

# Programming Lab - III

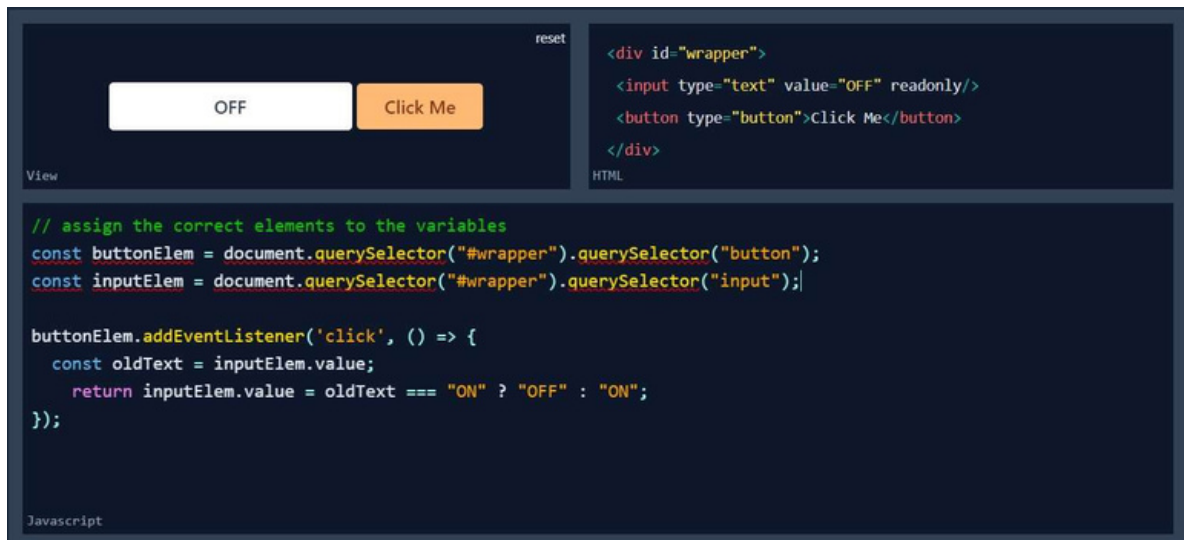
## Assignment No. 4

Name : Shantanu Kantak

PRN : 21510001

Batch : T8

existing code expects the variables 'buttonElem' and 'inputElem' to represent the button and input elements in the example UI. Assign the respective elements to the variables. In this case, the two elements do not have unique identifiers - like for example an id. Instead they are direct descendents of a div element with id 'wrapper'. Use an appropriate selector method! Click the button to verify that the code is working.



The screenshot displays a web application interface and its corresponding code. The interface, shown in the 'View' pane, features a white text input field containing the word 'OFF' and an orange button labeled 'Click Me'. A 'reset' link is visible in the top right corner of the interface area. The 'HTML' pane shows the structure of the UI, consisting of a single `<div id="wrapper">` containing an `<input type="text" value="OFF" readonly/>` and a `<button type="button">Click Me</button>`. The 'Javascript' pane contains the following code:

```
// assign the correct elements to the variables
const buttonElem = document.querySelector("#wrapper").querySelector("button");
const inputElem = document.querySelector("#wrapper").querySelector("input");

buttonElem.addEventListener('click', () => {
  const oldText = inputElem.value;
  return inputElem.value = oldText === "ON" ? "OFF" : "ON";
});
```

In this scenario, we are looking for a list of elements gathered in one variable - rather than only one element. Assign the list items in the view to the variable 'listItems' by using an appropriate selector method. Once you have completed the code below, verify it by hovering over the list items until all items have the value 'ON'

reset

ON

ON

ON

ON

ON

ON

View

```
<ul id="list">
  <li>OFF</li>
  <li>OFF</li>
  <li>OFF</li>
  <li>OFF</li>
  <li>OFF</li>
  <li>OFF</li>
</ul>
```

HTML

```
// assign the correct elements to the variable
const listItems = document.querySelectorAll("li");

const handleHover = (event) => {
  return event.target.innerText = 'ON';
};
if(listItems.length > 1) {
  listItems.forEach(item => item.addEventListener('mouseover', handleHover));
}
```

The Javascript function handleText fills the input field with the words Hello World. But, there is no code to execute this function. Complete the existing code below such that the function is called when the button is clicked. Verify by clicking the button.

reset

Click Me

View

```
<input type="text" id="input" readonly/>
<button type="button" id="button">Click Me</button>
```

HTML

```
const button = document.getElementById('button');
const input = document.getElementById('input');

const handleClick = () => {
  input.value = 'Hello World';
};

// type in your code here
document.getElementById("button").addEventListener("click", handleClick);
```

Javascript

The Javascript function `changeText` changes the text inside the circle. But again, there is no code to execute this function. Complete the existing code below such that the function is called when the cursor moves onto the circle. Verify that your code works by hovering over the circle.



```
const element = document.getElementById('element');

const changeText = () => {
  element.innerText = 'Thanks!';
};

// type in your code here
document.getElementById("element").addEventListener("mouseenter", changeText);
```

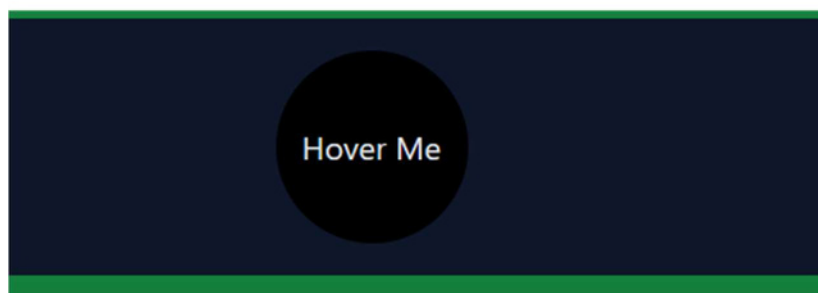
In this scenario we want the color of the circle to change depending on the type of cursor movement. Use the function `toggleColor` to turn the circle orange when the cursor moves onto it. Reuse the same function to turn it black when the cursor leaves it. The tricky part is that you have to call `toggleColor` with different values for the parameter `isEntering`. Verify that your code is working by hovering the circle with the mouse cursor and leaving it again.



```
const element = document.querySelector('#element');

const toggleColor = (isEntering) => {
  element.style.background = isEntering ? 'orange' : 'black';
};

// type in your code here
element.addEventListener('mouseover', () => toggleColor(true));
element.addEventListener('mouseleave', () => toggleColor(false));
```



Remove element from the DOM. Create 2 circles red and green and a button clickme. Place them such a way that red circle hides the green circle. Add the function removeRedCircle to remove the circle with id red from the DOM when clicked on clickme button. Make sure that you really remove the element instead of just hiding it. Code:

```
<!DOCTYPE html>
<html>
<head>
<title>Remove Red Circle</title>
<style>
.container {
display: flex;
height: 100vh;
width: 100vw;
align-items: center;
justify-content: center;

.circle {
width: 100px;
height: 100px;
border-radius: 50%;
text-align: center;
line-height: 100px;
position: absolute;

#red {
background-color: red;
z-index: 2; /* Red circle is on top */

#green {
background-color: green;
z-index: 1;

#button {
position: relative;
top: 120px;

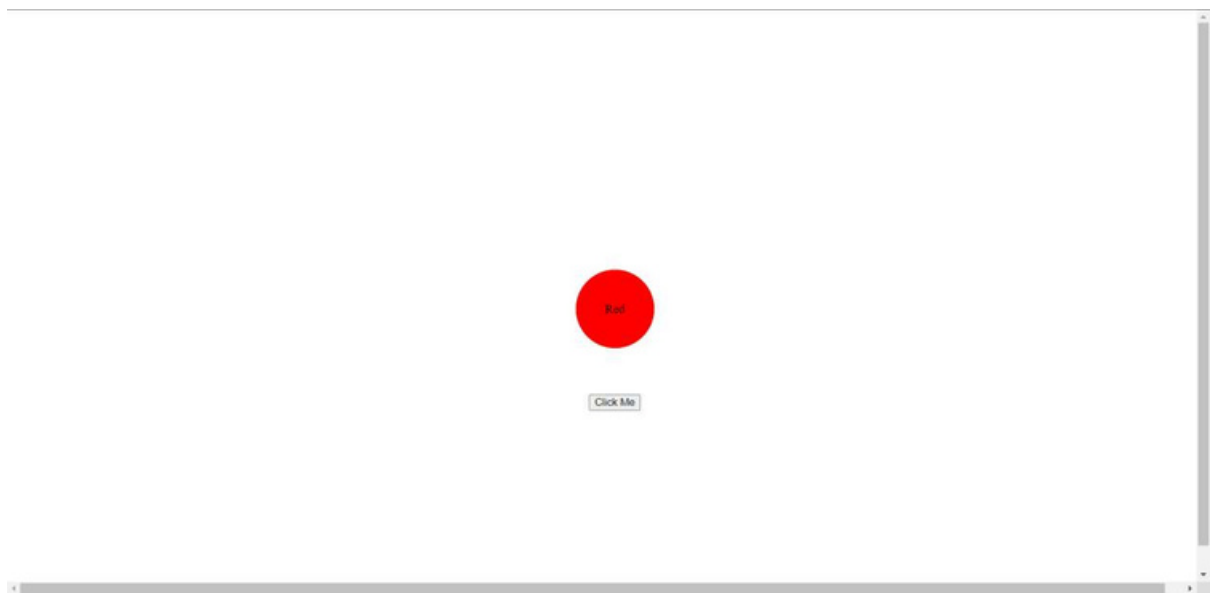
</style>
</head>
<body>
<div class="container">
<div class="circle" id="red">Red</div>
    <div class="circle" id="green">Green</div>
```

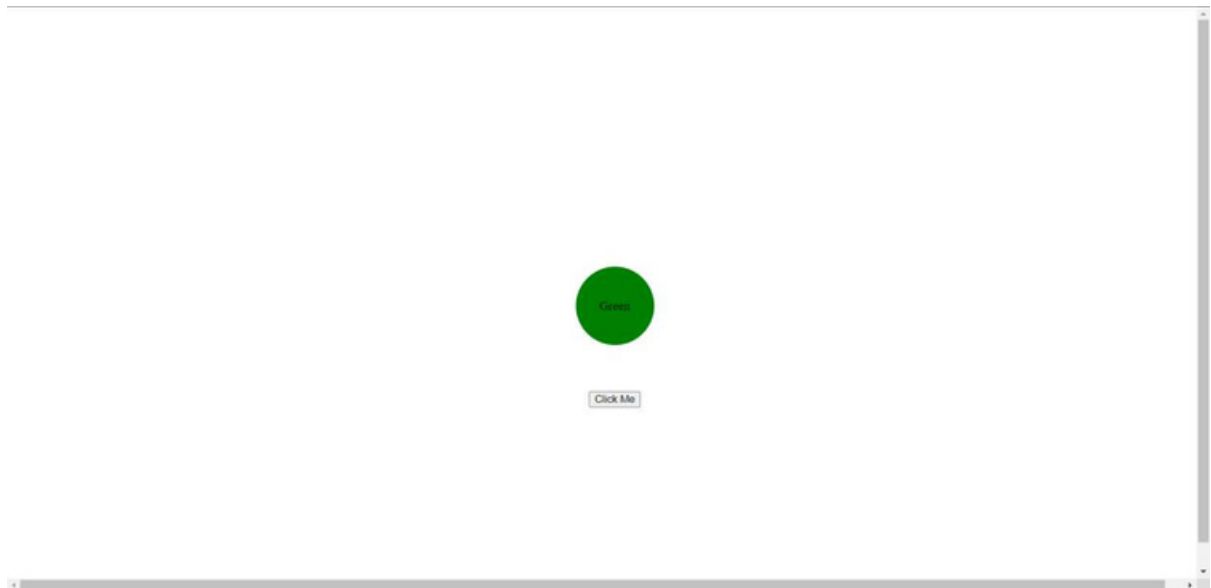
```
<button id="button">Click Me</button>
</div>

<script>
//red circle is removed by this function
function removeRedCircle() {
    const redCircle = document.getElementById("red");
    redCircle.parentNode.removeChild(redCircle);
}

const button = document.getElementById("button");
button.addEventListener("click", removeRedCircle);
</script>
</body>
</html>
```

Live server:





Create JavaScript code to interact with the displayed HTML elements. Create a checkbox and a button. Once you click the button, the checkbox should be checked.

Code:

```
<!DOCTYPE html>
<html>
<head>
<title>Checkbox Interaction</title>
</head>
<body>
<input type="checkbox" id="myCheckbox" />
<button id="myButton">Check the Checkbox</button>

<script>
    const checkbox = document.getElementById("myCheckbox");
    const button = document.getElementById("myButton");

    button.addEventListener("click", () => {
        checkbox.checked = true;
    });
</script>
</body>
</html>
```

Live server:

☐ Check the Checkbox☒ Check the Checkbox

Create 3 textboxes and a button. First 2 checkboxes contain first name and last name respectively. When the button is clicked, combine the names of the first two input fields. Insert the full name in the third input field (textbox). Check if your code still works if you change the first or last name.

Code:

```
<!DOCTYPE html>
<html>
<head>
<title>Combine Names</title>
</head>
<body>
<label for="firstName">First Name:</label>
<input type="text" id="firstName" /><br />

<label for="lastName">Last Name:</label>
<input type="text" id="lastName" /><br />

<label for="fullName">Full Name:</label>
<input type="text" id="fullName" readonly /><br />

<button id="combineButton">Combine Names</button>

<script>
// Get references to the input fields and the button
const firstNameInput = document.getElementById("firstName");
const lastNameInput = document.getElementById("lastName");
const fullNameInput = document.getElementById("fullName");
const combineButton = document.getElementById("combineButton");
```

```
        // Add an event listener to the button
        combineButton.addEventListener("click", function () {
// Combine the first name and last name
const firstName = firstNameInput.value;
const lastName = lastNameInput.value;
const fullName = `${firstName} ${lastName}`;

// Insert the full name in the third input field
fullNameInput.value = fullName;
});
</script>
</body>
</html>
```

Create three buttons. One button displays value of 0. Other two buttons are for increment and reset. By clicking increment button each time, increase the value of the button by 1. By clicking the reset button set the value of button to 0. Confirm your code by clicking the buttons.

Code:

```
<!DOCTYPE html>
<html>
<head>
    <title>Increment and Reset Value</title>
```



```
</head>
<body>
<button id="displayButton">Value: 0</button>
<button id="incrementButton">Increment</button>
<button id="resetButton">Reset</button>

<script>
// Get references to the buttons
const displayButton = document.getElementById("displayButton");
    const incrementButton = document.getElementById("incrementButton");
const resetButton = document.getElementById("resetButton");

// Initialize a variable to store the value
let value = 0;

// Function to update and display the value
function updateValue() {
displayButton.textContent = `Value: ${value}`;
}

// Add an event listener to the increment button
incrementButton.addEventListener("click", function () {
// Increment the value by 1
value++;
updateValue();
});

// Add an event listener to the reset button
resetButton.addEventListener("click", function () {
// Reset the value to 0
value = 0;
updateValue();
});

// Initialize the display with the initial value
updateValue();
</script>
</body>
</html>
```

Live server:

---

Value: 5   Increment   Reset

Value: 0   Increment   Reset

create a dynamic input filter with JavaScript. Type a search term in the input field. The displayed items in the list should match your search term. The rest of the list elements should be hidden.

Code:

```
<!DOCTYPE html>
<html>
<head>
<title>Dynamic Input Filter</title>
</head>
<body>
    <input type="text" id="searchInput" placeholder="Search..." />
    <ul id="itemList">
    <li>Frame</li>
    <li>Tires</li>
    <li>Brakes</li>
    <li>Gears</li>
    <li>Handlebars</li>
    <li>Saddle</li>
    <li>Chain</li>
    <li>Pedals</li>
    </ul>

    <script>
// Get references to the input field and list
const searchInput = document.getElementById("searchInput");
    const itemList = document.getElementById("itemList");
const items = itemList.getElementsByTagName("li");

    // Add an event listener to the input field for input changes
searchInput.addEventListener("input", function () {
```

```

        const searchTerm = searchInput.value.toLowerCase();

// Loop through the list items
for (let i = 0; i < items.length; i++) {
  const item = items[i];
  const itemName = item.textContent.toLowerCase();

  // Check if the search term is found in the item's name
  if (itemName.includes(searchTerm)) {
    item.style.display = "block"; // Show matching items
  } else {
    item.style.display = "none"; // Hide non-matching items
  }
}
});
</script>
</body>
</html>

```

live server:

---



Frame  
 Tires  
 Brakes  
 Gears  
 Handlebars  
 Saddle  
 Chain  
 Pedals

Create 10 balloons as shown below. Every time you hover over a balloon, it should become invisible. Your goal is to pop all the balloons one after the other. Create a refresh button. After clicking refresh button it will again display all the balloons

Code:

```

<!DOCTYPE html>
<html>
<head>
  <title>Pop the Balloons</title>

```

```
<style>
body {
display: flex;
height: 100vh;
width: 100vw;
overflow-x: hidden;
align-items: center;
justify-content: center;
flex-wrap: wrap;
flex-direction: column;
}
#balloonContainer {
background-color: rgb(243, 130, 235);
width: 400px;
padding: auto auto;
}

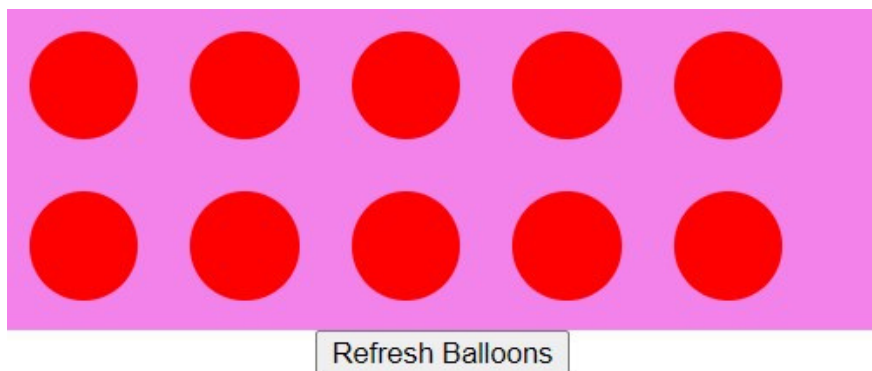
.balloon {
width: 50px;
height: 50px;
background-color: red;
border-radius: 50%;
display: inline-block;
margin: 10px;
cursor: pointer;
}
</style>
</head>
<body>
<div id="balloonContainer">
<div class="balloon"></div>
<div class="balloon"></div>
<div class="balloon"></div>
<div class="balloon"></div>
<div class="balloon"></div>
<div class="balloon"></div>
<div class="balloon"></div>
<div class="balloon"></div>
<div class="balloon"></div>
<div class="balloon"></div>
</div>
<button id="refreshButton">Refresh Balloons</button>

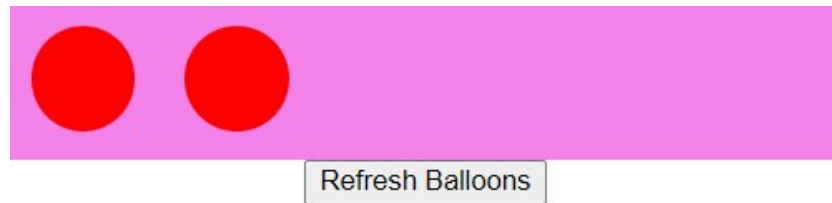
<script>
const balloons = document.querySelectorAll(".balloon");
const refreshButton = document.getElementById("refreshButton");

// Function to make a balloon disappear
```

```
function popBalloon(event) {  
  event.target.style.display = "none";  
}  
  
// Add a hover event listener to each balloon  
balloons.forEach((balloon) => {  
  balloon.addEventListener("mouseenter", popBalloon);  
});  
  
// Add a click event listener to the refresh button  
refreshButton.addEventListener("click", () => {  
  balloons.forEach((balloon) => {  
    balloon.style.display = "inline-block";  
  });  
});  
</script>  
</body>  
</html>
```

Live server:





Create a function move that moves the button 1px to the left or the right. It is recursive because it calls itself again and again. This keeps the button moving. Extend the JavaScript code. Once you click the button, it should stop moving. When you click it again, it should move again.

Code:

```
<!DOCTYPE html>
<html>
<head>
<title>Move Button</title>
<style>
#movingButton {
width: 100px;
height: 30px;
background-color: #3498db;
color: #fff;
font-size: 16px;
border: none;
border-radius: 5px;
cursor: pointer;
position: relative;
left: 0;
transition: left 0.1s linear;

#buttonContainer {
text-align: center;
margin-top: 50px;

</style>
</head>
<body>
<div id="buttonContainer">
  <button id="movingButton" onclick="toggleMove()">Move</button>
```

```
</div>

<script>
let movingInterval; // Variable to store the interval ID
// Initial direction

function move() {
const button = document.getElementById("movingButton");
const currentPosition = parseFloat(getComputedStyle(button).left);
const newPosition = movingLeft
? currentPosition - 1
: currentPosition + 1;
button.style.left = newPosition + "px";

// Check if the button has reached the edge of the container
const containerWidth = document.body.clientWidth - button.clientWidth;
if (newPosition <= 0 || newPosition >= containerWidth) {
// Change direction when reaching the edge
movingLeft = !movingLeft;

function startMoving() {
movingInterval = setInterval(move, 10); // Move every 10 milliseconds

function stopMoving() {
clearInterval(movingInterval);

function toggleMove() {
const button = document.getElementById("movingButton");
if (movingInterval) {
stopMoving();
button.textContent = "Move";
} else {
startMoving();
button.textContent = "Stop";

// Start moving initially
startMoving();
</script>
</body>
</html>
```

Live server:

Move