Advanced Functions

First Class Functions, Function Expressions, Predicates, Currying, IIFE, Closure

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

Have a Question?

Software University

sli.do

#js-advanced

Functions Behaving Like Variables

First Class Functions

First-Class Functions

- Can be passed as an **argument** to another function

```
function sayHello() {
    return "Hello, ";
}
```

```
function greeting(helloMessage, name) {
    return helloMessage() + name;
}
```

```
console.log(greeting (sayHello, "JavaScript!"));
// Hello, JavaScript!
```

First-Class Functions

- Can be **returned** by another function
  - We can do that, because we treated functions in JavaScript as a **value**

```
function sayHello() {
    return function () {
        console.log('Hello!');
    }
}
```

First-Class Functions

- Can be assigned as a **value** to a **variable**

```
const write = function () {
    return "Hello, world!";
}
```

- That function can be **invoked** by adding parentheses "**()**" at the end after the variable name

```
console.log(write()); // Hello, world!
```

Higher-Order Functions

- Take other **functions** as an **argument** or **return a function** as a result

```
const sayHello = function () {
  return function () {
     console.log("Hello!");
  }
}
```

```
const myFunc = sayHello();
myFunc(); // Hello!
```

Predicates

- Any function that returns a **bool based** on evaluation of the **truth** of an **assertion**
- Predicates are often found in the form of **callbacks**

```
let found = array1.find(isFound);

function isFound(element) {
    return element > 10; //True or false
}

console.log(found); // 12
```

predicate

Currying

Currying

- Currying is a technique for **function decomposition**

```
function sum3(a) {
    return (b) => {
        return (c) => {
            return a + b + c;
        }
    }
}
console.log(sum3(5)(6)(8)); // 19
```

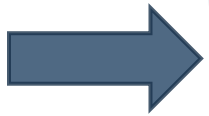
Currying Usage

- Usage
  - Template functions
  - Code reuse
  - Partial implementation
  - Retain scope

Partial Application

- Converting a function with a **given number** of arguments into a function with **smaller number** of arguments
- Pass the **remaining parameters** when a result is needed
  - The partially applied function can be **used multiple times**

```
f(x, y) = x + y
```



```
g(x) = f(1, x)
```

Currying vs Partial Application

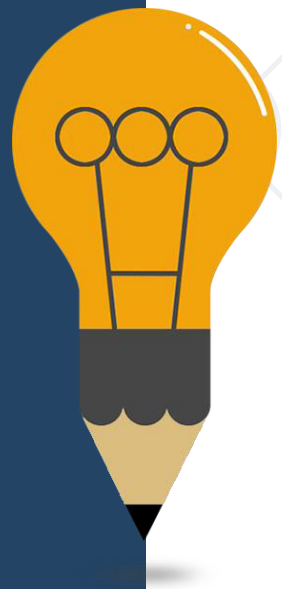
- **Currying** always produces nested unary functions
- **Partial** application produces functions of arbitrary number of arguments
- Currying is **NOT** partial application
  - It can be implemented using partial application

Immediately-Invoked Function Expressions

What is IIFE?

Immediately-Invoked Function Expressions (IIFE)

- Define **anonymous** function expression
- Invoke it **immediately** after declaration



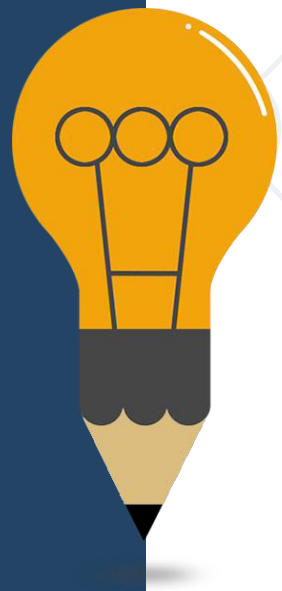
```
(function () { let name = "Peter"; })();
// Variable name is not accessible from the outside scope
console.log(name); // ReferenceError
```

```
let result = (function () {
    let name = "Peter";
    return name;
})();
// Immediately creates the output:
console.log(result); // Peter
```

Closure

Closure

- One of the most **important features** in JavaScript
- The **context** of an inner function **includes** the **scope** of the outer function
- An inner function **enjoys** that **context** even **after** the parent function have **returned**



Functions Returning Functions

- A **state** is preserved in the outer function (**closure**)

```
const f = (function () {
    let counter = 0;
    return function () {
        console.log(++counter);
    }
})();
```

```
f(); // 1
f(); // 2
f(); // 3
f(); // 4
f(); // 5
f(); // 6
f(); // 7
```

Problem: Command Processor

- Write a program, which:
  - Keeps a string **inside its scope**
  - Can execute different **commands** that modify a string:
    - `append()` - add **str** to the end of the internal string
    - `removeStart()` - **remove** the **first n** characters
    - `removeEnd()` - remove the **last n** characters
    - `print()` - print the stored string

Solution: Command Processor

```
function solution() {
    let str = '';
    return {
        append: (s) => str += s,
        removeStart: (n) => str = str.substring(n),
        removeEnd: (n) => str = str.substring(0, str.length - n),
        print: () => console.log(str)
    }
}
```

Live Exercises

Summary

- **First Class Functions**
  - Can be passed as an **argument**
  - Can be **returned**
- **Higher-Order Functions**
  - Take other **functions** as an **argument** or **return** a function
- **IIFE** is immediately-invoked anonymous function
  - Encapsulates **JS code** + **data** (state)

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

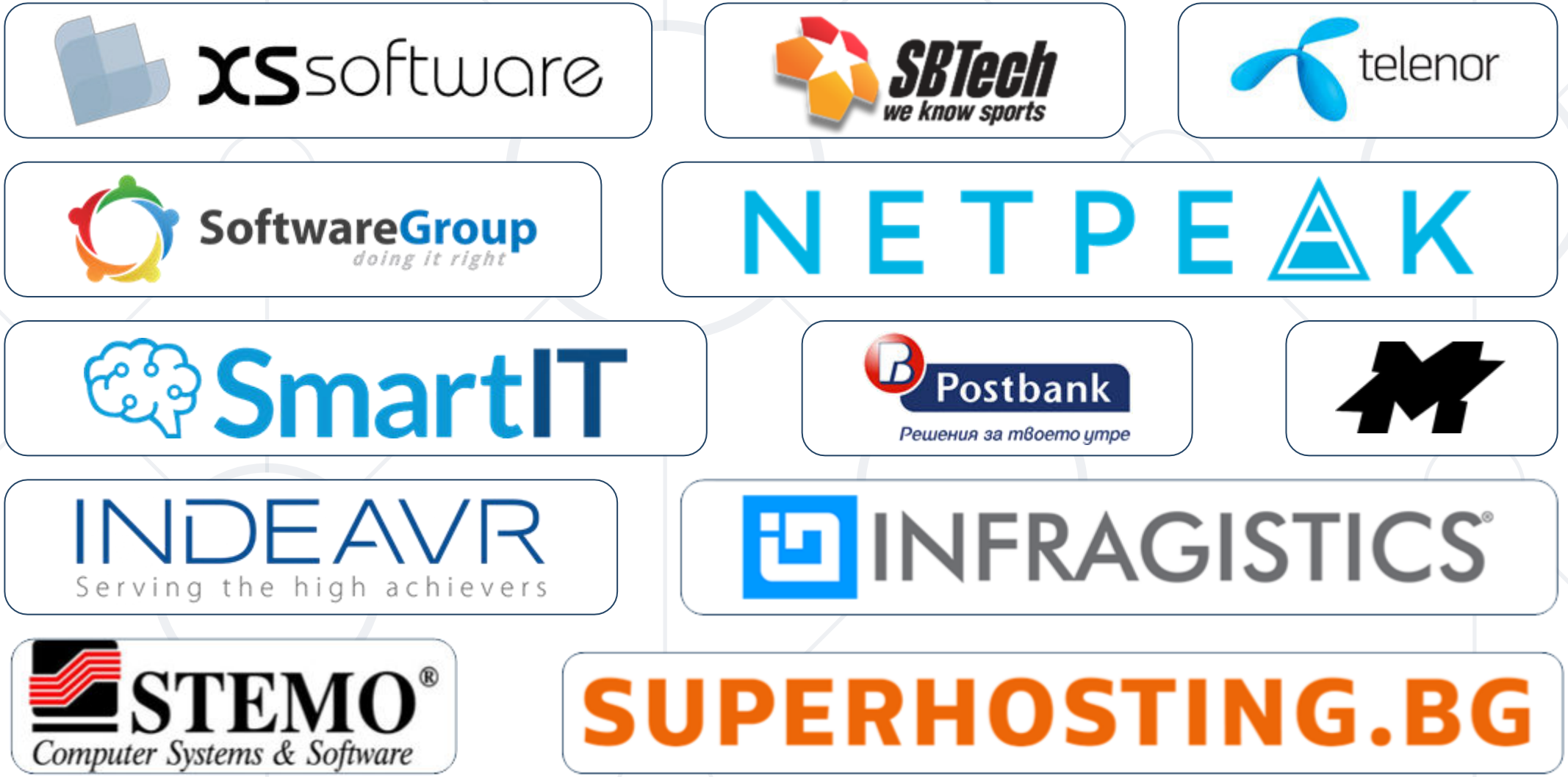
SoftUni Digital

SoftUni Foundation

SoftUni Kids

SoftUni Diamond Partners

Software University



SoftUni Organizational Partners

Software University

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity
- Software University Forums
  - forum.softuni.bg

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://softuni.org
- © Software University – https://softuni.bg