CoGrammar

Welcome to this session:
Functional Programming II (Higher-Order Functions and Callbacks)

The session will start shortly...

Questions? Drop them in the chat.
We'll have dedicated moderators answering questions.

Safeguarding & Welfare

We are committed to all our students and staff feeling safe and happy; we want to make sure there is always someone you can turn to if you are worried about anything.

If you are feeling upset or unsafe, are worried about a friend, student or family member, or you feel like something isn't right, speak to our safeguarding team:

Ian Wyles
Designated Safeguarding Lead

Simone Botes

Nurhaan Snyman

Rafiq Manan

Ronald Munodawafa

Tevin Pitts

Scan to report a safeguarding concern

or email the Designated Safeguarding Lead:
Ian Wyles
safeguarding@hyperiondev.com

CoGrammar HyperionDev

Skills Bootcamp Cloud Web Development

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Skills Bootcamp Cloud Web Development

- For all **non-academic questions**, please submit a query: ***www.hyperiondev.com/support***
- **Report a safeguarding incident: *www.hyperiondev.com/safeguardreporting***
- We would love your feedback on lectures: Feedback on Lectures
- If you are hearing impaired, please kindly use your computer's function through Google chrome to enable captions.

Stay Safe Series:

Mastering Online Safety One week at a Time

While the digital world can be a wonderful place to make education and learning accessible to all, it is unfortunately also a space where harmful threats like online radicalization, extremist propaganda, phishing scams, online blackmail and hackers can flourish.

As a component of this BootCamp the *Stay Safe Series* will guide you through essential measures in order to protect yourself & your community from online dangers, whether they target your privacy, personal information or even attempt to manipulate your beliefs.

Digital Decorum:

The Importance of Respect in Online Spaces

- Diverse Audiences.
- Mental Health.
- Community Building.
- Conflict Prevention.
- Professionalism

In JavaScript, A function can return a value. (True or False)

A. True
B. False

In functional programming, functions can be assigned to variables.
(True or False)

A. True
B. False

CoGrammar

Learning Outcomes

- Explain the concept of higher-order functions and their uses in JavaScript.
- Create and utilize callback functions to handle asynchronous operations.
- Integrate built-in higher-order functions like map, filter, and reduce in functional programming tasks.

Lecture Overview

- Introduction to functional programming
- Built-in and custom Higher-Order Functions (HOF)
- Callbacks in JavaScript

INTRODUCTION TO FUNCTIONAL PROGRAMMING

- Functional programming is a way of writing code where functions are treated like any other value.
- This means they can be assigned to variables, passed as arguments, and returned from other functions.
- **Higher-order functions (HOF)** and **callbacks** are key examples of this concept in action.

INTRODUCTION TO FUNCTIONAL PROGRAMMING

- First-class functions:
  - JavaScript treats functions as first-class citizens.
  - Functions are special values that can be used like any other value.
  - This means you can assign them to **variables**, pass them **as arguments** to other functions, and even **return** them from functions.

INTRODUCTION TO FUNCTIONAL PROGRAMMING

- Let us look at the following example:

```
// This is a higher-order function (HOF)
function higherOrder() {
  // Regular function returns the greeting "Hello, World!"
  function greet() {
    return "Hello, World!";
  }

  // Return the function itself
  return greet;
}

// Call the HOF and store the returned function in a variable
let useGreeting = higherOrder();

// Call the function stored in useGreeting and log the output to the console
console.log(useGreeting()); // Output: Hello, World!
```

BUILT IN AND CUSTOM HOF

- Built-in higher-order functions:
  - JavaScript actually already has a bunch of built-in higher-order functions that make our lives easier.
  - The array class has built-in methods that take functions as arguments, such as the map(), filter(), and reduce() methods.

BUILT IN AND CUSTOM HOF

- Problem
  - Let's look at an example of how the **map()** method can be used. Let's say we have an array of numbers. We want to create a new array that will contain the doubled values of the first one. Let's see how we can solve this problem with and without a **higher-order function** and how to solve the same problem with a **higher-order function**.

BUILT IN AND CUSTOM HOF

- Without an HOF:

```
function doubleAllValues() {
  // Define the first array to store the single values
  const firstArray = [1, 2, 3, 4];

  // Define the second array (empty array) to store the doubled values
  const secondArray = [];

  // Loop through the length of the first array
  for (let i = 0; i < firstArray.length; i++) {
    /* Multiply each value in the first array by 2 and push it into the
    second array */
    secondArray.push(firstArray[i] * 2);
  }

  // Return the doubled values stored in the second array
  return secondArray;
}

// Call the function and log the results in the console
console.log(doubleAllValues()); // Output: [ 2, 4, 6, 8 ]
```

BUILT IN AND CUSTOM HOF

- With an HOF:

```
function doubleAllValues() {
  // Define the first array to store the single values
  const firstArray = [1, 2, 3, 4];

  /* Arrow function takes the argument "item". Each item is an element in
the first array that is multiplied by 2 */
  const multiplyByTwo = (item) => item * 2;

  /* Map each element in the first array by applying the multiplyByTwo
function. Define the second array to store the doubled values. */
  const secondArray = firstArray.map(multiplyByTwo);

  // Return the new array
  return secondArray;
}
// Call the function and log the results in the console
console.log(doubleAllValues()); // Output: [ 2, 4, 6, 8 ]
```

Higher Order Functions

Higher order functions are functions that can accept other functions as arguments or return functions as results.

- The **map()** function **applies** a provided function to each element of an array and returns a new array with the results.

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map(num => num * 2);
console.log(doubled);
```

Higher Order Functions

Higher order functions are functions that can accept other functions as arguments or return functions as results.

- The **filter()** function creates a new array with all elements that pass the test implemented by the provided function.

```
const scores = [80, 90, 60, 45, 75];
const passed = scores.filter(score => score >= 70);
console.log(passed);
```

Higher Order Functions

Higher order functions are functions that can accept other functions as arguments or return functions as results.

- The **reduce()** function executes a **reducer** function on each element of the array, resulting in a single output value.

```
const numbers = [1, 2, 3, 4, 5];
const sum = numbers.reduce((acc, num) => acc + num, 0);
console.log(sum);
```

CoGrammar

CALLBACKS IN JAVASCRIPT

- 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.
- The callback will never run until the encompassing function executes it when the time comes.
- The callback is at the mercy of the function receiving it.

CALLBACKS IN JAVASCRIPT

- Why do we need callbacks in JavaScript:
  - JavaScript is an event-driven language. This means that instead of waiting for a response before moving on, JavaScript will keep executing while listening for other events.
  - JavaScript code runs from top to bottom, meaning that code runs sequentially. There are times, however, when we do not want code to run sequentially, but rather have certain code executed only when another task is completed or an event takes place. This is called asynchronous execution.

CALLBACKS IN JAVASCRIPT

- Let's look at the normal Javascript code below:

```
// Define two functions that execute sequentially
function first() {
  console.log(1);
}

function second() {
  console.log(2);
}

// Call the functions
first();
second();
// Output:
// 1
// 2
```

CALLBACKS IN JAVASCRIPT

- The output of the code changes when we add a delay time callback function:

```
// Define first function with a delay
function first() {
  // Use an anonymous function as a callback for the setTimeout
  setTimeout(function () {
    console.log(1); // Body of the anonymous function
  }, 5000); // Delay is 5000ms for the second argument
}

// Define a second function without a delay
function second() {
  console.log(2);
}

// Call the functions in the same order as before
first();
second();
// Output:
// 2
// 1
```

CALLBACKS IN JAVASCRIPT

- Ways of creating and using callback functions
  - Anonymous functions

```
/* This is an anonymous function (function without a name) created and
passed as an argument at the same time */
setTimeout(function () {
  console.log("This line is returned after 5000ms");
}, 5000);
```

CALLBACKS IN JAVASCRIPT

- Ways of creating and using callback functions
  - Arrow functions

```
// This is an arrow function expression used as an anonymous callback
function
setTimeout(() => {
  console.log("This line is returned after 5000ms");
}, 5000);
```

CALLBACKS IN JAVASCRIPT

- Ways of creating and using callback functions
  - Defined functions passed as an argument

```
// Assign a function to a variable
let callback = function () {
  console.log("This line is returned after 5000ms");
};

// Pass the callback variable as the callback function to setTimeout
setTimeout(callback, 5000);
```

JavaScript has a bunch of built-in Higher-Order Functions.
(True or False)

A. True
B. False

CoGrammar

A Callback function is at the mercy of the function receiving it.
(True or False)

A. True
B. False

Which of the following Fundamental British Values do you think best aligns with the concept of Higher-Order Functions in JavaScript?

A. Respect for Others.
B. Tolerance of Different Faiths and Beliefs.
C. Democracy.
D. Rule of Law.
E. Individual Liberty.

Let's take a break

CoGrammar

Questions and Answers

CoGrammar

Thank you for attending

CoGrammar

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education