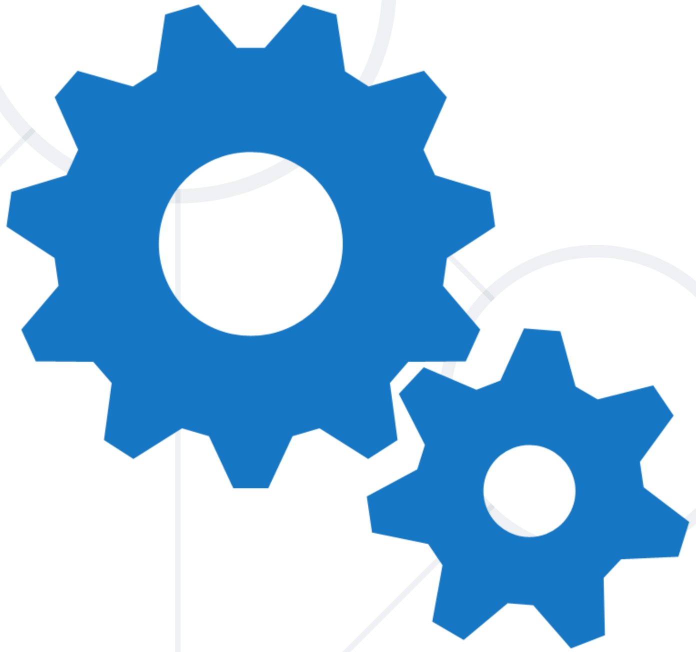
Advanced Functions

First-Class Functions, Function Expressions, IIFE, this, call, apply



SoftUni Team

Technical Trainers

Software University

SoftUni Foundation

Software University

http://softuni.bg

Table of Content

Have a Question?

SoftUni Foundation

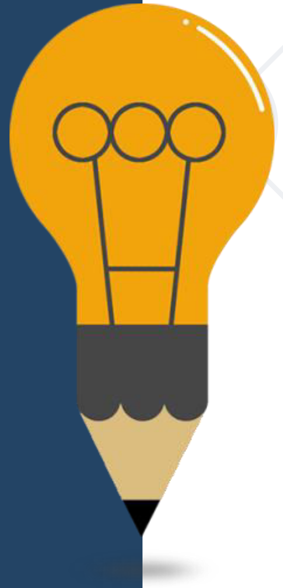
sli.do

#JSCORE

First Class Functions

First-Class Functions in JS

- What does "**first-class functions**" mean?
  - **Functions** and **objects** are treated as the same thing



```
function hello() {
  console.log("Function hello() invoked.");
}

hello();

hello.speed = 200;
console.log(hello.name + ' ' + hello.speed);
```

Function Declarations in JS

```
function myfunc1(val) {
 return val + 1;
}
```

Function declaration

```
let myfunc2 = function(val) {
  return val + 1;
}
```

Function expression

Function constructor

```
let myfunc3 = new Function("val", "return val + 1;");
```

Higher-Order Functions

- What does "**higher-order functions**" mean?
  - Take other **functions as argument** or **return a function** as result

```
function invokeAll(functionsArr) {
  for (let func of functionsArr){
     func();
  }
}
let last = function() {
   console.error("last");
}

invokeAll([() => console.info('first'), () =>
console.warn('second'), last]);
```

Example: Reducer Function

- A **reducer** applies a function over a sequence of elements to produce a single result, a.k.a. **aggregate function** (e.g. **sum**, **max**)

```
function reduce(arr, func) {
  let result = arr.shift();
  for (let nextElement of arr)
    result = func(result, nextElement);
  return result;
}
reduce([5, 10, 20], (a,b) => a + b); // 35
reduce([5, 10, 20], (a,b) => a * b); // 1000
```

Problem: Aggregates

- You are given an array of numbers
  - Using a **reducer** function, print its: **sum**, **min**, **max**, **product**, **join**

```
2
3
10
5
```

```
Sum = 20
Min = 2
Max = 10
Product = 300
Join = 23105
```

```
5
-3
20
7
0.
5
```

```
Sum = 29.5
Min = -3
Max = 20
Product = -1050
Join = 5-32070.5
```

Solution: Aggregates

```
function calcAggregates(arr) {
  console.log("Sum = " + arr.reduce((a,b) => a + b));
  console.log("Min = " + arr.reduce((a,b) => Math.min(a,b)));
  console.log("Max = " + arr.reduce((a,b) => Math.max(a,b)));
  console.log("Product = " + arr.reduce((a,b) => a * b));
  console.log("Join = " + arr.reduce((a,b) => '' + a + b));
}
```

```
calcAggregates([2, 3, 10, 5])
```

Check your solution here: https://judge.softuni.bg/Contests/330

Solution: Aggregates

```
function calcAggregates(arr) {
  console.log("Sum = " + arr.reduce((a,b) => a + b));
  console.log("Min = " + arr.reduce((a,b) => Math.min(a,b)));
  console.log("Max = " + arr.reduce((a,b) => Math.max(a,b)));
  console.log("Product = " + arr.reduce((a,b) => a * b));
  console.log("Join = " + arr.reduce((a,b) => '' + a + b));
}
```

```
calcAggregates([2, 3, 10, 5])
```

Check your solution here: https://judge.softuni.bg/Contests/330

Partial Application

- Set **some of the parameters** of a function to a **fixed value**
- Pass the **remaining parameters** when a final **result** is needed
  - The partially applied function can be **used multiple times**
- Example:

Set first parameter to 1

Same as increment operator (++)

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

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

- This helps write **reusable code** with **fewer bugs**

Problem: Currency Format

- You are **given** a function that formats **currency** values

```
function formatCurrency(separator, symbol, symbolFirst, value) {
  let result = Math.trunc(value) + separator;
  result += value.toFixed(2).substr(-2,2);
  if (symbolFirst) return symbol + ' ' + result;
  else return result + ' ' + symbol;
}
```

- **Return a function** that formats dollar values

```
let formatter = getDollarFormatter(formatCurrency);
formatter(5345); // $ 5345,00
```

Solution: Currency Format

- We take the initial **function as parameter**
- We **return a function** that takes only one parameter

```
function getDollarFormatter(formatter) {
  function dollarFormatter(value) {
      return formatter(',', '$', true, value);
  };
  return dollarFormatter;
}
```

Fix parameters

Return result of **original** function

- This is called "**function currying**" (after **Haskell Curry**)

Function Properties

```
function max(arr) { return arr; }
console.log(max.length); // 1 (number of arguments)
console.log(max.name); // max
console.log((function(){}).name); // (empty string)
```

```
function inner() {
  console.log("Caller: " + inner.caller);
}
function outer() { inner() };
outer(); // Caller: function outer()
```

IIFE

Immediately-Invoked Function Expressions (IIFE)

Using IIFE to Hide State inside a Function

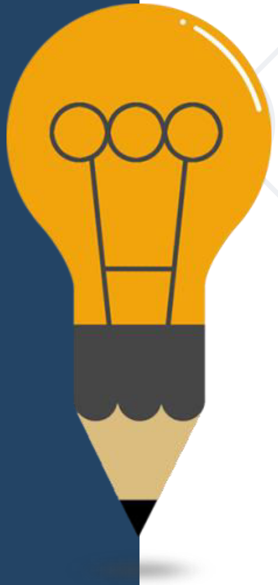
What is IIFE?

- Immediately-Invoked Function Expressions (IIFE)
  - Define anonymous function expression
  - Invoke it immediately after declaration

```
(function() { console.log("invoked!"); }());
```

```
(function() { console.log("invoked!"); })();
```

```
let iife = function() { console.log("invoked!"); }();
```



IIFE: The Problem

```
let arr = [10, 20, 30];
```

```
let sum = 0;
for (let x of arr) {
  sum += x;
}
console.log(sum);
// "sum" and "arr" remain visible in the current scope
```

IIFE: The Problem (2)

```
function sumArray(arr) {
  let sum = 0;
  for (let x of arr)
    sum += x;
  console.log(sum);
}
sumArray([10, 20, 30]);
// The function "sumArray" remains in the current scope
// The "sum" variable is "hidden" in the function
```

IIFE: The Solution

SoftUni Foundation

```
(function(arr) {
  let sum = 0;
  for (let x of arr)
    sum += x;
  console.log(sum);
})([10, 20, 30])

// Nothing remains in the current scope
// "sum" and "arr" are "hidden" in annonymous function
```

IIFE

Functions Returning Functions

- In JS a function can **return another function**
  - A **state** is preserved in the outer function, a.k.a. **closure**

```
let 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: String Command Processor

- Using a **closure** (IIFE holding a state inside it) implement a command execution engine to **process string commands** like shown below

```
append hello
append again
removeStart 3
removeEnd 4
print
```

```
loa
```

```
hello
```

```
helloagain
```

```
loagain
```

```
loa
```

```
loa
```

Solution: String Command Processor

```
let commandProcessor = (function() {
  let text = '';
  return {
    append: (newText) => text += newText,
    removeStart: (count) => text = text.slice(count),
    removeEnd: (count) => text = text.slice(0, text.length - count),
    print: () => console.log(text)
  }
})();
```

Return object with functions as properties

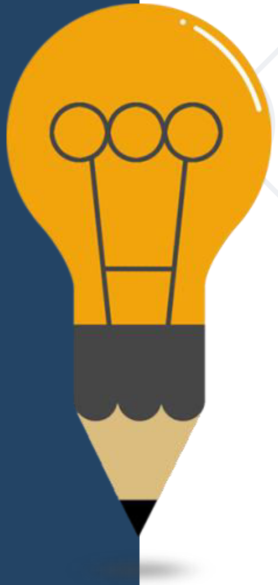
Check your solution here: https://judge.softuni.bg/Contests/330

Function "this" Context

this, call, apply, bind

What is Function Context?

- The **function context** is the object that "**owns**" the currently executed code
  - Function context == "**this**" object
  - Depends on how the function is invoked
    - Global invoke: **func()**
    - **object.function()**
    - **domElement.event()**
    - Using **call()** / **apply()** / **bind()**



The Function Context

```
function f() {
  console.log(this);
}
f(); // Window ("this" is the global context)
```

```
function f() {
  'use strict';
  console.log(this);
}
f(); // undefined ("this" is missing)
```

The Function Context with Object

```
function func() {
  console.log(this);
}

let obj = {
  name: 'Peter',
  f: func
};

obj.f(); // Object {name: "Peter"}
```

The Function Context for Objects

```
let obj = {
  name: 'Todor',
  getName: function () {
    return this.name;  // "this" refers to "obj"
  }
};
console.log(obj.getName()); // Todor
```

```
function Car() {
  console.log(this);
}
let car = new Car(); // Car {}
```

The Function Context with Inner Function

```
function outer() {
  console.log(this); // Object {name: "Peter"}
  function inner() {
    console.log(this); // Window
  }
  inner();
}
let obj = { name: 'Peter', f: outer }
obj.f();
```

The Function Context with Arrow Function

SoftUni Foundation

```
function outer() {
  let inner = () => console.log(this);
  inner();
}
let obj = {
  name: 'Peter',
  f: outer
};
obj.f(); // Object {name: "Peter"}
```

The Function Context for DOM Events

```
<button onclick="alert(this)">Click Me</button>
// Shows "[object HtmlButtonElement]" when clicked
```

```
<button onclick="f(this)">Click Me</button>
function f(btn) { alert(btn); };
// Shows "[object HtmlButtonElement]" when clicked
```

```
<button onclick="f()">Click Me</button>
function f() { alert(this); };
// Shows "[object Window]" when clicked
```

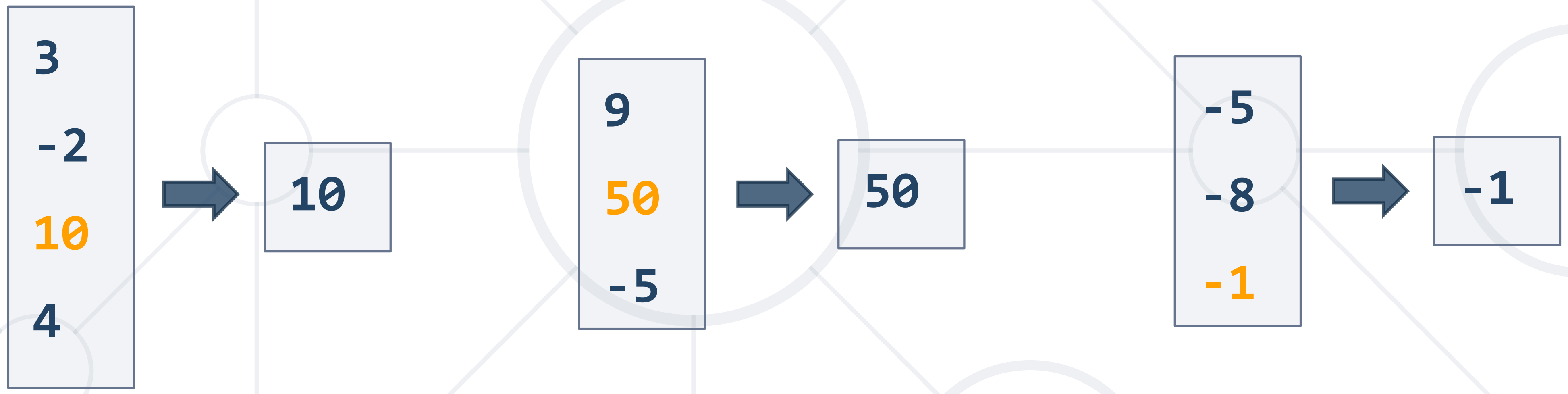
Avoided by using **addEventListener**

Changing the Context: Call and Apply

```
let maria = {
  name: "Maria",
  hello: function(thing) {
    console.log(this.name + " says hello " + thing);
  }
}
maria.hello("world"); // Maria says hello world
let ivan = { name: 'Ivan' };
maria.hello.call(ivan, "now"); // Ivan says hello now
maria.hello.apply(ivan, ["again"]); // Ivan says hello again
```

Problem: Max Number in Array

- Given an array of numbers, find the biggest number



- Solution:

```
function maxElement(arr) {
  return Math.max.apply(null, arr);
}
```

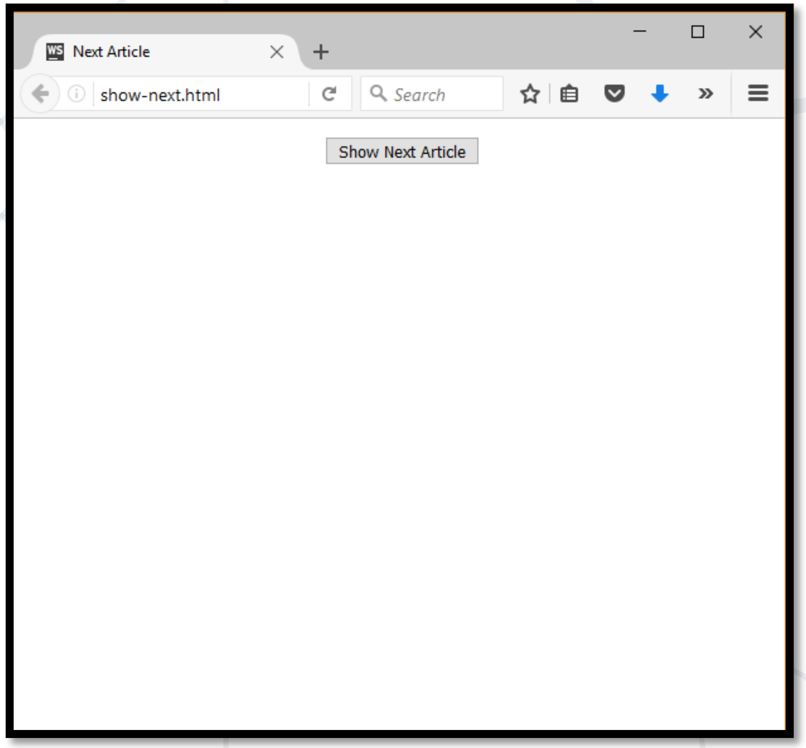
Check your solution here: https://judge.softuni.bg/Contests/330

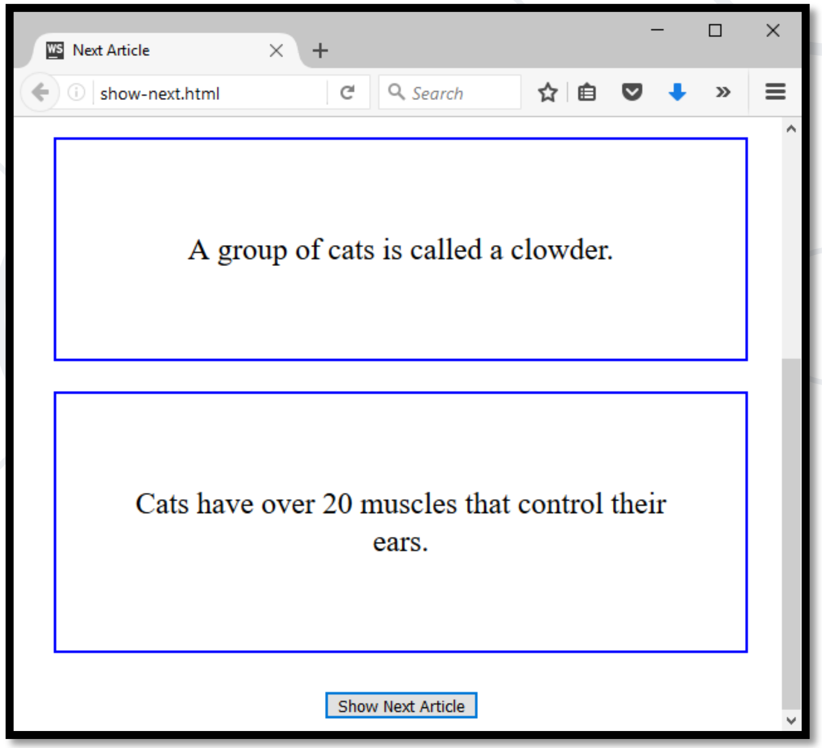
Changing the Context: Bind

```
let maria = {
  name: "Maria",
  hello: function(thing) {
    console.log(this.name + " says hello " + thing);
  }
}
let ivan = { name: 'Ivan' };
let helloIvan = maria.hello.bind(ivan);
maria.hello("world"); // Maria says hello world
helloIvan("from me"); // Ivan says hello from me
```

Problem: Next Article

- Initialize **closure** with array of strings
- When "Show Next" is **clicked**, remove first element from array and **display** it inside an article





A group of cats is called a clowder.

Cats have over 20 muscles that control their ears.

Problem: Next Article (2)

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <title>Next Article</title>
  <style>div{width:600px; text-align: center; font-size: 1.5em} article{border: 2
px solid blue; padding: 2em; margin: 1em}</style>
  <script src="https://code.jquery.com/jquery-3.1.1.min.js" integrity="sha256-hVV
nYaiADRTO2PzUGmuLJr8BLUSjGIZsDYGmIJLv2b8=" crossorigin="anonymous"></script>
  <script src="next-article.js"></script>
</head>
<body>
  <div id="content"></div>
  <div><button onclick="showNext()">Show Next Article</button></div>
</body>
</html>
```

Problem: Next Article (2)

next-article.js

```
function getArticleGenerator(articles) {
    // TODO
}

let articles = [
  "Cats are the most popular pet in the United States: There are 88 million pet cats and 74 million dogs.",
  "A group of cats is called a clowder.",
  "Cats have over 20 muscles that control their ears.",
  "A cat has been mayor of Talkeetna, Alaska, for 15 years. His name is Stubbs.",
  "The world's largest cat measured 48.5 inches long."
];
let showNext = getArticleGenerator(articles);
```

Solution: Next Article

```
function getArticleGenerator(articles) {
  let contentHolder = $('#content');

  return function () {
    if (articles.length > 0) {
      let article = $('<article>');
      article.append($(`<p>${articles.shift()}</p>`));
      contentHolder.append(article);
    }
  }
}
```

Live Exercises

Summary

- In JS functions are objects (first-class functions)
- IIFE is immediately-invoked anonymous function
  - Encapsulates JS code + data (state)
- The **function context** "**this**" depends on how the function is invoked
  - Through object, as event-handler, inner function

Questions?

SoftUni

Software University

SoftUni Svetlina

SoftUni Creative

SoftUni Digital

SoftUni Foundation

SoftUni Kids

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)

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

SoftUni Foundation

License