Exercise: Functional Programming in JavaScript

Objective:

The goal of this exercise is to deepen your understanding of functional programming concepts like higher-order functions (`map`, `filter`, `reduce`), immutability, and pure functions.

Part 1: Higher-Order Functions

Map Function: Create a function named `doubleElements` that takes an array of numbers as an argument and returns a new array with each element doubled.

```
// Input: [1, 2, 3]
// Output: [2, 4, 6]
```

Filter Function: Create a function named `filterEven` that takes an array of numbers and returns a new array containing only even numbers.

```
// Input: [1, 2, 3, 4]
// Output: [2, 4]
```

Reduce Function: Create a function named `sumArray` that takes an array of numbers and returns the sum of all elements.

```
// Input: [1, 2, 3, 4]
// Output: 10
```

Part 2: Immutability

Absolutely! Below is an exercise that focuses exclusively on immutability, steering clear of using objects.

Insert Value: Write a function called `insertValueAtIndex` that takes an array, an index, and a value as parameters. This function should return a new array with the value inserted at the given index.

```
// Input: [1, 2, 3], 1, 4
// Output: [1, 4, 2, 3]
```

Part 3: Pure Functions

Pure Function: Create a function named `calculateArea` that takes the radius of a circle as an argument and returns its area. Make sure the function has no side-effects.

```
// Input: 5
// Output: 78.54
```