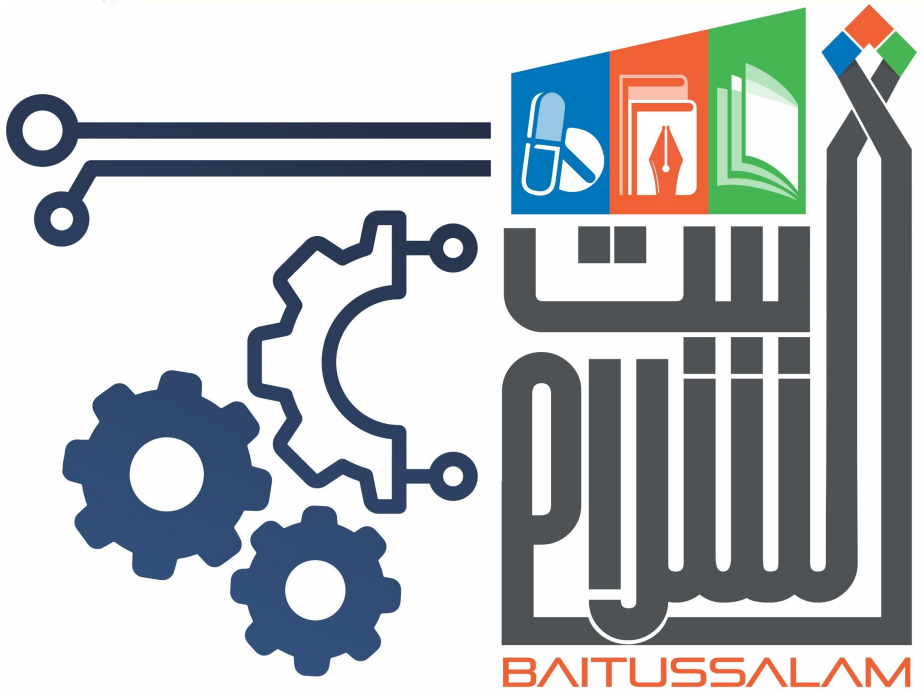

BAITUSSALAM
—TECH PARK—

Class Agenda

BAITUSSALAM
—TECH PARK—

Arrow Functions, Callback, higher order functions, Map and Filter

Functions in JavaScript

BAITUSSALAM
—TECH PARK—

functions.js > ...

```
1   // Function Declaration
2   function multiply(a, b) {
3     return a * b
4   }
5
6   // Function Expression
7   const divide = function (a, b) {
8     return a / b
9   }
10
11  // Arrow Function
12  const subtract = (a, b) => a - b
13
14  console.log(multiply(6, 3)) // 18
15  console.log(divide(6, 3)) // 2
16  console.log(subtract(6, 3)) // 3
17
```

Different ways of defining a function in JavaScript

Arrow Functions

- Arrow functions provide a more concise way to write functions
- When the function body contains only a single expression, we can omit the return keyword and the curly braces
- Arrow functions do not have their own this binding

Arrow to Regular Function

BAITUSSALAM
—TECH PARK—

Convert a Regular Function to an Arrow in three steps

1. Replace the function keyword with the variable keyword const
2. Add the = symbol after the function name and before the parentheses
3. Add the => symbol after the parentheses

Arrow Functions Exercises

BAITUSSALAM
—TECH PARK—

Convert Regular Function to Arrow Functions.

Get functions file from Slack Group

Callback Functions

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.

functions.js > result

```
1   function greet(name, callback) {
2     console.log('Hello, ' + name)
3     callback()
4   }
5
6   function sayGoodbye() {
7     console.log('Goodbye!')
8   }
9
10  greet('Alice', sayGoodbye)
11
```

Callback function Exercise #1

BAITUSSALAM
—TECH PARK—

Create a higher order function `calculate` that takes two numbers and a callback function. The callback function should perform an operation (addition, subtraction, division and multiplication) on the two numbers.

```
function calculate(a, b, callback) {
    // Implement the function
}

// Example usage:
calculate(5, 3, yourFunction);
```

Higher Order Functions

- Higher-order functions are functions that take other functions as arguments or return functions as their result
- They allow for more abstract and concise code

Map

map is a method that creates a new array by calling a provided function on every element in the calling array.

map Syntax and Example

```
const numbers = [1, 2, 3, 4, 5];
const doubled = numbers.map(number => number * 2);
console.log(doubled); // [2, 4, 6, 8, 10]
```

Map Exercise #1

BAITUSSALAM
—TECH PARK—

Task: Given an array of student objects with `firstName` and `lastName`, create a new array containing the full names.

```
const students = [
    { firstName: 'Alice', lastName: 'Smith' },
    { firstName: 'Bob', lastName: 'Johnson' },
    { firstName: 'Charlie', lastName: 'Williams' }
];

// Expected output:
// ["Alice Smith", "Bob Johnson", "Charlie Williams"]
```

Filter

filter is a method that creates a new array with all elements that pass the test implemented by the provided function.

filter Syntax and Example

```
const numbers = [1, 2, 3, 4, 5];
const evens = numbers.filter(number => number % 2 === 0);
console.log(evens); // [2, 4]
```

Filter Exercise #1

Given an array of numbers, create a new array containing only the numbers greater than 10.

```
const numbers = [5, 12, 13, 3, 7];
```

Map and Filter Combine

Task: Given an array of objects representing students, filter out only those students who passed (scored 60 or above) and create a new array containing their names in uppercase.

Map and Filter Exercise Data

```
const students = [
    { name: 'Alice', score: 52 },
    { name: 'Bob', score: 67 },
    { name: 'Charlie', score: 80 },
    { name: 'David', score: 45 }
];
```

Array and Object Destructuring

BAITUSSALAM
—TECH PARK—

functions.js > ...

```
1   // Array Destructing
2   const coordinates = [10, 20, 30]
3
4   // Destructure the array into variables x, y, z
5   const [x, y, z] = coordinates
6   console.log(x, y, z) // Output: 10 20 30
7
8   // Object Destructuring
9   const person = { firstName: 'Alice', lastName: 'Smith' }
10
11  const { firstName: first, lastName: last } = person
12  console.log(first, last) // Output: Alice Smith
```

Destructuring is a convenient way of extracting multiple values from arrays or objects and assigning them to variables.

Array and Object Destructuring

If an element is undefined or missing in the array, the **default value** is used.

Variables first and last are aliases for **firstName** and **lastName** properties of **person**.

```
14   const numbers = [1]
15   const [a = 0, b = 2] = numbers
16   console.log(a, b) // Output: 1 2
17
18   const person = { firstName: 'Alice', lastName: 'Smith' }
19   const { firstName: first, lastName: last = 'David' } = person
20   console.log(first, last) // Output: Alice Smith
21
```

Destructuring Exercise

BAITUSSALAM
—TECH PARK—

Task: Given an object `car` with properties `make` and `model`, destructure it into `make` and `model` variables.

```
const car = { make: 'Toyota', model: 'Camry' };
```

Task: Extract the values into variables `name`, `age`, and `city`.

```
const data = ['Alice', 30, 'New York']
```

Quiz App Project

BAITUSSALAM
—TECH PARK—

Make this Quiz app functional – You are provided HTML and CSS code

Finance Tracker Project

BAITUSSALAM
—TECH PARK—

Make this todo app functional – You are provided HTML and CSS code

The End

BAITUSSALAM
—TECH PARK—