

Walchand College of Engineering, Sangli

Department of Computer Science and Engineering

Batch: T6

Practical No.: 4

Title of Assignment: Study of Java Script and DOM.

Student Name: Harshavardhan Bamane

Student PRN: 22510112

Problem Statement 0: Basics of DOM

1. What is the DOM?

Ans:

The **DOM (Document Object Model)** is a programming interface for web documents. It represents the structure of a webpage in a way that programming languages like JavaScript can interact with it. Think of it as a tree where each branch is a part of the webpage, like text, images, or buttons.

2. What is DOM Tree Structure? Elaborate its elements with example (you may create DOM tree structure of previously created web pages)

Ans:

The **DOM Tree Structure** is a hierarchical model representing the elements of a webpage. Each element of the webpage (like headings, paragraphs, and images) is a node in this tree.

- **Elements of DOM Tree:**

- **Document Node:** The root node representing the entire document.
- **Element Nodes:** These are HTML tags like `<div>`, `<p>`, and `<h1>`.
- **Text Nodes:** The actual text inside an element.
- **Attribute Nodes:** The attributes of HTML tags, like `class` or `id`.
- **Comment Nodes:** Comments added in the HTML document.

3. Give examples for following:

o Accessing the DOM

o Manipulating the DOM

o Event Handling

o Traversing the DOM

Ans.

1. Accessing the DOM:

`document.getElementById("myElement");` // Access an element with a specific ID

2. Manipulating the DOM:

`document.getElementById("myElement").innerHTML = "New Content";` // Change the content of an element

3. Event Handling:

`document.getElementById("myButton").addEventListener("click", function() { alert("Button clicked!"); });` // Add a click event to a button

4. Traversing the DOM:

`var parent = document.getElementById("myElement").parentNode;` // Get the parent node of an element

4. What are Performance Considerations while implementing the DOM and can DOM supports all browsers?

Ans.

Performance Considerations:

- Minimize DOM Manipulations: Frequent changes to the DOM can slow down the performance, especially in large documents.
- Batch Updates: Combine several updates into one to reduce the number of reflows (page layout recalculations).
- Avoid Layout Thrashing: Minimize the number of times you read and write to the DOM consecutively, as this can cause performance issues.

Browser Support:

- The DOM is supported by all modern browsers. However, older versions may have different levels of support for certain DOM methods and properties. Using feature detection or libraries like jQuery can help handle these differences.

5. Elaborate Common Methods and Properties of DOM

Ans.

Common Methods:

- `getElementById(id)`: Returns the element with the specified ID.
- `getElementsByClassName(className)`: Returns a collection of all elements with the specified class.
- `querySelector(selector)`: Returns the first element that matches a specified CSS selector.
- `createElement(tagName)`: Creates a new HTML element.
- `appendChild(node)`: Adds a new child node to a parent node.

Common Properties:

- `innerHTML`: Gets or sets the HTML content of an element.
- `textContent`: Gets or sets the text content of an element.
- `className`: Gets or sets the class attribute of an element.
- `style`: Allows you to get or set the inline style of an element.
- `attributes`: Returns a collection of all attributes of an element.

Problem Statement 1: DOM selector methods

Here, the 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.

Ans.



The screenshot shows a web application interface with a dark theme. At the top right is a 'reset' button. The main UI consists of a white text input field containing 'OFF' and an orange button labeled 'Click Me'. Below the UI is a 'View' tab showing the HTML structure, and a 'Javascript' tab showing the JavaScript code.

HTML

```
<div id="wrapper">
  <input type="text" value="OFF" readonly/>
  <button type="button">Click Me</button>
</div>
```

Javascript

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

Ans.

The screenshot displays a web application interface with a list of six toggle switches. The first two are labeled 'OFF' and the last four are labeled 'ON'. A 'reset' button is located to the right of the list. Below the list, the HTML and JavaScript code are shown.

View

OFF
OFF
ON
ON
ON
ON

HTML

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

Javascript

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

const handleHover = (event) => {
  return event.target.innerText = 'ON';
};

if(listItems.length > 1) {
  listItems.forEach(item => item.addEventListener('mouseover', handleHover));
}
```

Problem Statement 2: Events and user interactions

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

Ans.

The image shows a web development environment with three panels: View, HTML, and Javascript.

View Panel: Displays a user interface with a white text input field and an orange button labeled "Click Me". A "reset" button is located in the top right corner of the panel.

HTML Panel: Contains the following HTML code:

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

Javascript Panel: Contains the following JavaScript code:

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

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

// type in your code here
button.addEventListener('click', () => {
  handleClick();
})
```

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.

Ans.



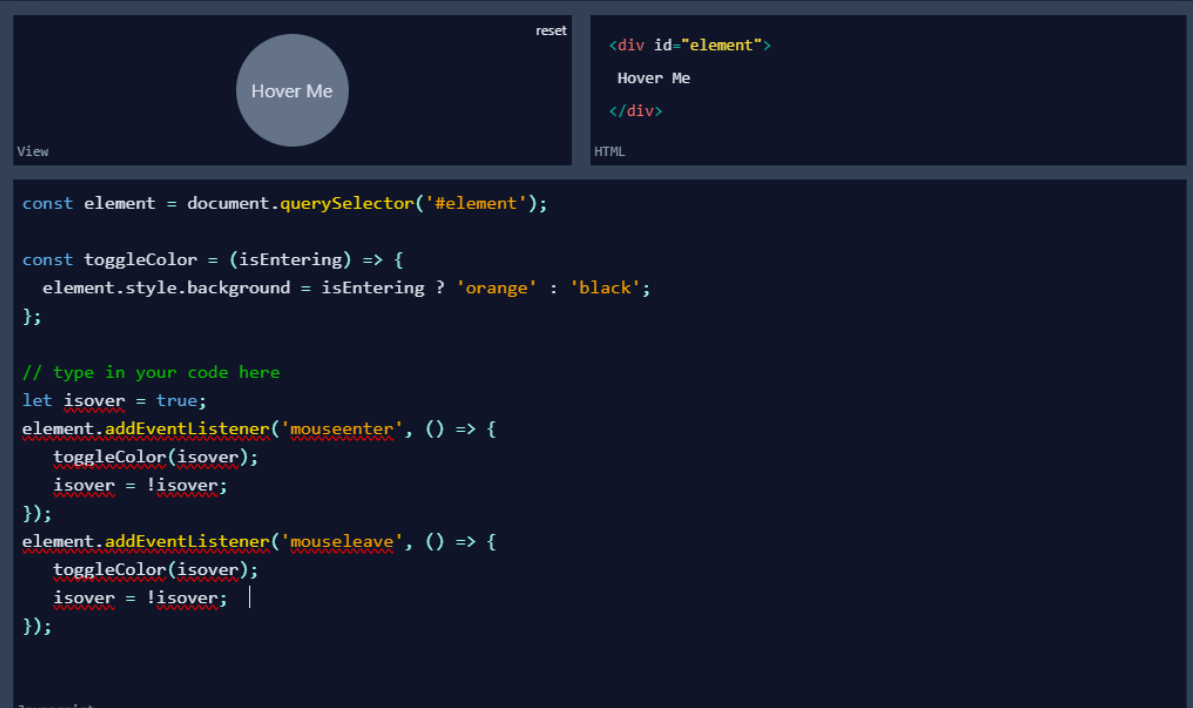
```
const element = document.getElementById('element');

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

// type in your code here
element.addEventListener('mouseover', 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

Ans.



The image shows a web development interface with three panels. The top-left panel, labeled 'View', displays a dark blue circle with the text 'Hover Me' inside. A 'reset' button is located in the top right corner of this panel. The top-right panel, labeled 'HTML', contains the following code:

```
<div id="element">
  Hover Me
</div>
```

. The bottom panel, labeled 'Javascript', contains the following code:

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

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

// type in your code here
let isover = true;
element.addEventListener('mouseenter', () => {
  toggleColor(isover);
  isover = !isover;
});
element.addEventListener('mouseleave', () => {
  toggleColor(isover);
  isover = !isover;
});
```


Problem Statement 3: DOM manipulation with JavaScript

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.



Problem Statement 4: DOM fundamentals

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.



Check me!

Click to Check

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.



Check Me

Harsh

Bamane

Harsh Bamane

Full Name

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.

Ans.

7

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.

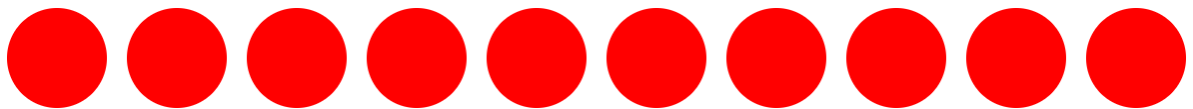
Ans.

- Apple
- Banana
- Orange
- Mango
- Grapes
- Pineapple
- Strawberry

- Orange

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.

Ans.



Refresh Balloons



Refresh Balloons

Problem Statement 5: Recursive functions

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.