Advanced Functions -Exercise

1. Using bind to Set this

Create an object with a method that logs a message using this.name. Create a second object and use bind to set the method's this context to the second object. Log the result.

Starter code

```
const obj1 = {
  name: 'Alice',
  greet: function() {
    console.log(`Hello, my name is ${this.name}`);
  }
};

const obj2 = {
  name: 'Bob'
};
```

2. Using call to Chain Constructors.

Create two constructor functions. In the first constructor, use call to chain the second constructor, passing in arguments from the first constructor.

Starter code

```
function Person(name) {
  this.name = name;
}

function Employee(name, job) {
// ....
}

const emp = new Employee('Alice', 'Engineer');
console.log(emp); // Output: Employee { name: 'Alice', job: 'Engineer' }
```

3. Using bind for Partial Application.

Create a function that multiplies two numbers. Use bind to create a new function that always multiplies by a specific number.

Starter code

```
function multiply(a, b) {
  return a * b;
}
```

```
// …
console.log(multiplyByFive(3)); // Output: 15
```

4. Function Composition.

Create two functions, one that doubles a number and one that squares a number. Create a third function that composes these two functions to first double and then square a number.

```
//…
const doubleThenSquare = compose(double, square);

console.log(doubleThenSquare(3)); // Output: 36
```

5. Closure for Data Privacy

Create a function that returns an object with two methods: one to get a private variable and one to set it. Use a closure to maintain the private variable.

```
//…
const counter = createPrivateCounter();
counter.increment();
console.log(counter.getCount()); // Output: 1
```

6. Curry a Function

Create a function that takes three arguments and returns their product. Curry this function so it can be called with one argument at a time.

```
console.log(multiply(2)(3)(4)); // Output: 24
```

7. Partial Application with Closures

Create a function that takes four arguments and returns their sum. Create a partially applied version of this function that always adds 5 to the sum of three other numbers.

```
const addFive = partialSum(5);
console.log(addFive(1, 2, 3)); // Output: 11
```

8. Function Chaining

Create an object with methods that manipulate a string (e.g., toUpperCase, toLowerCase). Ensure the methods return this so the methods can be chained together.

```
stringManipulator.setValue('Hello')
  .toUpperCase()
  .print()
  .toLowerCase()
  .print();

// HELLO
//  hello
```

9. Fibonacci

Write a JS function that when called, returns the next Fibonacci number, starting at 0, 1. Use a closure to keep the current number.

```
let fibonacci = getFibonacci();
console.log(fibonacci()); // 1
console.log(fibonacci()); // 1
console.log(fibonacci()); // 2
console.log(fibonacci()); // 3
console.log(fibonacci()); // 5
console.log(fibonacci()); // 8
console.log(fibonacci()); // 13
console.log(fibonacci()); // 21
```

10. TODO List *

Write JavaScript to add new tasks to the list when the button is clicked.

Each task should have a "Remove" button to delete the task from the list.

NB: Check how to add Event Listener to your remove button

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>To-Do List</title>
  <style>
body {
  font-family: Arial, sans-serif;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  margin: 0;
}

.container {
  text-align: center;
  width: 300px;
}

#todo-input {
  width: 80%;
  padding: 10px;
  margin-bottom: 10px;
}

#add-button {
  padding: 10px 20px;
}

#todo-list {
  list-style-type: none;
  padding: 0;
}

#todo-list li {
  background: #f0f0f0;
  margin: 5px 0;
  padding: 10px;
  display: flex;
  justify-content: space-between;
}

.remove-button {
  background: red;
  color: white;
  border: none;
  cursor: pointer;
}
  </style>
</head>
<body>
  <div class="container">
    <h1>To-Do List</h1>
    <input type="text" id="todo-input" placeholder="Add a new task">
    <button id="add-button">Add</button>
    <ul id="todo-list"></ul>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

```
//script.js
document.getElementById('add-button').addEventListener('click', function() {
  const input = document.getElementById('todo-input');
  const task = input.value.trim();

    // … your code … /
  }
});
```