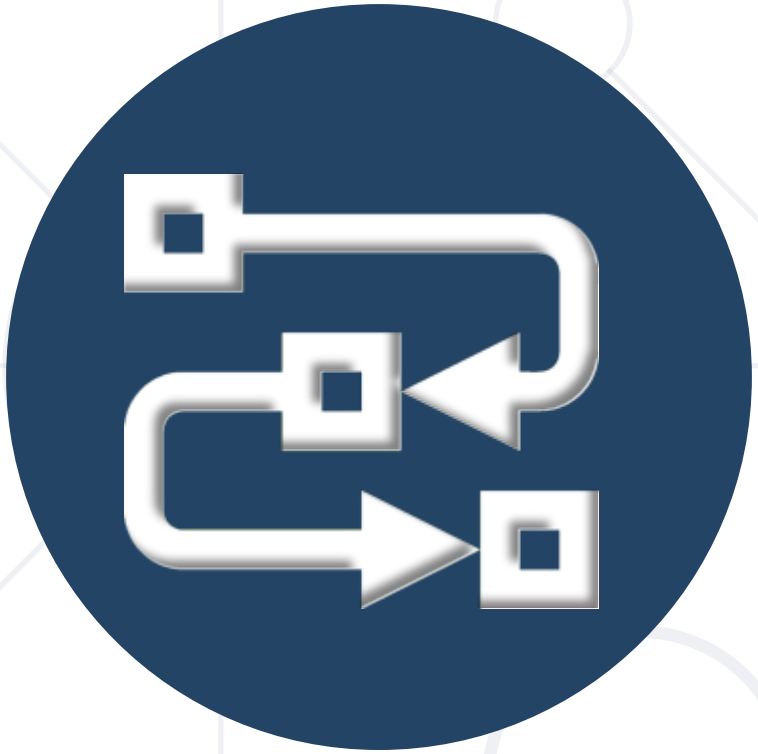
SoftUni Foundation

Have a Question?

SoftUni Foundation

sli.do

#tech-fund



JavaScript Functions Overview

Declaring and Invoking Functions

Functions in JS

- Function == named piece of code
  - Can take parameters and return result

Use camel-case

Function parameter

```
function printStars(count) {
  console.log("*".repeat(count));
}
```

Why Use Functions?

- More **manageable programming**
  - Splits large problems into small pieces
  - Better organization of the program
  - Improves code readability
  - Improves code understandability
- Avoiding **repeating code**
  - Improves code maintainability
- Code **reusability**
  - Using existing functions several times

Function Without Parameters

- Executes the code between the brackets
- Does not return result

```
function multiplyNumbers() {
    let result = 5 * 5;
    console.log(result);
}
multiplyNumbers();          //25
```

Prints result on the console

Declaring and Invoking Functions

Declaring Function

Function Name

Parameters

```
function printText(text){
  console.log(text);
}
```

Function Body

- Variables of type "**let**" inside a function
- Functions can have **several parameters**
- It is possible for function to **not** return a value

Invoking a Function

- Functions are first **declared**, then **invoked** (many times)

```
function printHeader(){
  console.log("----------");
}
```

Function **Declaration**

- Functions can be **invoked (called)** by their name:

```
function main(){
  printHeader();
}
```

Function **Invocation**

Invoking a Function (2)

SoftUni Foundation

- A function can be invoked from:
  - Other functions

```
function printHeader() {
  printHeaderTop();
  printHeaderBottom();
}
```

Function invoking functions

  - Itself (recursion)

```
function crash() {
  crash();
}
```

Function invoking itself

Problem: Car tax calculator

- Write a function that receives a **power** in kW of car's, between 0.00 and 150.00, and **calculates** and **prints** the tax you have to pay in lv.
  - under 37 kW - 0.43 lv./kW
  - 37.01 – 55 kW - 0.50 lv./kW
  - 55.01 – 74.00 - 0.68 lv./kW
  - 74.01 – 110.00 - 1.38 lv./kW
  - up 110.00 - 1.54 lv./kW

Input	Output
57.50	39.10 lv.

Solution: Car tax calculator

```
function solve(kW) {
    let power = Number(kW);
    calculate(power);
}
function calculate(power) {
    let tax = 0;
    if (power > 110 ){
        tax = power * 1.54;
    } else if (power > 74 ){
        tax = power * 1.38;
    } else //TODO
    tax = tax.toFixed(2);
    console.log(tax + ' lv.');
}
```

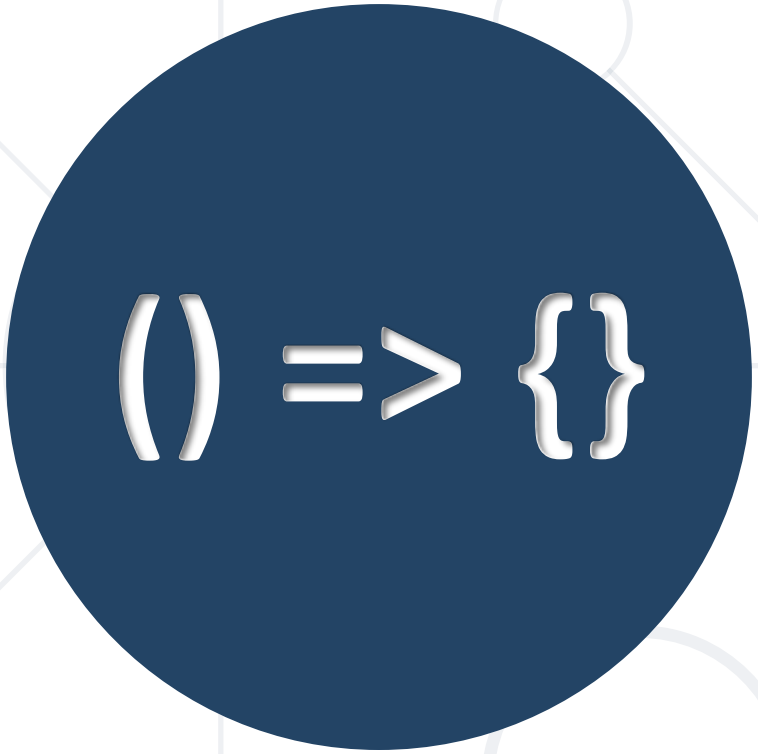
Problem: Car tax calculator II

- Add a new tax calculation function to the **previous code**
- Receive **two parameters**
  - Kilowatts
  - Car age
- The **coefficient** depends on the age of the vehicle
  - Under 5 years - **2.80**
  - 5 - 14 year - **1.50**
  - Up to 14 - **1.00**

Input	Output
45, 10	33.75 lv.

Solution: Car tax calculator II

```
function solve(kW, age) {
    let coefficient = calcCoeff(age);
    let powerPrice = calcPowerPrice(kW);
    totalPrice = (powerPrice * coefficient).toFixed(2) + " lv.";
    console.log(totalPrice);
}
function calcCoeff(age) {
    let coefficient = 0;
    if (age > 14){
        coefficient = 1;
    } else //TODO
    return coefficient;
} //TODO: implement the calcPowerPrice function
```



Arrow Functions

Arrow Functions

- Functions in JS can be written in **short form** using "**=>**" (**arrow**)

```
let increment = x => x + 1;
console.log(increment(5));  // 6
```

```
let increment = function(x) {
  return x + 1;
}
```

This is the same as the above function

```
let sum = (a, b) => a + b;
console.log(sum(5, 6));  // 11
```

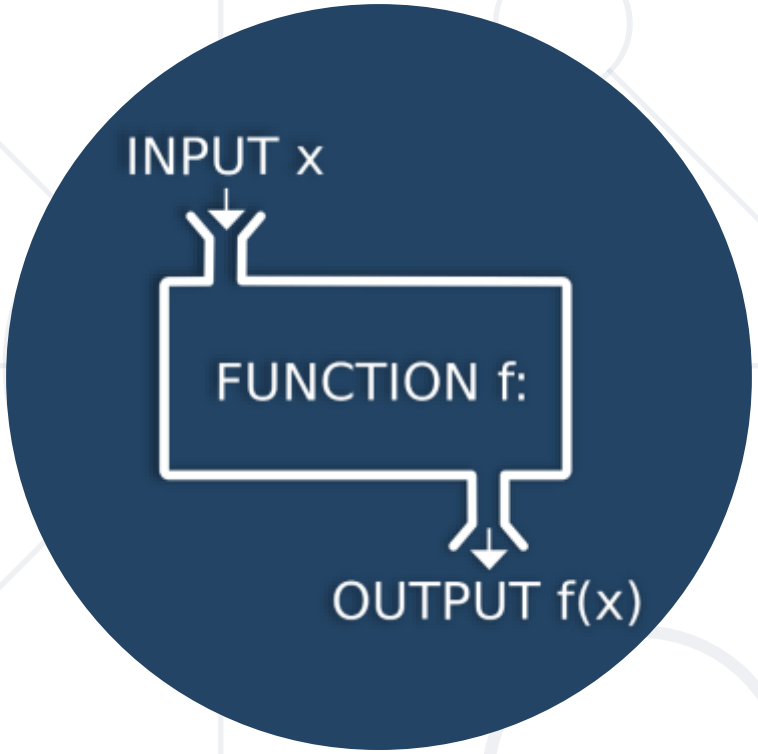
Problem: Simple calculator

- Write a function that **receives three parameters** and write an arrow function, that calculates a result depending on operator
- The operator can be '**multiply**', '**divide**', '**add**', '**subtract**'
- The input comes as three parameters - two **numbers** and an operator as a **string**

Input	Output
`5, 10, 'multiply'`	25

Solution: Simple calculator

```
function solve(a, b, operator) {
    switch (operator) {
        case 'multiply':
            let multyiply = (a, b) => a * b;
            console.log(multyiply(a, b));
            break;
        case 'divide':
            //TODO
        case 'add':
            //TODO
        case 'subtract':
            //TODO
    }
}
```



Nested Functions

Example

- Functions in JS can be **nested**, i.e. hold other functions
- Inner functions have access to variables from their parent

Main function

```
function drawDiamond(size) {
  drawTop(size / 2)
  drawBottom(size / 2)
}
```

Nesting the functions

Problem: Car tax calculator III

- Add new functionality to the calculator for the motorcycle tax
- Parameters:
  - First – **vehicle type**
  - Second – engine's **volume**
  - Third - **years**

Input	Output
'motorcycle', 450, 10	63.00 lv.

Solution: Car tax calculator III

```
function solve(type, kW, age) {
    switch (type){
        case "motorcycle":
            totalCalc(kW);
            break;
        case "car":
            let result = totalCalc(powerCalc(kW),calcCoeff(age));
            console.log(result);
            break;
    }
} //TODO: implement the other functions
```

Naming and Best Practices

Naming Functions

- Functions naming guidelines
  - Use **meaningful** function names
  - Should be in **camelCase**
  - Function names should answer the question:
    - **What does this function do**?

```
findStudent, loadReport, sine
```

```
Method1, DoSomething, handleStuff, DirtyHack
```

  - If you cannot find a good name for a function, think about whether it has a **clear intent**

Naming Function Parameters

- Function parameters names
  - Preferred form: [**Noun**] or [**Adjective**] + [**Noun**]
  - Should be in **camelCase**
  - Should be **meaningful**

```
firstName, report, speedKmH,
usersList, fontSizeInPixels, font
```

  - **Unit of measure should be obvious**

```
p, p1, p2, populate, LastName, last_name, convertImage
```

Functions – Best Practices

SoftUni Foundation

- Each function should perform a **single**, well-defined task
  - A Function's name should **describe that task** in a clear and non-ambiguous way
- **Avoid** functions **longer than one screen**
  - **Split them** to several shorter functions

```
function printReceipt(){
    printHeader();
    printBody();
    printFooter();
}
```

Self documenting and easy to test

Code Structure and Code Formatting

SoftUni Foundation

- Make sure to use correct **indentation**

```
function main(){
    // some code…
    // some more code…
}
```

```
function Main()
      {
            // some code…
// some more code…
}
```

- Leave a **blank line** between **functions**, after **loops** and after **if** statements
- Always use **curly brackets** for loops and if statements bodies
- **Avoid long lines** and **complex expressions**

Forms Definition

What is a Form

- A **webform**, **web form** or **HTML form** on a web page allows a user to enter data that is sent to a server for processing
- Forms can resemble **paper** or **database** forms because web users fill out the forms using **checkboxes**, **radio buttons** or **text fields**

Form example

Need Information

First Name	
Middle Name	
Last Name	
Address	
City	State
Zip/Postal Code	
Country	
Phone (country code, area code and number)	(+) -
E-mail	
Birth date	Day Month Year (4 digit)
Gender	
Starting date	Spring 2006 Summer 2006
Comments/Questions	This is the place for comments and questions...

Submit Reset this form

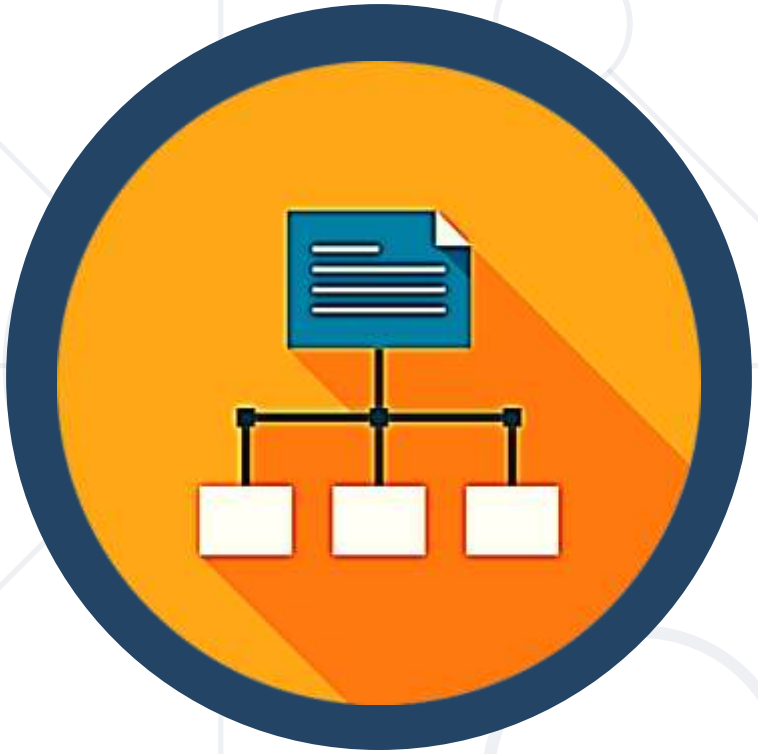
Input field

Text area

Select

Radio buttons

Submit button



Structure

Components of a form

Components

- **Form Name** - specifies the name of a form
- **Form Action** - specifies where to send the form-data when a form is submitted
- **Input Type** - defines an input control
- **onClick -** is an event handler

Code example

```
<form method="post">
Name: <input type="text" name ="name"/><br>
Email: <input type="text" name ="email"/><br>
Gender: <input type="radio" name ="gender"/>Male<br>
   <input type="radio" name ="gender"/>Female<br>
Comments: <br>
   <textarea name="comments">Any other
comments?</textarea><br>
<input type="submit"/><br>
<form>
```

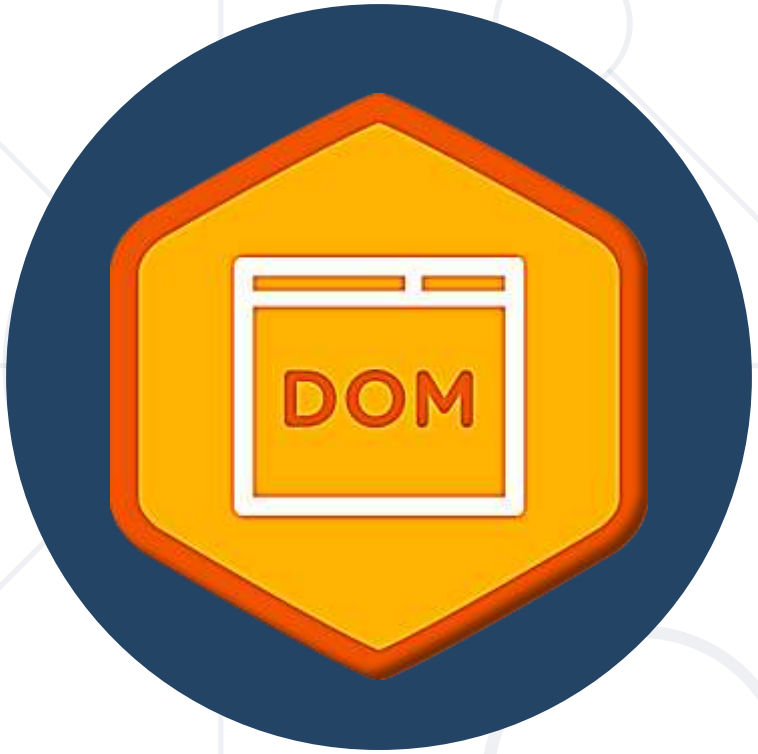
Type

Type of input

Radio buttons

Textarea

Submit button



DOM

Document Object Model

What is DOM

- What is **D**ocument **O**bject **M**odel (**DOM**)?
  - HTML documents in the browser are stored as "**DOM tree**"
  - Consists of **elements** with **child elements**
  - Elements have **properties** (attribute + value) and **events**
- The **DOM API** allows **searching / modifying** the DOM tree

```
let menu = document.getElementById('menu');
menu.style.display = 'none';
menu.appendChild(document.createElement('hr'));
```

Selecting HTML Elements from DOM

SoftUni Foundation

- Select a single element → returns **HTMLElement**

```
let header = document.getElementById('header');
let nav = document.querySelector('#main-nav');
let root = document.documentElement;
```

- Select a collection of elements → returns a **collection**

```
let inputs = document.getElementsByTagName('li');
let towns = document.getElementsByName('towns[]');
let header = document.querySelectorAll('#nav li');
let allLinks = document.links;
```

Accessing Element Text

- The contents of HTML elements are stored in text nodes
  - To access the contents of an element:

```
let element = document.getElementById('main');
let text = element.textContent;
element.textContent = "Welcome to the DOM";
```

```
<html>
  <head></head>
  <body>
    <div id="main">This is JavaScript!</div>
  </body>
</html>
```

```
<html>
  <head></head>
  <body>
    <div id="main">Welcome to the DOM</div>
  </body>
</html>
```

  - If the element has children, returns all text concatenated

Getting and setting values of a form

Getting and Setting Values

- Use DOM to access an element from the form

```
<input type= "text" name="name" id= "uniqueID"/>
```

```
let nameValue = document.getElementById("uniqueID").value
```

- Use DOM to set the value of element from the form

```
<input type= "text" name="name" id= "uniqueID"/>
```

```
document.getElementById("uniqueID").value = "Peter"
```

Additional components

More events

- **onFocus** - an event is triggered with a form object, gets input focus (the insertion point is clicked there).
- **onBlur** - an event is triggered with a form object, loses input focus (th e insertion point is clicked out of there).
- **onChange** - an event is triggered when a new item is selected in a list box.
- **onSelect** - an event is triggered when text in a text or text area box is selected.
- **onSubmit** - an event is triggered when the form is submitted to the server (more about this important hander later in the column).

Validating Forms

Data Validation

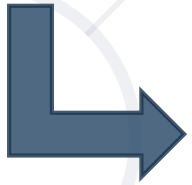
- Data validation is the process of ensuring that user input is **clean**, **correct**, and **useful**
- Typical validation tasks are:
  - Required fields must be filled
  - The input data is valid
  - The input must be the required type

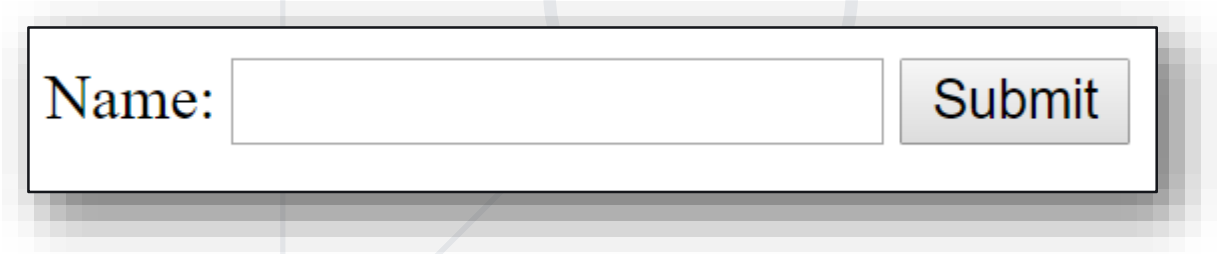
Automatic HTML Form Validation

- Make a field **required** to validate it:

```
<form method="post">
  <input type="text" name="fname" required>
  <input type="submit" value="Submit">
</form>
```

Making a field value **required**





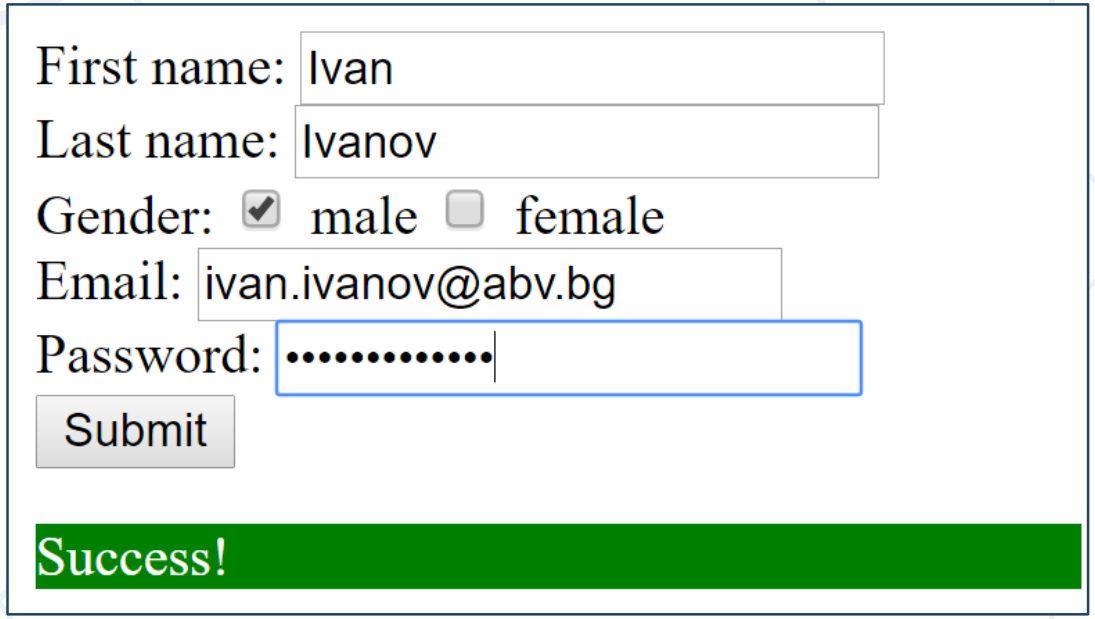
CSS Pseudo Selectors

- Use **CSS** to "validate" input fields

Selector	Description
:disabled	Selects input elements with the "disabled" attribute
:invalid	Selects input elements with invalid values
:optional	Selects input elements with no "required" attribute
:required	Selects input elements with "required" attribute
:valid	Select input elements with valid values

Problem: Forms

- Create a form that gathers information about a person
- There should be validation for each field:
  - First and Last name can not be empty
  - The user should check only one of the gender types
  - Email should include @ and a dot (.)
  - Password should be at least 6 symbols
  - If there are invalid fields print appropriate messages



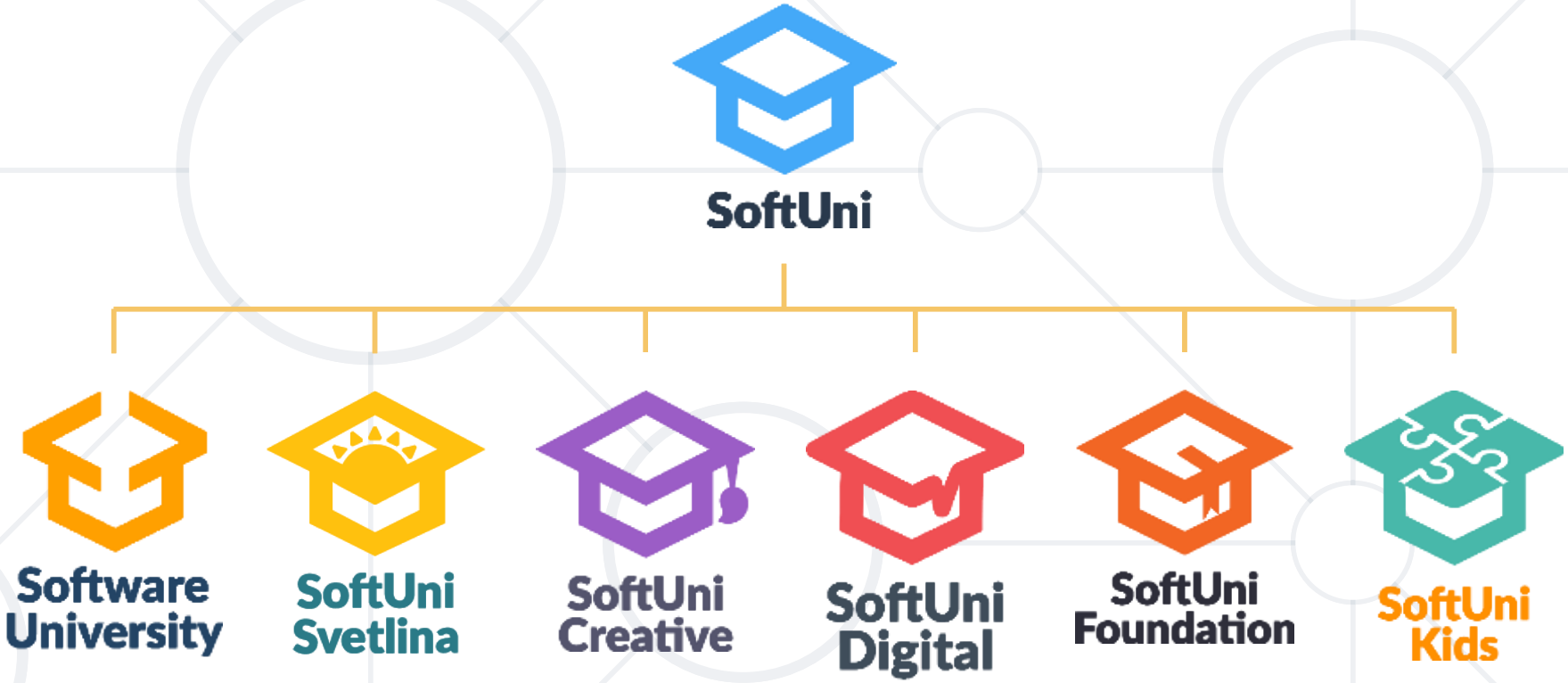
Live Exercises

Summary

SoftUni Foundation

- Functions:
  - Break large programs into simple **functions** that solve small sub-problems
  - Consist of **declaration** and **body**
  - Are invoked by their **name**
  - Can accept **parameters**
- Forms:
  - Are used to enter data that is sent to a server for processing
  - Are used to get and set values of the form with the help of **DOM**

Questions?



https://softuni.bg/trainings/courses

SoftUni Diamond Partners

SoftUni Foundation

XSsoftware

SBTech
we know sports

telenor

SoftwareGroup
doing it right

NETPEAK

SmartIT

Postbank
Решения за твоето утре

SUPER HOSTING .BG

INDEAVR
Serving the high achievers

INFRAGISTICS

LIEBHERR

æternity

codexio

SoftUni Organizational Partners

SoftUni Foundation

ИНФОРМАЦИОННО ОБСЛУЖВАНЕ

OneBit SOFTWARE

WORLD OF MYTHS

Lukanet.com

codeXio

Trainings @ Software University (SoftUni)

- 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