COURSEWARE

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

Higher Order Functions

Callbacks and Promises

Contents

- Overview
 - What is a promise
 - what is a Callback
 - Why Do We Need Callbacks
- <u>Tutorial</u>
 - Promises
 - Callbacks
 - Callbacks in a nutshell
- Exercises

Overview

In this module we will be exploring Callbacks and promises within JavaScript.

What is a promise

"A promise is a placeholder for some data that will be available: immediately, some time in the future or possibly not at all."

When data is needed and is not potentially available straight away we may wait for the data to come through but we also need a way to execute code when data is available or deal with the fact it will never be available.

This is the job of promises.

A promise can be in one of three states:

- Pending Hasn't been fulfilled or rejected yet
- Fulfilled/ Success The action relating to the promise succeeded
- Rejected/ Failed The action relating to the promise failed

There are two further methods that can be used in conjunction with promises for further functionality.

- .then() if successful what should happen next
- .catch() if failed/rejected what should happen next

what is a Callback

A callback function is a function passed into another function as an argument, which is then invoked inside the outer function to complete some kind of routine or action.

Why Do We Need Callbacks

JavaScript is an event driven language. This means JavaScript will keep executing while listening for other events as opposed to waiting for a response before moving on.

Tutorial

Promises

How to create a promise:

```
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

IDE Cheatsheet

Markdown

```
//we create a new promise
let newProm = new Promise((resolve, reject)=>{
    let a = 1+0;
    if (a==2){
        //we say what happens in the success outcome
        resolve("Success");
    }else{
        //we say what happens in the reject outcome
        reject("Failed");
})
//we then pass the value from resolve or reject and we set it to message.
//.then() is executed if Fulfilled/successful.
newProm.then((message)=>{
    console.log(`This is in the then block and the status is: ${message}`);
    //.catch() is executed if rejected/failed.
}).catch((message)=>{
    console.log(`This is in the catch block and the status is: ${message}`);
//.then can be chained and will execute regardless of the outcome.
}).then(()=>{
    console.log("I will take place regardless of what happened before.");
})
```

Callbacks

How to create a call back

```
const greeting = (name) => {

    // Creates an alert(pop-up box) which says hello and appends the value of the variable `name`
    alert(`Hello ${name}`);
}

const processUserInput = (callback) => {

    // A simple user input box appears on the browser that take a value and assigns it to `name`
    let name = prompt('Please enter your name.');

    // Then pass the variable `name` to the callback function as parameter callback(name);
}

//Call the function `processUserInput()` and then pass `greeting` as a parameter.
processUserInput(greeting);
```

Callbacks in a nutshell

A callback is a function that takes in another function as parameter.

Exercises

- Create a callback function that asks for a user to enter a value, then increase that value by 10 through another function.
 - ▶ Solution