Tutorial 5

SNO	QUESTIONS
1	Write a JavaScript function that takes input from an HTML input field and adds it to a variable. After adding the input value to the variable, the function should print the variable's contents to the console.
2	How can you add an event listener to a button element with the id 'myButton' on a webpage using JavaScript? Ensure that the event listener logs 'Button clicked!' to the console each time the button is clicked.

3	**Develop a web page with the following features:**
	• Write a code to an input field labeled "Enter name" allowing users to input a name.
	• Add a button labeled "Search" to execute the search process.
	• The array `sampleObject` contains two objects, each representing a user with properties for name, age, and city.
	`const sampleObject = [`
	`{`
	`name: "Bishal",`
	`age: 26,`
	`city: "Salyan"`
	`},`
	`{`
	`name: "Shyam",`
	`age: 12,`
	`city: "Kathmandu"`
	`}`
	`];`

- Implement the `searchUser` function to iterate through the `sampleObject` array and return the age of the user if found; otherwise, return "User not found".

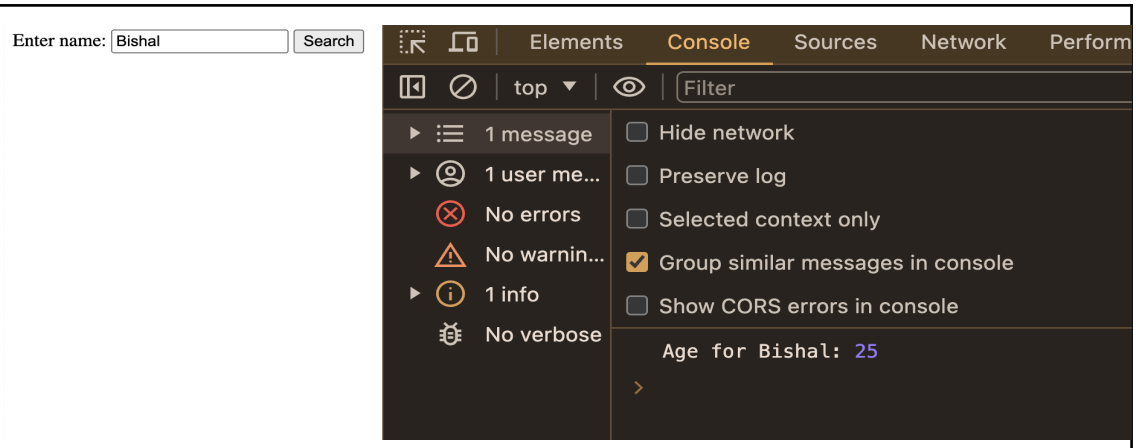
```
function searchUser(name) {
//Write Your Logic here
return userAge;
}
```

- Attach an event listener to the "Search" button to trigger the search process when clicked.

- When the button is clicked, retrieve the value entered in the input field, perform a search for the user by name, and log the user's age to the console along with their name.

```
document.getElementById('searchButton').addEventListener('click',
function() {
// Get the value from the input field
// Perform search for user
// Output the result
});
```

Screenshots:

	
4	Create an HTML page with a list of 5 items. Using JavaScript: ● Select all the list items and change their background color to yellow using a for loop. ● Select all the list items and change their font weight to bold using a forEach loop. ● Select all the list items and wrap them in tags using the map method.
5	Consider a scenario where you have a function `fetchUserData` that simulates fetching user data from a server asynchronously. The function takes two parameters: a user ID and a callback function. The callback function is called once the user data is successfully retrieved, and it receives the user data as its argument. **Instructions:** - Ensure that the `fetchUserData` function simulates an asynchronous operation using `setTimeout` to introduce a delay. - Simulate the user data as an object with properties like `name`, `email`, and `age`. - Use the `console.log` function within the `displayUserData`

callback to log the received user data to the console.

Sample Object:

```
const userData = {
name: "Hari",
email: "hari@gmail.com",
age: 30
};
```

1. Define the `fetchUserData` function as:

```
// Define the fetchUserData function
function fetchUserData(userId, callback) {
// Simulate asynchronous operation with setTimeout
setTimeout(() => {
// Simulate user data as in given sample:
//Also include userid passed as in screenshot below
// Invoke the callback function with the user data
}, 1000); // Simulate a delay of 1 second
}
```

2. Write a callback function named `displayUserData` that logs the received user data to the console.
3. Call the `fetchUserData` function with a sample user ID and the `displayUserData` function as the callback.

```
fetchUserData(123, displayUserData);
```

Sample Output:


{userId: 123, name: 'John Doe', email: 'john@example.com', age: 30}
age: 30
email: "john@example.com"
name: "John Doe"
userId: 123
[[Prototype]]: Object