**LAB: functional programming concepts**

**Introduction:**

In this lab, you will practice **functional programming concepts** by creating functions in JavaScript that generate styled console log messages. You will write functions, use template literals, and conditionally log messages to the console based on inputs.

**Goal:**

The goal of this lab is to develop a strong understanding of **functional programming concepts** in JavaScript. By building styled console message generators, learners will practice writing modular, reusable, and readable functions while exploring JavaScript's string manipulation and conditional logic capabilities.

**Objectives:**

* **Create and use functions in JavaScript** to perform specific tasks, ensuring modular and reusable code.
* **Dynamically style console messages** using template literals and the %c format in console.log.
* **Apply conditional logic with if-else statements** to handle different scenarios within functions.

**Learner Instructions:**

**Lab Tasks:**

**Task 1: Create a Styled Console Log Message Generator**

**Objective:** Code a function named **consoleStyler** that logs a styled console message.

**Steps:**

**Step 1:** Open the **functionalprogramming.js** file present under the **LEARN** folder. This is where you will write your functional program.

**Step 2:** Define a function **consoleStyler** that accepts the following four parameters:

* **color** (text color)
* **background** (background color)
* **fontSize** (size of the font)
* **txt** (message to display)

**Step 3:** Inside the function body:

* Create a variable **message** and assign it:

1

"%c" + txt;

* Create a **variable** message and assign it:

1

`color: ${color};`

* Update the **style** variable by appending the background and font size using the **+=** operator:

1

2

style += `background: ${background};`;

style += `font-size: ${fontSize};`;

* Log the **message** and **style** variables to the console:

1

console.log(message, style);

**Task 2: Create a Celebratory Message Generator**

**Objective:** Code a function named **celebrateStyler** that logs a celebratory message based on a given reason.

**Steps:**

**Step 1:** Define a function **celebrateStyler** that accepts a single parameter:

* **reason** (a string indicating the reason for celebration)

**Step 2:** Inside the function body:

* Create a variable **fontStyle** and assign it:

1

"color: tomato; font-size: 50px";

* Use an **if-else** block to log messages based on the value of **reason**.
* If **reason** equals **"birthday"**, log:

1

console.log(`%cHappy birthday`, fontStyle);

* Else if **reason** equals **"champions"**, log:

1

console.log(`%cCongrats on the title!`, fontStyle);

* Otherwise, log:

1

console.log(message, style);

**Task 3: Invoke the Functions**

**Objective:** Call the **consoleStyler** and **celebrateStyler** functions to verify their functionality.

**Steps:**

**Step 1:** Call the **consoleStyler** function with the following arguments:

1

consoleStyler('#1d5c63', '#ede6db', '40px', 'Congrats!');

**Step 2:** Call the **celebrateStyler** function with the following argument:

1

celebrateStyler('birthday');

**Task 4: Combine the Functions into a New Function**

**Objective:** Code a new function named styleAndCelebrate to combine the functionality of both consoleStyler and celebrateStyler.

**Steps:**

**Step 1:** Define a function **styleAndCelebrate** that accepts the following parameters:

* **color**
* **background**
* **fontSize**
* **txt**
* **reason**

**Step 2:** Inside the function body:

* Call **consoleStyler** with the first four parameters.
* Call **celebrateStyler** with the last parameter.

**Step 3:** Invoke the **styleAndCelebrate** function with these arguments:

1

styleAndCelebrate('ef7c8e', 'fae8e0', '30px', 'You made it!', 'champions');

**Step 4:** After successfully modifying the **functionalprogramming.js** file, navigate to **File** > **Save** to save changes in the file.

**To execute your modified JavaScript code and verify the function is working:**

* Click the **Run Code** button in the top-right corner (shaped like a triangular "Play" button).

**Explanation:**

* **Task 1:** The **consoleStyler** function dynamically generates a styled message using template literals and logs it to the console.
* **Task 2:** The **celebrateStyler** function logs different celebratory messages based on the **reason** parameter.
* **Task 3:** Both functions are invoked with example arguments to demonstrate their functionality.
* **Task 4:** The **styleAndCelebrate** function combines the two earlier functions, making it modular and reusable. It’s also invoked with sample arguments to show combined behaviour.

**Key Takeaways:**

* Use backticks (**``**) and **${}** to dynamically create strings for console styling. This technique enables customizable and reusable code for styled console messages.
* The **%c** format in **console.log** allows you to apply styles directly to messages in the console, making it useful for debugging or creating visually distinct logs.
* Functions like **consoleStyler** and **celebrateStyler** demonstrate how to encapsulate logic into reusable blocks.
* Use **if-else** blocks to implement different behaviours based on the input. This helps in tailoring output for specific scenarios or user inputs.
* Combining functions, as demonstrated with **styleAndCelebrate**, illustrates how to create higher-level functionality by leveraging simpler, well-defined functions.

**Final Step: Submit Your Code:**

* Go to **File** > **Save** to ensure your work is saved.
* **Submit your assignment**: Click the "**Submit Assignment"** button in the Lab toolbar.
  + Your code will be **autograded** and feedback will be available shortly on the **Grades** tab.