COURSEWARE

Professional Skills Agile Fundamentals Jira Git **Databases Introduction** Java Beginner Maven Testing (Foundation) Java Intermediate HTML CSS Javascript What is JavaScript Getting started with JS Variables Data types ASI 0 Strict mode Iteration Conditionals with Truthy / Falsey Objects, Arrays + JSON Structuring JS Code Destructuring Scope Functions, function expressions and arrow functions The ECMAScript 6 Specification OOP in JavaScript **Best Practices** Closures Callbacks and Promises Cookies Hoisting Prototypes **Query Parameters**

Higher Order Functions

Closures

Contents

- Overview
- <u>Tutorial</u>
 - Closure Example 1
 - Getters and Setters with closures
- Exercises

Overview

Closure controls what is and isn't in scope in a particular function along with which variables are shared between functions in the same containing scope.

A closure is a combination of functions bundled together with reference to its surrounding state.

Lets break that down further...

Closures give you access to an outer function's scope from an inner function. To use it, we must define a function inside another function and expose it. Exposing is done by either returning it or passing it to another function.

The inner function will have access to the variables in the outer function scope, even after it has returned its value.

Closures are important for data privacy, the enclosed variables are only available to the functions that are encapsulated together within the outer function.

Let's have a look at an example...

Tutorial

Closure Example 1

```
// Closures have access to outer function's variables even after the outer
function returns
let celebrityName = (firstName) => {
 let nameIntro = "Celebrity is ";
 // Inner function has access to outer function's variables, including
parameters
 let lastName = (surname) => {
   return `${nameIntro} ${firstName} ${surname}`;
 }
 return lastName;
}
let mjName = celebrityName("Michael"); // celebrityName outer function has
returned
console.log(mjName);
// Closure (lastName) is called after outer function has returned
// Yet, closure still has access to outer function's vars and params
let mjNameFull = mjName("Jackson");
console.log(mjNameFull); // Celebrity is Michael Jackson
```

1. Function celebrityName() takes in a firstName parameter

Web Storage DOM Manipulation Handling Events and Timed Events Asynchronous Programming HTTP-Requests **XMLHttpRequests** Fetch API Spring Boot Selenium Sonarqube Advanced Testing (Theory) Cucumber MongoDB **Express NodeJS** React **Express-Testing** Networking Security Cloud Fundamentals **AWS Foundations AWS Intermediate** Linux **DevOps** Jenkins Introduction Jenkins Pipeline Markdown

IDE Cheatsheet

- 2. Local variable nameIntro is declared inside celebrityName();
- 3. Create lastName() function inside ceblebrityName() this is a closure
- 4. Return lastName from celebrityName()
- 5. mjName is declared and initialised to celebrityName('Michael')
- 6. When we console.log(mjName) we get the function lastName() returned
- 7. Declare mjNameFull as a variable and initialise it to mjName('Jackson')
- 8. When logged we get the final value "Celebrity is Michael Jackson"

Getters and Setters with closures

```
let celebrityID = () => {
 let celebrityID = 999;
 return {
   getID: function () {
     return celebrityID;
   setID: function (newID) {
     celebrityID = newID;
   },
 };
}
let mjID = celebrityID();
let returnedID = mjID.getID();
console.log(`The initial value of getID() is ${returnedID}`); // 999
mjID.setID(567);
returnedID = mjID.getID();
console.log(`The value of getID now is ${returnedID}`); // 567
```

- 1. Inside the function celebrityID() we have a local variable initialised to 999
- 2. When this function is called it returns two other functions:
 - getId() which returns the value of celebrityID variable
 - setId(newID) which sets the current celebrityID value with the value passed in through the paramater of the function - newID
- 3. Outside the function, a variable mjID is declared and initalised with the value of the function celebrityId() now we can access the functions within.
- 4. Retrieve the current value of ID using mjId.getID();
- 5. To update the value of the ID use the method mjID.setID(< number >);
- 6. To see if the value has been updated use the method mjID.getID()
- 7. We can check the value with a simple console.log() statement.

Exercises

- 1. Create a simple multiplication closure function which meets the following criteria:
 - Outer function takes a single parameter as argument
 - Outer function returns another function
 - Inner function takes in a single parameter as argument
 - Inner function computes multiplication on the outer function parameter and inner function parameter
 - ► Solution
- 2. Create a Person function using getters and setters for the property first name.
 - ► Solution
- 3. Create a function that increases and decreases the value of a counter through the use of functions and closures.
 - ▶ Solution 1
 - ▶ Solution 2