

Experiment - 2

Aim: To write simple JavaScript programs to perform arithmetic operations, manipulate strings, and use conditional statements and loops.

Theory:

1. Introduction to JavaScript:

JavaScript (JS) is a versatile, high-level programming language primarily used for web development. It enables interactive web pages by manipulating the Document Object Model (DOM), handling events, and performing client-side computations. JS is widely supported by modern browsers and can be used for both front-end and back-end development with environments like Node.js.

2. Different Ways to Include JavaScript in an HTML File:

JavaScript can be included in an HTML file in three ways:

- **Inline JavaScript:** JavaScript code is written directly inside an HTML element using the `onclick` or `onload` attributes.

For eg: `<button onclick="alert('Hello, World!')">Click Me</button>`

- **Internal JavaScript:** JavaScript code is written inside a `<script>` tag within an HTML file.

For eg: `<script> console.log("Hello, World!"); </script>`

- **External JavaScript:** JavaScript code is written in a separate `.js` file and linked using the `<script>` tag.

For eg: `<script src="script.js"></script>`

3. Data Types in JavaScript:

JavaScript provides several data types, categorized into **primitive** and **non-primitive** types:

a. Primitive Data Types:

- Number: `let num = 10;`
- String: `let name = "John";`
- Boolean: `let isActive = true;`

- Null: `let y = null;`
- Symbol: `let sym = Symbol("id");`
- BigInt: `let bigNum = 12345678901234567890123n;`

b. Non-Primitive Data Types:

- Object: `let person = {name: "Alice", age: 25};`
- Array: `let colors = ["red", "blue", "green"];`
- Function: `function greet() { console.log("Hello"); }`

4. Variables in JavaScript:

Variables store data and are declared using `var`, `let`, or `const`:

- `var`: Function-scoped, can be redeclared.
- `let`: Block-scoped, cannot be redeclared within the same scope.
- `const`: Block-scoped, cannot be reassigned.

5. Conditional Statements:

Conditional statements control the execution flow based on conditions.

- IF statement:

```
if (age > 18) {
  console.log("Adult");
}
```
- IF-ELSE statement:

```
if (age >= 18) {
  console.log("Adult");
} else {
  console.log("Minor");
}
```
- ELSE-IF statement: *if (score >= 90) { console.log("A grade"); } else if (score >= 80) { console.log("B grade"); } else { console.log("C grade"); }*
- SWITCH statement:

```
switch(day) { case "Monday": console.log("Start of the week");
break; case "Friday": console.log("Weekend is near"); break;
default: console.log("Regular day"); }
```

}

6. Flow-Control Statements:

Flow-control statements help manage iterations:

- **FOR Loop:**

```
for (let i = 0; i < 5; i++) {  
  console.log(i);  
}
```
- **WHILE Loop:**

```
let i = 0; while (i < 5) { console.log(i); i++;  
}
```
- **DO-WHILE Loop:**

```
let i = 0; do { console.log(i); i++;  
} while (i < 5);
```
- **BREAK:** It is used to exit a loop.

```
for (let i = 0; i < 5; i++) { if  
(i === 3) break;  
  console.log(i);  
}
```
- **CONTINUE:** It is used to skip the current iteration.

```
for (let i = 0; i < 5; i++) { if  
(i === 2) continue;  
  console.log(i);  
}
```

7. Functions:

Functions encapsulate re-usable logic in JavaScript. Types of Function Declarations:

- **Regular Function:** A named function that can be called anywhere in the code after declaration.

```
function greet() {  
  console.log("Hello, World!");  
}
```
- **Function Expression:** A function assigned to a variable, which cannot be called before its definition

```
const greet = function() { console.log("Hello!");  
};
```

- **Arrow Function:** A concise way to write functions using `=>`, commonly used for shorter expressions.

`const greet = () => console.log("Hello!");`

8. DOM Manipulation:

JavaScript allows modifying HTML elements dynamically using the Document Object Model (DOM):

- Selecting elements: *`let heading = document.getElementById("title");`*
- Modifying content: *`heading.innerHTML = "New Title";`*
- Changing styles: *`heading.style.color = "blue";`*

9. Event Listeners:

Event listeners allow handling user interactions dynamically: For eg:

Click Event

*`document.getElementById("btn").addEventListener("click", function() {
alert("Button Clicked!");
});`*

10. Some built-in JS Classes:

- **Math Class:** Provides mathematical functions like square root, random numbers, and rounding.

*`console.log(Math.sqrt(25)); // 5 console.log(Math.random()); // Random
number between 0 and 1`*

- **Date Class:** Represents date and time, allowing operations like getting current date, time, and formatting. *`let currentDate = new Date();`*

*`console.log(currentDate.toString()); // Displays current date in
String form`*

Codes:

1. Digital Clock:

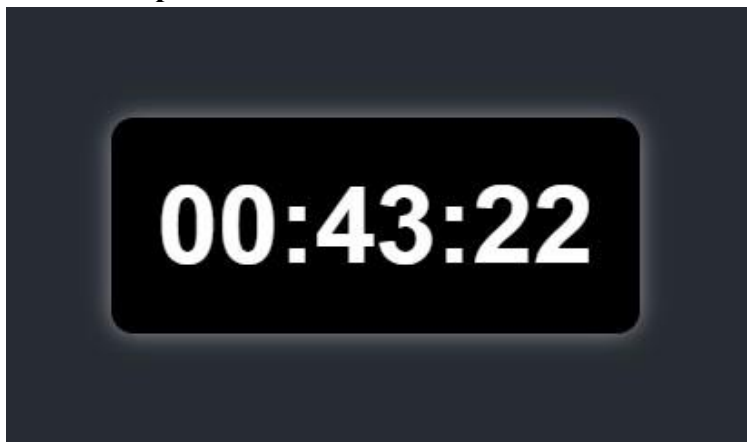
HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">      <meta name="viewport"
content="width=device-width, initial-scale=1.0">  <title>Digital
Clock</title>
  <link rel="stylesheet" href="clock.css">  <style>
body {
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  background-color: #282c34;      color: white;
  font-family: Arial, sans-serif;
}
#clock {
  font-size: 3rem;
  font-weight: bold;
  background: black;
  padding: 20px;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(255, 255, 255, 0.5);
}
</style>
</head>
<body>
  <div id="clock">00:00:00</div>
  <script src="clock.js"></script>
</body>
</html>
```

JS File:

```
function updateClock() {  const now = new Date();
let hours = now.getHours().toString().padStart(2, '0');
let minutes = now.getMinutes().toString().padStart(2, '0');
let seconds = now.getSeconds().toString().padStart(2, '0');
document.getElementById('clock').innerText =
`$${hours}:$${minutes}:$${seconds}`; }
setInterval(updateClock, 1000); updateClock();
```

Output:



2. Changing BG Color every 5 seconds:

HTML file:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">      <meta name="viewport"
content="width=device-width,  initial-scale=1.0">
<title>Changing BG</title>  <style>    body {      display:
flex;      justify-content: center;      align-items: center;
height: 100vh;      background-color: #282c34;      color:
white;      font-family: Arial, sans-serif;      transition:
background-color 1s;    }
</style>
```

```

</head>
<body>
  <h1>Background Color changes every 5 sec!</h1>
  <script src="changeBGColor.js"></script>
</body>
</html>

```

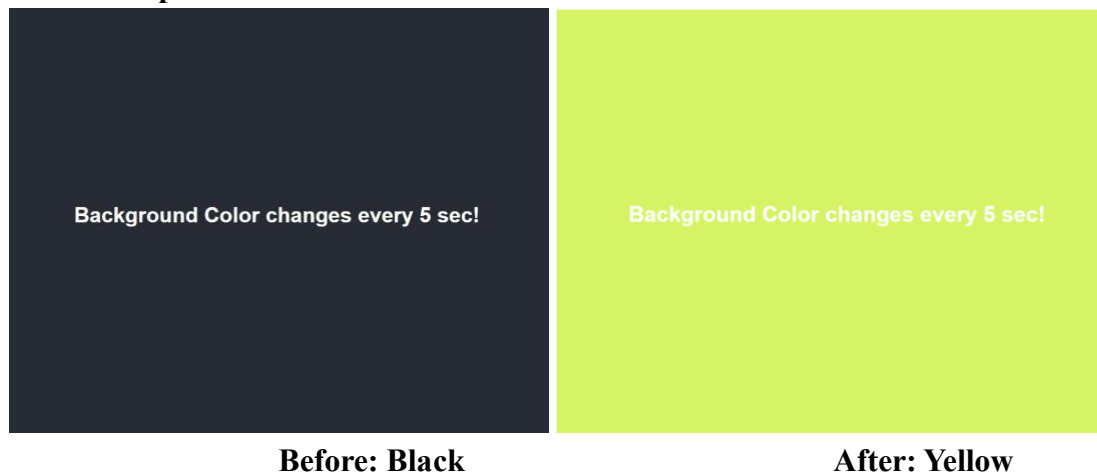
JS File:

```

function changeBackgroundColor() {      function randomNum() {
return Math.floor(Math.random()*256);    }      const randomColor =
`rgb(${randomNum()},    ${randomNum()},    ${randomNum()})`;
document.body.style.backgroundColor = randomColor; }
setInterval(changeBackgroundColor, 5000);

```

Output:



3. Calculator:

HTML file:

```

<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">          <meta name="viewport"
content="width=device-width,      initial-scale=1.0">
<title>Calculator</title>

```

```

</head>
<body>
  <label for="inp1">Number 1: </label>
  <input type="number" id="inp1">
  <br><br>
  <label for="inp2">Number 2: </label>
  <input type="number" id="inp2">
  <br><br>
  <button id="add">Add</button>&nbsp;&nbsp;&nbsp;<button
id="sub">Subtract</button>&nbsp;&nbsp;&nbsp;<button
id="mul">Multiply</button>&nbsp;&nbsp;&nbsp;<button
id="div">Divide</button>&nbsp;&nbsp;&nbsp;<button id="clr">Clear</button>
<br><br>
  <p>Output: <span id="result"></span></p>
  <script src="calculator.js"></script>
</body>
</html>

```

JS File:

```

let inp1 = document.querySelector("#inp1"); let inp2 =
document.querySelector("#inp2"); let add =
document.querySelector("#add");
let sub = document.querySelector("#sub");
let mul = document.querySelector("#mul");
let div = document.querySelector("#div");
let res = document.querySelector("#result"); let clr =
document.querySelector("#clr");
add.addEventListener("click", () => {
let output = parseFloat(inp1.value) + parseFloat(inp2.value); res.innerText = output;
});
sub.addEventListener("click", () => {
let output = parseFloat(inp1.value) - parseFloat(inp2.value); res.innerText = output;
}); mul.addEventListener("click", () => { let output = parseFloat(inp1.value)
* parseFloat(inp2.value); res.innerText = output; });

```



```
div.addEventListener("click", () => {    let output = parseFloat(inp1.value) /
    parseFloat(inp2.value);                res.innerText = output; });
clr.addEventListener("click", () => {    res.innerText = ""; });
```

Output:

Number 1:

Number 2:

Output: 72

4. Toggle Bulb:

HTML File:

```
<html lang="en">
<head>
    <meta charset="UTF-8">                <meta name="viewport"
    content="width=device-width,          initial-scale=1.0">
    <title>Bulb</title>
</head>
<body>
    <div style="text-align: center; display: block;" id="one">
        
        <br><br>
        <button id="on">Turn On</button>
    </div>
    <div style="text-align: center; display: none;" id="two">
        <imgsrc="https://th.bing.com/th/id/OIP.VrhKwCo-
B9IcANh6aLl_qAHaHa?w=198& h=198&c=7&r=0&o=5&pid=1.7" alt="bulb-off"
style="margin-top: 2rem;">
```

```

<br><br>
  <button id="off">Turn Off</button>
</div>
<script src="bulb.js"></script>
</body>
</html>

```

JS File:

```

let one = document.querySelector("#one"); let two =
document.querySelector("#two"); let on =
document.querySelector("#on");
let off = document.querySelector("#off");
on.addEventListener("click", () => {
  one.style.display = "none";
  two.style.display = "block"; });
off.addEventListener("click", () => {
  one.style.display = "block";
  two.style.display = "none"; });

```

Output:



Turn On

Off state



Turn Off

On state

5. Hide & Show paragraph:

HTML File:

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,initial-
scale=1.0">
  <title>Hide & Show</title>
  <style>
  body {
    height: 100vh;
    background-color: #282c34;
    color: white;
    font-family: Arial, sans-serif;
  }
  div {
    margin: 20% auto;
    justify-content: center;
    height: 100%;
    width: 35%;
    text-align: center
  }
</style>
</head>
<body>
  <div>
    <h1>This Page Hides and Shows the below Paragraph based on which button
is pressed!</h1>
    <p>Lorem, ipsum dolor sit amet consectetur adipisicing elit. Minus corporis
tempora eum dolor culpa doloribus architecto, animi praesentium sapiente
quo recusandae quos ab porro tempore. Quae suscipit unde cum quam.</p>
    <button id="show">Show</button>
    <button id="hide">Hide</button>
  </div>
  <script src="Hide&Show.js"></script>
```

```
</body>
```

```
</html>
```

JS File:

```
let p = document.querySelector('p');  
let show = document.querySelector('#show'); let hide  
= document.querySelector('#hide');  
show.addEventListener('click', function() {  
  p.style.opacity = 1; });  
hide.addEventListener('click', function() {  
  p.style.opacity = 0; });
```

Output:

**This Page Hides and Shows the
below Paragraph based on which
button is pressed!**

Lorem, ipsum dolor sit amet consectetur adipisicing elit. Minus corporis
tempora eum dolor culpa doloribus architecto, animi praesentium sapiente
quo recusandae quos ab porro tempore. Quae suscipit unde cum quam.

Show Hide

**This Page Hides and Shows the
below Paragraph based on which
button is pressed!**

Show Hide

6. Adding Interactivity to the webpage:

HTML File: I have added an extra card to demonstrate the working of the **Delete** button. Alongwith the delete button, I have also added the **Show** button, which redirects the user to the ‘**Details**’ page of that particular item. I have used form to achieve the result.

```
<div class="container">
  <div class="gallery">
    <div class="gallery-item">
      
      <h3>Antique Vase</h3>
      <p>A beautiful antique vase from the 18th century.</p>
      <button>Show</button>
      <button class="delete">Delete</button>
    </div>
    <div class="gallery-item">
      
      <h3>Antique Vase</h3>
      <p>A beautiful antique vase from the 18th century.</p>
      <button>Show</button>
      <button class="delete">Delete</button>
    </div>
    <div class="gallery-item">
      
      <h3>Antique Clock</h3>
      <p>An exquisite antique clock with intricate design.</p>
      <form action="/AntiqueClock.html">
        <button>Show</button>
      </form>
      <button class="delete">Delete</button>
    </div>
  </div>
</div>
```

```

<div class="gallery-item">
  
  <h3>Antique Painting</h3>
  <p>A rare painting from the Renaissance period.</p>
  <button>Show</button>
  <button class="delete">Delete</button>
</div>
</div>
</div>

```

JS File:

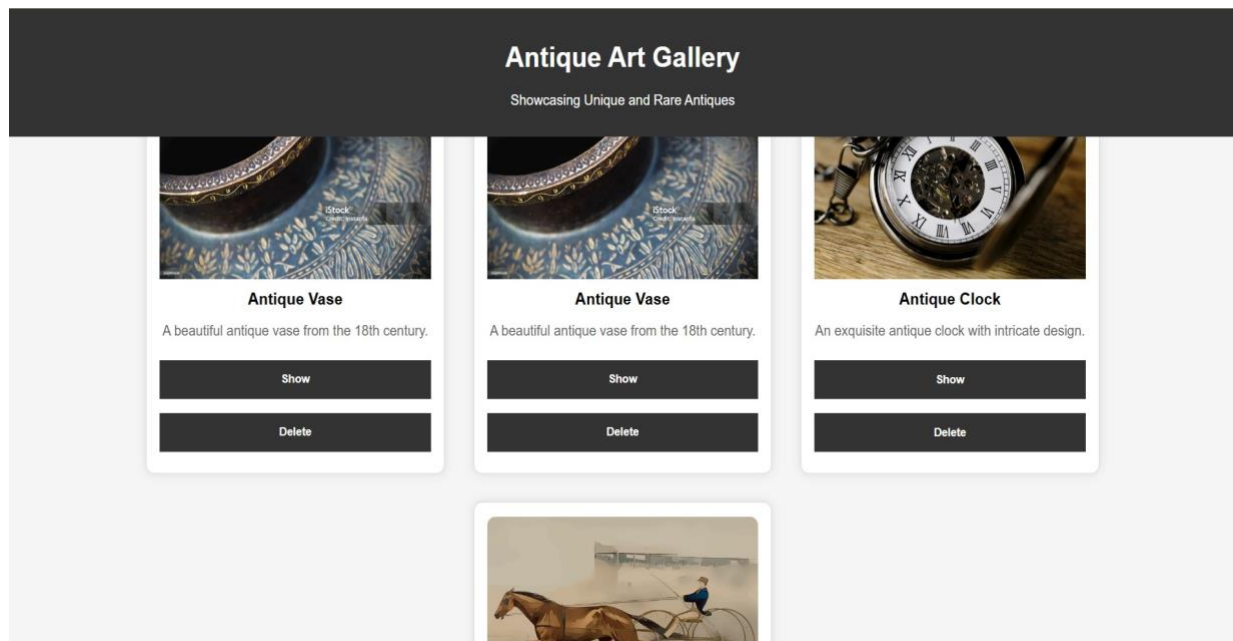
```

let delbtns = document.querySelectorAll('.delete'); for (delbtn
of delbtns) { delbtn.addEventListener('click', function() {
console.log('delete'); this.parentElement.remove();
});
}

```

Output:


1. At first, there are 4 cards on the webpage.



2. Clicking on the delete button, on the 2nd card - this deletes the card from the webpage. So, now there are only 3 cards left.

Antique Art Gallery

Showcasing Unique and Rare Antiques




Antique Vase

A beautiful antique vase from the 18th century.

Show

Delete




Antique Clock

An exquisite antique clock with intricate design.

Show

Delete



Antique Painting

A rare painting from the Renaissance period.

Show

Delete

© 2025 Antique Art Gallery. All rights reserved.

3. Clicking on the Show button, on the middle card - this redirects the user to a new webpage, showing complete details of the selected card.

Antique Clock



Description: An exquisite antique clock with intricate design. It is speculated to be more than 100 years old. It was used by Sailors and Travellers in ancient time, to check time. Due to it's compact size, it was very handy to carry. Only 100 of these pieces are known to exist in the world!

Price: \$1500

Availability: In Stock

Location: Rome Int. Museum, Italy

Go Back

Conclusion: This experiment provided a comprehensive understanding of JavaScript fundamentals, including arithmetic operations, string manipulation, conditional statements, loops, functions, and DOM manipulation. By implementing these concepts, we learned how to enhance web interactivity, handle user inputs efficiently, and perform essential mathematical and date-related operations. Mastering these core JavaScript features lays the foundation for building dynamic and responsive web applications, making it an indispensable skill for modern web development.